Multimodal Trajectory Prediction via Topological Invariance for Navigation at Uncontrolled Intersections

Junha Roh[*], **Christoforos Mavrogiannis**[*], **Rishabh Madan**[*]
Dieter Fox, Siddhartha S. Srinivasa
Paul G. Allen School, University of Washington, USA
`{rohjunha, cmavro, rishabhm, fox, siddh}@cs.washington.edu`

Abstract: We focus on decentralized navigation among multiple non-communicating rational agents at *uncontrolled* intersections, i.e., street intersections without traffic signs or signals. Avoiding collisions in such domains relies on the ability of agents to predict each others' intentions reliably, and react quickly. Multiagent trajectory prediction is NP-hard whereas the sample complexity of existing data-driven approaches limits their applicability. Our key insight is that the geometric structure of the intersection and the incentive of agents to move efficiently and avoid collisions (rationality) reduces the space of likely behaviors, effectively relaxing the problem of trajectory prediction. In this paper, we collapse the space of multiagent trajectories at an intersection into a set of *modes* representing different classes of multiagent behavior, formalized using a notion of topological invariance. Based on this formalism, we design *Multiple Topologies Prediction* (MTP), a data-driven trajectory-prediction mechanism that reconstructs trajectory representations of high-likelihood modes in multiagent intersection scenes. We show that MTP outperforms a state-of-the-art multimodal trajectory prediction baseline (MFP) [1] in terms of prediction accuracy by 78.24% on a challenging simulated dataset. Finally, we show that MTP enables our optimization-based planner, MTPnav, to achieve collision-free and time-efficient navigation across a variety of challenging intersection scenarios on the CARLA simulator. A link to our code implementation and demo videos can be found at this website.

Keywords: Trajectory prediction, multiagent navigation, topological methods

1 Introduction

The widespread interest in autonomous driving technology in recent years [2] has motivated extensive research in multiagent navigation in driving domains. One of the most challenging driving domains [3] is the *uncontrolled* intersection, i.e., a street intersection that features no traffic signs or signals. Within this domain, we focus on scenarios in which agents do not communicate explicitly or implicitly through e.g., turn signals. This model setup gives rise to challenging multi-vehicle encounters that mimic real-world situations (arising due to human distraction, violation of traffic rules or special emergencies) that result in fatal accidents [3]. The frequency and severity of such situations has motivated vivid research interest in uncontrolled intersections [4, 5, 6].

In the absence of explicit traffic signs, signals, rules or explicit communication among agents, avoiding collisions at intersections relies on the ability of agents to predict the dynamics of interaction amongst themselves. One prevalent way to model multiagent dynamics is via trajectory prediction. However, multistep multiagent trajectory prediction is NP-hard [7], whereas the sample complexity of existing learning algorithms effectively prohibits the extraction of practical models. Our key insight is that the geometric structure of the intersection and the incentive of agents to move efficiently and avoid collisions with each other (rationality) compress the space of possible multiagent trajectories, effectively simplifying inference.

[*]Denotes equal contribution.

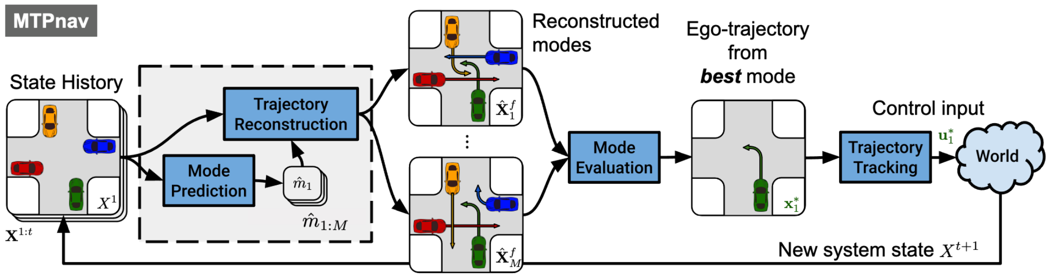


Figure 1: **Overview of our decentralized navigation planner, MTPnav:** The ego-vehicle (shown in green) invokes MTP (Multiple Topologies Prediction), our trajectory prediction mechanism, which determines M *modes* of highly likely future multiagent behavior, and reconstructs trajectory representatives for them. These modes are then evaluated based on the quality of their representatives incorporating considerations such as efficiency, clearance, likelihood, and smoothness. The ego-trajectory from the best mode is then passed to a controller which tracks the first waypoint.

In this paper, we explicitly account for this structure by collapsing the space of multiagent trajectories at an intersection into a finite set of *modes* of joint behavior. We represent modes using a notion of topological invariance [8]: all trajectories belonging to the same mode leave the same topological signature (Fig. 2a depicts an example). Leveraging the flexibility and expressiveness of Graph Neural Networks [9], we present a trajectory-prediction architecture that biases inference towards high-likelihood modes. We show that our architecture, entitled *Multiple Topologies Prediction* (MTP), outperforms a state-of-the-art baseline (MFP [1]) by 78.24% in terms of reconstruction accuracy on a dataset of challenging intersection crossing tasks. Based on MTP, we design MTPnav, an optimization-based planning framework (see Fig. 1) which achieves time-efficient, collision-free navigation across a variety of challenging multiagent intersection-crossing scenarios in the CARLA simulation environment [10].

2 Related Work

In recent years, the end goal of autonomous driving has motivated extensive research on prediction and planning for navigation in multiagent domains. Considerable attention focuses on the task of multiagent trajectory prediction. Earlier works in the area employed physics-based models [11], hidden Markov models [12], and dynamic Bayesian networks [13]. With the advent of deep learning and the availability of several public trajectory datasets [14, 15], many recent works are using Recurrent Neural Networks [16, 17, 1] and graph-based models [18, 19, 20]. For instance, Deo and Trivedi [17] use social pooling layers along with maneuver-based trajectory generation by labeling trajectories with maneuvers, which leads to multimodal predictions. Li et al. [18] make use of graph-convolutional networks combined with an LSTM-based encoder-decoder to account for inter-vehicle interactions. A recent line of work employs models for *multimodal* trajectory prediction in an effort to account for the uncertainty in multiagent navigation [21, 22, 1, 23, 24]. For example, Tang and Salakhutdinov [1] propose the Multiple Futures Prediction (MFP) method for multimodal multiagent trajectory predictions by capturing different behaviors present in the data using discrete latent variables in an unsupervised manner. Our work also embraces the virtues of multimodality but differs by incorporating a mathematically sound, compact and interpretable formalism for representing multimodality using concepts from topology to drive the learning process.

In parallel, relevant works on planning and control are focusing on integrating models of multiagent dynamics in the decision making. Some works employ implicit models of interaction, such as risk level sets [25], deep reinforcement learning [4], or imitation learning [26]. While such approaches may yield desirable performance in interesting driving scenes, they abstract away the richness of interaction that unfolds in driving domains, thus showing limited applicability, and they are often further constrained by data dependencies. In an effort to explicitly model interaction, some works model multiagent dynamics as *games* by employing tools from game theory [27, 6]. While game-theoretic approaches elegantly capture the richness of interactions in driving domains, the challenge of computing (Nash) equilibria and the intractability of scaling to large numbers of agents limits their applicability. Other works have employed topological abstractions such as braids [5, 28] to enable agents to reason over multiagent collision-avoidance strategies. Topological braids capture critical events in multiagent navigation, such as the order of passing maneuvers, in a compact and interpretable fashion [29, 30]; however, their lack of analytic descriptions complicates the construction of generalizable prediction models and so limits their applicability. Instead of braids, our work leverages a notion of topological invariance [8] with analytic descriptions that enable the use of data-driven methods. This allows us to retain the benefits of topological reasoning while achieving state-of-the-art performance in challenging and realistic multiagent intersection-crossing tasks.

3 Problem Statement

Consider a four-way uncontrolled intersection (see Fig. 2a), where no explicit rules (e.g., right-turn priority) or signals (e.g., traffic lights) are set in place to regulate traffic. Assume that $1 < n \leq 4$ non-communicating agents with *simple-car* kinematics are navigating. Denote by $x_i \in \mathcal{X} \subseteq SE(2)$ the state of agent $i \in \mathcal{N} = \{1, \ldots, n\}$ with respect to (wrt) a fixed reference frame. Agent i is tracking a path $\tau_i : I \to \mathcal{X}$, for which it holds that $\tau_i(0) = s_i$ and $\tau_i(1) = d_i$, where $s_i, d_i \in \mathcal{X}$ are states lying at different sides of the intersection (i.e., right, left, up, down) and $I = [0, 1]$ is a path parameterization. At timestep t (assume a fixed time discretization dt), agent i executes a policy $\pi_i : \mathcal{X} \to \mathcal{U}_i$ that generates controls $u_i \in \mathcal{U}_i$ (speed and steering angle) contributing progress along τ_i while avoiding collisions with $j \neq i$. Agent i is not aware of the intended path τ_j, the destination d_j, or the policy π_j of agent $j \neq i \in \mathcal{N}$.

Let agent #1 be the *ego* agent. The ego agent perfectly observes the current system state $X^t = (x_1^t, \ldots, x_n^t) \in \mathcal{X}^n$ and has access to the complete system state history $\mathbf{X}^{1:t} = (X^1, \ldots, X^t)$. Our goal is to design a policy π_1 that enables the ego agent to follow collision-free and time-efficient intersection crossings despite the uncertainty resulting from the constraint of no communication.

4 Navigation with Multiple Topologies Prediction

We describe a policy π_1 for decentralized navigation at uncontrolled intersections. Our policy is based on a data-driven mechanism for multimodal multiagent trajectory prediction. Our mechanism leverages a topological formalism that effectively compresses the space of possible multiagent trajectories into a set of *modes* of multiagent behavior. This allows us to guide trajectory prediction towards high-likelihood modes. At planning time, our policy evaluates a set of high-likelihood multiagent trajectory alternatives and follows the ego trajectory from the one with minimum cost.

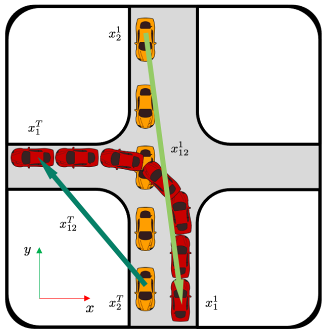
4.1 A Topological Formalism of Modes for Navigation at Intersections

Consider two agents navigating a four-way intersection. Denote by $x_{12}^t = x_1^t - x_2^t$ a vector–referred henceforth as the *winding vector*– expressing the relative location of agents and by θ_{12}^t its angle with respect to a global frame at timestep $t \geq 1$ (see Fig. 2a). From time t to $t+1$, agents' displacement, $\Delta x_{12}^t = x_{12}^{t+1} - x_{12}^t$, results in a rotation $\Delta\theta_{12}^t = \theta_{12}^{t+1} - \theta_{12}^t$. Taking the sum of these rotations from the beginning of the execution ($t = 1$) until the end ($t = T$), we derive a *winding number* [8] characterizing the total relative rotation of both agents:

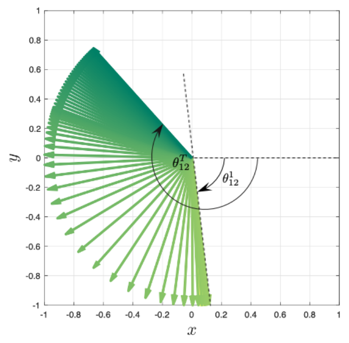
$$\lambda_{12} = \frac{1}{2\pi} \sum_{t=1}^{T-1} \Delta\theta_{12}^t. \tag{1}$$

The value λ_{12} tracks the signed number of times agents $1, 2$ "revolved" around each other from time $t = 1$ to $t = T$.

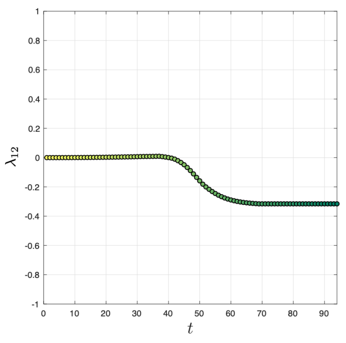
The winding number is a *topological invariant*: fixing agents' endpoints s_1, s_2, d_1, d_2, any topology-preserving deformations of agents' trajectories (continuous trajectory deformations not involving penetrations or cuts) would be identified using the same winding-number value λ_{12}. Importantly, the sign of the winding number $w_{12} = \mathrm{sgn}(\lambda_{12})$ indicates the side on which the two agents pass each other. A right-side passing corresponds to a clockwise rotation of the winding vector x_{12}, yielding a positive winding number $\lambda_{12} > 0$; a left-side passing corresponds to a counterclockwise rotation of the winding vector x_{12}, yielding a negative winding number $\lambda_{12} < 0$. Fig. 2 illustrates the machinery of the winding number at an intersection example with two agents.



(a) Top view of agents' trajectories.



(b) Unit winding vector trajectory.



(c) Winding number convergence.

Figure 2: **Identifying modes of intersection crossing:** (a) top view of agents' trajectories, indicating the initial and terminal winding vectors; (b) unit winding vector trajectory (darkness increases with time); and (c) winding number convergence. Any execution of this scenario in which the orange agent passes before the red agent converges to a negative winding number.

Extending this idea to a scene with n agents with endpoints $S = (s_1, \dots, s_n)$ and $D = (d_1, \dots, d_n)$, we define the *topology* of an intersection-crossing task as:

$$W = (\dots, w_{ij}, \dots), ij \in \mathcal{N}_e, \tag{2}$$

where $\mathcal{N}_e$ is the set of all pairs formed among n agents. Further, we define a *mode* of intersection crossing as a tuple $m = (S, D, W)$. At an intersection-crossing task, a mode encodes *where* agents are heading (destination D) and *how* they are getting there (topology W) from their initial state, S.

4.2 Multiple Topologies Prediction

Leveraging the formalism of modes, we build *Multiple Topologies Prediction* (MTP), a multimodal trajectory prediction framework. MTP biases trajectory inferences towards a set of high-likelihood modes. Given a recent state history $\mathbf{X}^p = \mathbf{X}^{t_p+1:t}$ of $h_p = t - t_p$ timesteps in the past, MTP: 1) determines a set of highly likely modes of future multiagent behavior $\hat{\mathbf{m}} = \{\hat{m}_1, \cdots, \hat{m}_M\}$; 2) predicts a corresponding set of trajectory representatives $\hat{\boldsymbol{\mathcal{X}}} = \{\hat{\mathbf{X}}_1^f, \dots, \hat{\mathbf{X}}_M^f\}$ where $\hat{\mathbf{X}}_l^f = \mathbf{X}_l^{t+1:t_f}$, $l = 1, \dots, M$, and $h_f = t_f - t$ is a horizon into the future. We construct models for the two stages of prediction using a GraphNet architecture [9]. We first describe how to reconstruct a trajectory for a desired mode m and then describe a technique for generating high-likelihood modes.

4.2.1 Mode-Conditioned Trajectory Reconstruction

Our model represents the world state at time t, X^t as a directed graph g^t (see Fig. 3). The graph g^t consists of a set of node attributes $V = \{\mathbf{v}_i^t : i \in \mathcal{N}\}$, a set of edge attributes and corresponding node indices $E = \{(\mathbf{e}_k^t, s_k, r_k) : k \in \mathcal{N}_e, s_k, r_k \in \mathcal{N}\}$, and a global attribute $\mathbf{u}^t$. We set the position and velocity of agent i at time t to be a node attribute, i.e., $\mathbf{v}_i^t = (x_i^t, \Delta x_i^t)$, and the relative position of the k-th pair of agents at time t to be an edge attribute, i.e., $\mathbf{e}_k^t = \Delta x_{s_k r_k}^t$. We initialize the global attribute with a zero vector since the model does not have a dynamic global context. However, we leave the global attribute for the model to have ability to keep aggregated information in the attribute.

Our model takes as input the world state g^t and outputs a predicted world state $\hat{g}^{t+1}$. Computation takes place within a network of interconnected blocks: an encoder block G_{en}, a decoder block G_{de}, and a recurrent block G_{re} (see Fig. 4a), introduced to handle temporal data effectively. The block G_{en} encodes the graph g^t into a *latent* graph g_{l0}^t. Then, the recurrent block G_{re} modifies it into g_{l1}^t and passes it to G_{de}, which returns an updated graph $\hat{g}^{t+1}$ that represents the predicted state of the world in Cartesian coordinates. Within each block, the following computations take place (we set $g = g^t$, $g' = g^{t+1}$, and assume $i \in \mathcal{N}, k \in \mathcal{N}_e$ for brevity):

$$\begin{aligned}
\mathbf{e}'_k &= \mathbf{e}_k + \phi_e\left(\mathbf{e}_k, \mathbf{v}_{s_k}, \mathbf{v}_{r_k}, \mathbf{u}, \mathbf{w}_k\right) & \bar{\mathbf{e}}'_i &= \rho_{e\to v}\left(E'_i\right) & E'_i &= \left\{\left(\mathbf{e}'_k, s_k, r_k\right), r_k = i\right\} \\
\mathbf{v}'_i &= \mathbf{v}_i + \phi_v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}, \mathbf{d}_i\right) & \bar{\mathbf{e}}' &= \rho_{e\to u}\left(E'\right) & V' &= \left\{\mathbf{v}'_i\right\} \\
\mathbf{u}' &= \mathbf{u} + \phi_u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right) & \bar{\mathbf{v}}' &= \rho_{v\to u}\left(V'\right) & E' &= \left\{\left(\mathbf{e}'_k, s_k, r_k\right)\right\},
\end{aligned} \tag{3}$$

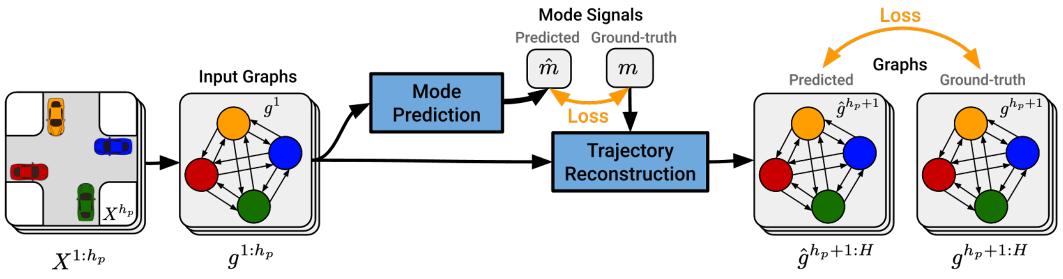
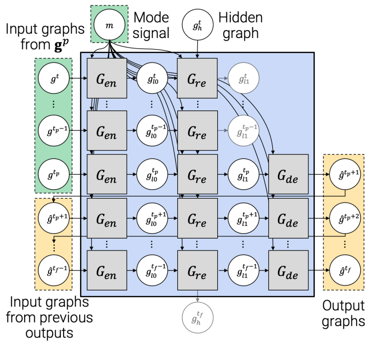
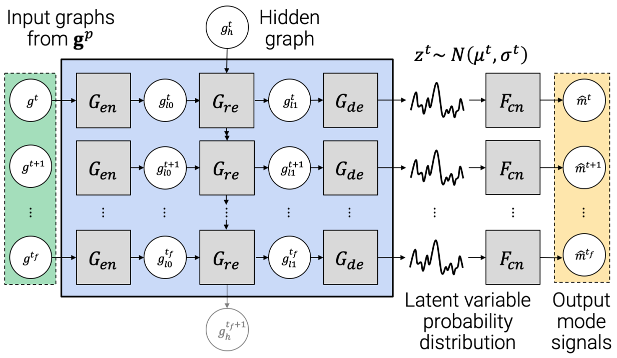


Figure 3: **Overview of model training.** We train a mode prediction and a trajectory reconstruction model, using the GraphNet [9] framework. Encoding the system state as a graph, we supervise predicted mode signals and reconstructed trajectories.


Input graphs from $\mathbf{g}^p$
Mode signal
Hidden graph
G_{en}
G_{re}
G_{de}
Input graphs from previous outputs
Output graphs

(a) Trajectory reconstruction model.


Input graphs from $\mathbf{g}^p$
Hidden graph
$z^t \sim N(\mu^t, \sigma^t)$
G_{en}
G_{re}
G_{de}
F_{cn}
Latent variable probability distribution
Output mode signals

(b) Mode prediction model.

Figure 4: **Internal topology of trajectory reconstruction (a) and mode prediction (b) models.** At time t, the trajectory reconstruction model takes as input the graph g^t, a hidden graph g_h^t, and a mode m and generates an output graph $\hat{g}^{t+1}$. This procedure is repeated up to timestep t_f, yielding a trajectory in the form of a sequence of graphs $\hat{\mathbf{g}}^f = \hat{g}^{t+1:t_f}$. The mode prediction model takes as input the graph g^t and a hidden graph g_h^t and produces a probability distribution over a latent variable $\mathbf{z}^t$. A sample is drawn from the distribution and transformed into a predicted mode $\hat{m}^t$. Variables in green shades indicate input from the ground-truth in training, variables in yellow shades are the outputs, and grey variables are discarded. At inference, graphs up to current timestep t, $g^{t_p:t}$, are fed to the mode-prediction model to estimate modes, $\{\hat{m}_{1:M}\}$. Given estimated modes and the graphs, the trajectory-reconstruction model predicts future trajectories.

where ϕ and ρ are respectively update and aggregation functions, $\mathbf{d}_i \in \{0,1\}^4$, $\mathbf{w}_k \in \{0,1\}^2$ are one-hot encoded vectors of the destination of agent i and of the sign of the winding number of edge k respectively. We model update functions ϕ_e, ϕ_v, ϕ_u using fully connected layers with ReLU and layer normalization and aggregation functions $\rho_{e\to v}$, $\rho_{e\to u}$, $\rho_{v\to u}$ as means over the updates of their corresponding sets. In the recurrent block, G_{re}, we use a single-layer GRU (Gated Recurrent Unit) [31] to efficiently infer the sequence of vehicle positions.

4.2.2 Sampling High-Likelihood Modes

The trajectory reconstruction model takes a mode m as input. The number of modes is exponential in the number of agents, and not all modes are equally likely. To address this issue, we sample high-likelihood modes by adapting our reconstruction architecture (see Fig. 4a) into a mode prediction model (see Fig. 4b). The model fits a Gaussian distribution over a latent variable $\mathbf{z}^t \sim \mathcal{N}(\mu^t, \sigma^t)$. This variable is transformed into a mode via fully connected layers, i.e., $\hat{m}^t = F_{cn}(\mathbf{z}^t)$. This process resembles a variational auto-encoder [32]; however, our model is supervised to generate conditional signals (modes) instead of reconstructing original input signals (configurations).

Scenario	Two Agents		Three Agents		Four Agents	
Metric	minADE (m)↓	minFDE (m)↓	minADE (m)↓	minFDE (m)↓	minADE (m)↓	minFDE (m)↓
MTP	**0.301 ± 0.346**	**0.611 ± 0.893**	**0.304 ± 0.459**	**0.717 ± 1.366**	**0.264 ± 0.410**	**0.588 ± 1.067**
MTP-fc	0.523 ± 0.440	1.063 ± 1.020	0.456 ± 0.612	1.024 ± 1.743	0.334 ± 0.342	0.704 ± 0.996
GRU	0.929 ± 0.566	1.578 ± 1.159	1.176 ± 0.727	2.119 ± 1.870	0.958 ± 0.551	1.760 ± 1.408
MFP	0.834 ± 0.969	0.825 ± 1.318	1.603 ± 0.981	1.504 ± 1.857	1.556 ± 0.707	1.444 ± 1.758

Table 1: **Prediction accuracy** (mean, standard deviation) measured as minADE and minFDE (average and final displacement error) for models trained on datasets with two, three, and four agents. MTP performs significantly better than other models (Wilcoxon signed-rank test, $p < 0.001$).

4.3 Following High-Quality Modes

Based on MTP, we build MTPnav, a cost-based planner (see Fig. 1). At planning time t, MTPnav invokes MTP, which returns a set of M high-likelihood trajectory predictions $\mathcal{X}$, corresponding to high-likelihood modes $\mathbf{m}$ of multiagent behavior. These predictions are then evaluated with respect to a trajectory cost $J : \mathcal{X}^n \to \mathbb{R}$:

$$J(\mathbf{X}) = w_{sm} J_{sm}(\mathbf{X}) + w_{ref} J_{ref}(\mathbf{X}) + w_{col} J_{col}(\mathbf{X}) + w_p J_p(\mathbf{X}), \tag{4}$$

where J_{sm} penalizes non-smooth trajectories, J_{ref} penalizes deviations from the ego-vehicle reference trajectory, J_{col} penalizes collisions, J_p penalizes low-likelihood trajectories, and w_{sm}, w_{ref}, w_{col}, w_p are corresponding weights (refer to Appendix, Sec. A for detailed formulation). The trajectory of minimum cost $\mathbf{X}_*$ is selected and passed to a tracking controller, which returns a control input u^* for the agent to execute.

5 Evaluation

We evaluate the performance of the proposed framework in two ways. First, we verify the ability of MTP to produce high-quality, mode-conditioned trajectory reconstruction. Next, we illustrate the value of MTPnav for tasks involving navigation at uncontrolled intersections through a simulated case study on a series of challenging intersection-crossing scenarios.

5.1 Datasets

Existing open datasets [14, 15] do not contain organized and sufficient amounts of interactions at uncontrolled intersections. For this reason, we generated a series of simulated datasets of challenging intersection crossings. Our data-generation pipeline consists of: (1) extracting realistic path references τ_i, $i \in \mathcal{N}$ following non-holonomic car kinematics using CARLA [10], and (2) executing these references under a variety of different behaviors and scenarios using a custom, computationally efficient, purely kinematic simulator. Our simulator controls vehicles in a centralized fashion using a priority system that provides acceleration/deceleration signals based on a time-to-collision heuristic and according to a set of parameters representing agents' behavioral models (desired speed, acceleration, deceleration, etc.). The vehicles track their paths using PID-based steering and speed controllers. This pipeline lets us combine the realism of CARLA with the ability to generate data efficiently and control scenarios and agent behaviors. This strategy also provides the ability to deploy our models directly to CARLA without re-training or fine-tuning to evaluate navigation performance (Sec. 5.4). Overall, we generate three diverse datasets of the form $\mathcal{D}_n = \{\mathbf{X}_1^n, \dots, \mathbf{X}_{|\mathcal{D}_n|}^n\}$, containing $|\mathcal{D}_2| = 116k$, $|\mathcal{D}_3| = 1,180M$, and $|\mathcal{D}_4| = 466k$ multiagent trajectories involving $n = 2, 3, 4$ agents, respectively, by methodically varying: (1) agents' destinations D (spanning the set of combinations), (b) their target speeds ($3 - 12$m/s), and (c) their acceleration/deceleration ($1 - 5$m/s^2).

5.2 Model Training

We train different trajectory-reconstruction and mode-prediction models on each of the datasets $\mathcal{D}_2$, $\mathcal{D}_3$, $\mathcal{D}_4$, following 9:1 train/test splits. We partition the datasets into a set of subtrajectories with a length of $H = h_p + h_f$ where $h_p = 15$, and $h_f = 25$. We transform each subtrajectory into a sequence of graphs $\mathbf{g} = \{g^1, \cdots, g^H\}$ and split it into $\mathbf{g}^p = \{g^1, \cdots, g^{h_p}\}$ and

$\mathbf{g}^f = \{g^{h_p+1}, \cdots, g^H\}$. We also compute a corresponding mode signal m for each sequence $\mathbf{g}$. Then we train the trajectory reconstruction model with student-forcing, providing the ground-truth mode signal m and a sequence of history graphs $\mathbf{g}^p$ to produce $\hat{\mathbf{g}}^f$. The mode prediction model is trained with the full ground-truth sequence $\mathbf{g}$ and generates a mode prediction $\hat{m}$. Fig. 3 depicts an overview of the training process. We use a mean squared error (MSE) loss for training the trajectory reconstruction model. For training the mode prediction model, we use a loss similar to the one used by β-VAE [33] (setting $\beta = 8$), featuring a cross-entropy component for reconstructing the mode signal given the latent variable and a KL-divergence component keeping the latent distribution close to a Gaussian. Note that at inference time, we only take the predicted mode signal $\hat{m}^{h_p}$ from the mode-prediction model and feed it to the trajectory-reconstruction model to collect $\hat{\mathbf{g}}^f$. See Sec. B.2 for mode details on the training process.

5.3 Trajectory Prediction Accuracy

We evaluate the accuracy of MTP in making multimodal trajectory predictions on a set of 1,000 randomly selected subtrajectories from the test sets. For each subtrajectory, we sample 100 latent variables $\mathbf{z}$ from the predicted distribution and predict trajectories for distinct modes. We perform an ablation study comparing the performance of MTP to two alternative models: a GRU and a modified MTP without a GRU (MTP-fc). Both use the same structure to model the latent variable distribution. We also compare MTP to MFP [1], which constitutes a state-of-the-art multimodal trajectory prediction framework. We employ the original MFP implementation (fixing the number of generated futures to $k = 3$) and modify the code to transform each trajectory to its local coordinate frame, as described by Tang and Salakhutdinov [1] (the state embedding in their implementation differs from the original paper). Note that other models (including MTP) do not use a local transformation.

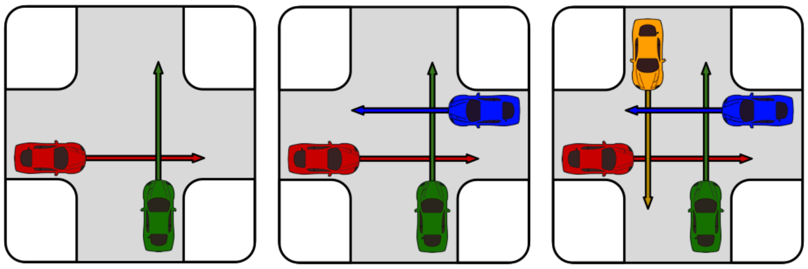
Table 1 shows the prediction accuracy of our framework, measured using the minADE and minFDE metrics (the minimum values of average/final displacement errors in multiple predictions). Our model outperforms the baselines across all datasets. It exhibits on average 71.63% lower minADE error than GRU and 33.82% less than MTP-fc. Importantly, our model outperforms MFP by 78.24% (on average) in terms of minADE. Further, note that MTP maintains comparable errors as the number of agents increases; in contrast, the performance of MFP drops. *Our insight is that the superior performance achieved by MTP can be attributed to the salient encoding of interaction provided by the formal definition of* modes. Our model can leverage physically meaningful signals as conditional signals instead of choosing one of a fixed number of latent behaviors. However, note that MFP was designed to apply more broadly to a variety of driving environments, unlike our approach, which was specifically designed for intersections.

5.4 Navigation Performance

We further evaluate the quality of the MTP predictions through a simulated case study on a decentralized navigation task at a four-way uncontrolled intersection of size 60×60 m on CARLA [10]. We consider three different intersection-crossing scenarios involving $n = 2, 3, 4$ agents, specifically selected to elicit challenging vehicle interactions, as shown in Fig. 5. During execution, the ego agent (agent $\#1$, the bottom agent in Fig. 5) is running our framework (MTPnav), while other agents are running the CARLA [10] Autopilot controller.[1] We consider two different conditions: *Easy* and *Hard*. *Easy* consists of *Cautious* and/or *Aggressive* agents with target speeds sampled uniformly from $[10, 25]$ and $[30, 45]$ (km/h), respectively. *Hard* consists of *Aggressive* and/or *Normal* agents with target speeds sampled from $[30, 45]$ and $[20, 40]$ (km/h), respectively, where *Cautious*, *Normal* and *Aggressive* are behavioral agents provided by CARLA. Uniformly sampling from these ranges, we construct 50 random experiments per scenario and condition.

We compare the performance of MTPnav to two baselines: the Autopilot (homogeneous case, serving as a reference for the difficulty of the task) and a purely reactive Model Predictive Controller (MPC) that treats other agents as obstacles. We employ using two metrics: (1) collision frequency $\mathcal{C}(\%)$ and (2) the time the ego-agent took to reach its destination $\mathcal{T}$ (s). If the ego-agent collides, we mark the trial as *in-collision* and assign it a time penalty of $30s$ corresponding to the time-to-destination for a *Cautious* agent with constant speed of $10km/h$. Trials that did not terminate by the $30s$ mark are also marked as in-collision.

[1]Autopilot is a widely available baseline that is part of the CARLA simulator [10].



(a) Two-agent scenario. (b) Three-agent scenario. (c) Four-agent scenario.

Figure 5: Navigation scenarios considered in our evaluation on CARLA.

Scenario		Two Agents		Three Agents		Four Agents	
Interaction		Easy	Hard	Easy	Hard	Easy	Hard
Autopilot	$\mathcal{T}(s)$	23.37 ± 6.01	26.00 ± 7.52	23.01 ± 5.78	28.69 ± 3.96	22.16 ± 7.56	28.41 ± 4.31
	$\mathcal{C}(\%)$	28.00 ± 6.35	78.00 ± 5.86	30.00 ± 6.48	86.00 ± 4.91	40.00 ± 6.93	88.00 ± 4.60
MPC	$\mathcal{T}(s)$	22.37 ± 6.26	26.98 ± 6.43	21.34 ± 6.26	27.78 ± 5.54	21.28 ± 7.98	28.25 ± 4.74
	$\mathcal{C}(\%)$	30.00 ± 6.48	82.0 ± 5.43	28.00 ± 6.35	86.00 ± 4.91	40.0 ± 6.93	88.0 ± 4.60
MTPnav	$\mathcal{T}(s)$	$\mathbf{19.13 \pm 3.78}$	$\mathbf{14.81 \pm 1.56}$	$\mathbf{20.85 \pm 3.71}$	$\mathbf{18.70 \pm 0.61}$	$\mathbf{19.34 \pm 8.03}$	$\mathbf{21.36 \pm 5.91}$
	$\mathcal{C}(\%)$	$\mathbf{2.00 \pm 1.98}$	$\mathbf{0}$	$\mathbf{2.00 \pm 1.98}$	$\mathbf{0}$	$\mathbf{28.00 \pm 6.35}$	$\mathbf{30.00 \pm 6.48}$

Table 2: **Navigation performance measured with respect to collision frequency ($\mathcal{C}$) and time to destination for ego-agent** ($\mathcal{T}$). Each entry contains a mean and a standard deviation over 50 trials. red, green and blue entries indicate scenarios in which MTPnav outperformed other methods at significance levels $p < 0.001$, $p < 0.01$, $p < 0.05$ respectively (Wilcoxon signed-rank test).

Table 2 shows the performance of MTPnav. We observe that MTPnav outperforms the baselines across all scenarios and interaction settings in terms of both time and collision frequency. In particular, its collision frequency is close to zero for scenarios with two and three agents, and significantly lower in scenarios involving four agents. We also see that it is significantly more time efficient, especially in the *Hard* scenarios. Overall, we attribute MTPnav's performance to its ability to exploit the domain structure: MTPnav agents are capable of rapidly adapting to the dynamic environment by foreseeing the multiagent interaction dynamics. We see a significant increase in collisions for scenarios involving four agents. This could be attributed to the steep increase in domain complexity (for reference, the number of possible modes is 162) but also possibly to the relatively small four-agent dataset that we employed.

6 Discussion

We introduced a prediction model (MTP) and a planner (MTPnav) for navigation at uncontrolled intersections, leveraging a mathematical insight about the topological structure of intersection crossings. MTP, trained on datasets acquired using a custom-built, lightweight simulator, outperformed a state-of-the-art, trajectory-prediction baseline [1] and enabled MTPnav to exhibit safe and time-efficient behavior across a series of challenging intersection-crossing tasks (without additional fine-tuning or retraining) that were conducted using the high-fidelity simulation engine CARLA. Our findings illustrate the value of reasoning about topological modes as a strategy to guide prediction towards a small representative set of outcomes. Ongoing work involves real-world experiments on a miniature racecar [34] to highlight the virtues of our approach for real-world applications.

Our work is not without limitations. Our data generation and evaluation pipelines are constrained by our behavior parametrizations and the mechanics of our simulators. Further, our evaluation experiments are limited by the computational load of running expensive CARLA experiments. Future work will focus on leveraging our empirical insights towards expanding our simulator to capture richer spaces of behaviors for data generation and testing. Finally, the MTPnav framework is proposed as an example, non-optimized incorporation of MTP on a cost-based planner; alternative use cases could apply reinforcement learning or model predictive control frameworks.

Acknowledgments

This work was (partially) funded by the National Science Foundation IIS (#2007011), National Science Foundation DMS (#1839371), the Office of Naval Research, US Army Research Laboratory CCDC, Amazon, Honda Research Institute USA, UW Reality Lab, Facebook, Google, Huawei, and a Samsung Scholarship. We would also like to thank NVIDIA for generously providing DGX via the NVIDIA Robotics Lab and the UW NVIDIA AI Lab (NVAIL).

References

[1] C. Tang and R. R. Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems*, pages 15398–15408, 2019.

[2] T. Baltic, R. Hensley, and J. Salazar. *The trends transforming mobility's future*. McKinsey & Company, March 2019.

[3] US Department of Transportation, Federal Highway Administration. Unsignalized intersections, 2018. URL https://safety.fhwa.dot.gov/intersection/conventional/unsignalized/.

[4] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2034–2039, 2018.

[5] C. Mavrogiannis, J. A. DeCastro, and S. S. Srinivasa. Implicit Multiagent Coordination at Unsignalized Intersections via Multimodal Inference Enabled by Topological Braids. *arXiv e-prints*, art. arXiv:2004.05205, 2020.

[6] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin. Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1475–1481, 2020.

[7] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2):393 – 405, 1990.

[8] M. A. Berger. Topological invariants in braid theory. *Letters in Mathematical Physics*, 55(3): 181–192, 2001.

[9] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv e-prints*, art. arXiv:1806.01261, 2018.

[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning*, pages 1–16, 2017.

[11] S. Ammoun and F. Nashashibi. Real time trajectory prediction for collision risk estimation between vehicles. In *IEEE International Conference on Intelligent Computer Communication and Processing*, pages 417–422, 2009.

[12] J. Firl, H. Stübing, S. A. Huss, and C. Stiller. Predictive maneuver evaluation for enhancement of car-to-x mobility data. In *IEEE Intelligent Vehicles Symposium*, pages 558–564, 2012.

[13] T. Gindele, S. Brechtel, and R. Dillmann. Learning driver behavior models from traffic observations for decision making and planning. *IEEE Intelligent Transportation Systems Magazine*, 7(1):69–79, 2015.

[14] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[15] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. One Thousand and One Hours: Self-driving Motion Prediction Dataset. *arXiv e-prints*, art. arXiv:2006.14480, 2020.

[16] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2255–2264, 2018.

[17] N. Deo and M. M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR Workshops*, pages 1468–1476, 2018.

[18] X. Li, X. Ying, and M. C. Chuah. Grip: Graph-based interaction-aware trajectory prediction. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3960–3966, 2019.

[19] R. Chandra, T. Guan, S. Panuganti, T. Mittal, U. Bhattacharya, A. Bera, and D. Manocha. Forecasting Trajectory and Behavior of Road-Agents Using Spectral Clustering in Graph-LSTMs. *arXiv e-prints*, art. arXiv:1912.01118, 2019.

[20] J. Li, F. Yang, M. Tomizuka, and C. Choi. EvolveGraph: Multi-Agent Trajectory Prediction with Dynamic Relational Reasoning. *arXiv e-prints*, art. arXiv:2003.13924, 2020.

[21] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[22] C. I. Mavrogiannis and R. A. Knepper. Multi-agent trajectory prediction and generation with topological invariants enforced by hamiltonian dynamics. In *Algorithmic Foundations of Robotics XIII*, pages 744–761, Cham, 2020. Springer International Publishing.

[23] J. Liang, L. Jiang, K. Murphy, T. Yu, and A. Hauptmann. The Garden of Forking Paths: Towards Multi-Future Trajectory Prediction. *arXiv e-prints*, art. arXiv:1912.06445, 2019.

[24] A. Monti, A. Bertugli, S. Calderara, and R. Cucchiara. DAG-Net: Double Attentive Graph Neural Network for Trajectory Forecasting. *arXiv e-prints*, art. arXiv:2005.12661, 2020.

[25] A. Pierson, W. Schwarting, S. Karaman, and D. Rus. Navigating congested environments with risk level sets. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5712–5719, 2018.

[26] N. Rhinehart, R. McAllister, and S. Levine. Deep Imitative Models for Flexible Inference, Planning, and Control. *arXiv e-prints*, art. arXiv:1810.06544, 2018.

[27] S. L. Cleac'h, M. Schwager, and Z. Manchester. ALGAMES: A Fast Solver for Constrained Dynamic Games. In *Robotics: Science and Systems*, 2020.

[28] E. Artin. Theory of braids. *Annals of Mathematics*, 48(1):pp. 101–126, 1947.

[29] C. I. Mavrogiannis and R. A. Knepper. Multi-agent path topology in support of socially competent navigation planning. *The International Journal of Robotics Research*, 38(2-3):338–356, 2019.

[30] C. I. Mavrogiannis, V. Blukis, and R. A. Knepper. Socially competent navigation planning by deep learning of multi-agent path topologies. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6817–6824, 2017.

[31] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014.

[32] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[33] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2017.

[34] S. S. Srinivasa, P. Lancaster, J. Michalove, M. Schmittle, C. Summers, M. Rockett, J. R. Smith, S. Choudhury, C. Mavrogiannis, and F. Sadeghi. MuSHR: A Low-Cost, Open-Source Robotic Racecar for Education and Research. *arXiv e-prints*, art. arXiv:1908.08031, 2019.

Appendix

A Cost functions definition

We now provide the formulation for individual cost functions used in Eq. 4:

$$
\begin{aligned}
&\text{Smoothness cost: } J_{sm}(\mathbf{X}) = \sum_{t=0}^{T-1} \gamma^t ||v_{t+1} - v_t||,\\
&\text{Cross-track error: } J_{ref}(\mathbf{X}) = dist(\tau^t, X^t) * sin(\psi_{\tau^t} - \psi_{X^t}),\\
&\text{Collision cost: } J_{col}(\mathbf{X}) = \sum_{t=0}^{T} \gamma^t c_t(X^t), \text{ where } c_t(X^t) = \begin{cases} 0 & \text{if } D_i(X^t) \geq d_{min} \\ 1 & \text{else} \end{cases},\\
&\text{Likelihood cost: } J_p(\mathbf{X}) = \frac{1}{p_{m_i}}
\end{aligned}
$$

where v_t is the velocity at timestep *t*. $dist(\tau^t, X^t)$ is the euclidean distance between agent position and reference waypoint at timestep *t*. ψ_{τ^t} is the heading angle on reference waypoint and ψ_{X^t} is the heading angle of the agent, at timestep *t*. p_{m_i} is the likelihood for mode m_i. D_i is a distance function and d_{min} is the distance threshold for collision. To reduce the computational burden of collision-checking, we use the circle-based collision checking method by reducing the car footprint to a set of three circles covering the car volume. Any object is thus in collision with the vehicle, if its distance from the center is less than the specified threshold.

B Implementation Details

In this section, we provide more details about the data generation pipeline, the training process and the navigation experiments.

B.1 Data generation

We generated three different datasets using our custom simulator by varying three simulation parameters: a) agents' configurations (combinations of starting locations and intended destinations); b) agents' target speed; c) agents' acceleration/deceleration.

- Two agents
  - Number of configurations: $N_{d2} = 3^3 = 27$ (taking all possible configurations)
  - Number of speeds: $N_{s2} = 7^2$ (sampled from $[2.8 - 12.5]$ m/s with step size 1.4 m/s)
  - Number of accelerations: $N_{a2} = 10^2$ (sampled uniformly from $[1 - 5]$) m/s
  - Number of total episodes: $N_{e2} = N_{d2} \times N_{s2} \times N_{a2} \approx 132K$
- Three agents
  - Number of configurations: $N_{d3} = N_{d2} \times 2 \times 3 = 162$
  - Number of speeds: $N_{s3} = 7^3$ (sampled from $[2.8 - 12.5]$ m/s, step size 2.8 m/s)
  - Number of accelerations: $N_{a3} = 5^3$ (sampled uniformly from $[1 - 5]$) m/s
  - Number of total episodes: $N_{e3} = N_{d3} \times N_{s3} \times N_{a3} \approx 1,29M$
- Four agents
  - Number of configurations: $N_{d4} = 3^4$
  - Number of speeds: $N_{s4} = 3^4$ (sampled from $[2.8 - 9.7]$ m/s, step size 2.8 m/s)
  - Number of accelerations: $N_{a4} = 3^4$ (sampled uniformly from $[1 - 5]$) m/s
  - Number of total episodes: $N_{e4} = N_{d4} \times N_{s4} \times N_{a4} \approx 531K$

The final datasets were extracted upon pruning out episodes including collisions (see main paper for dataset sizes).

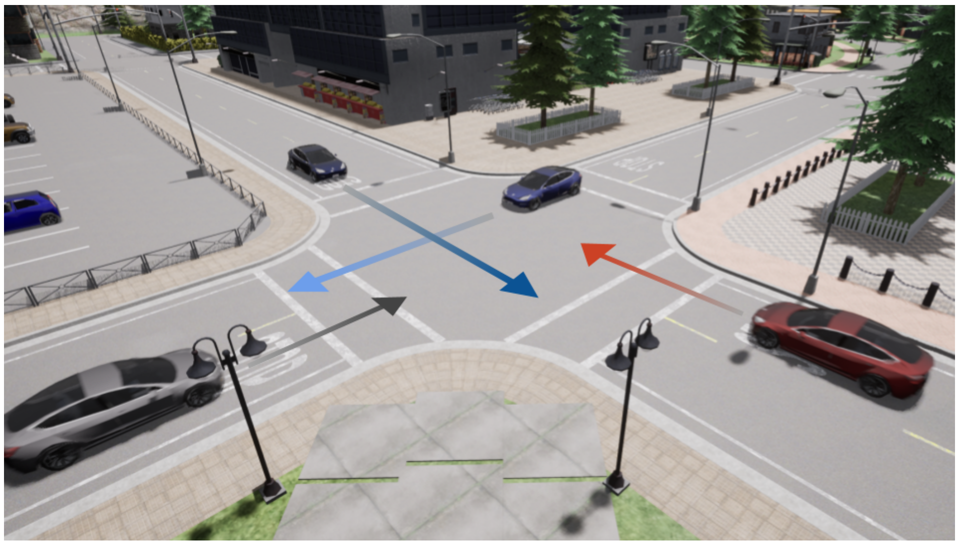


Figure 6: An example of the 4-agent scenario at an uncontrolled intersection in CARLA

B.2 Training

Dataset split We split the graph sequence $\mathbf{g} = g^{1:H}$ into a sequence of history graphs $\mathbf{g}^p = g^{1:h_p}$ and a sequence of target graphs $\mathbf{g}^f = g^{h_p+1:H}$.

Trajectory-reconstruction model training The trajectory-reconstruction model takes a graph g^t and a hidden graph g_h^t and predicts a graph at next timestep $\hat{g}^{t+1}$ along with an updated hidden graph g_h^{t+1} (see Fig. 4a). Training procedure has two stages: reading history sequence and predicting future sequence. In the first stage where $t < h_p$, the model takes g^t from the history graphs $\mathbf{g}^p$ and the output graph in this stage is discarded. Note that the hidden graph remains updated to keep the information propagated through time. In the second stage where $t \geq h_p$, the input graph is taken from the output graph at the previous timestep $t-1$, $\hat{g}^t$. The output graphs are collected to form the future trajectory. This process is repeated until the full predicted graph sequence $\hat{\mathbf{g}}^f = \hat{g}^{h_p+1:H}$ is produced. Loss is computed on $\hat{\mathbf{g}}^f$ and $\mathbf{g}^f$ by a mean squared error function. In the training, we provide ground-truth mode signals.

Mode-prediction model training The mode-prediction model shares a similar architecture (see Fig. 4b), with the difference that we provide the ground-truth sequence of graphs $\mathbf{g}$ to the model and collect mode signals $\hat{m}^t$ for $t \geq h_p$.

Hyperparameters We trained the models considered (both MTP and MFP) using PyTorch. For **MTP**, we set the following options – (Optimizer: Adam; Learning rate: 1×10^{-3}; Gradient clip: 1.0; Hidden unit size: 30. For **MFP**, we made use of the official implementation (https://github.com/apple/ml-multiple-futures-prediction) setting the following options – (Number of modes: 3; Subsampling: 2; Encoder size: 64; Decoder size: 128; Neighbor encoding size: 8; Neighbor attention embedding size: 20; Remove y mean: False; Use forcing: classmate forcing).

C Experiment Setup in CARLA

The navigation experiments are conducted at an uncontrolledf intersection within the `Town04` map of CARLA. For each experiment, agents are spawned on a specified side of the intersection and headed towards a destination waypoint corresponding to an intended direction (left, right, forward). At the beginning, all the agents are controlled using the Autopilot provided by CARLA. The ego agent controller switches to our method (MTPnav) or one of the baselines when the ego-agent reaches a distance of 25m from the center of the intersection, and switches back to Autopilot after crossing the intersection. The weights used in the cost function (see main paper) are set to the following values: $w_{sm} = 50$, $w_{ref} = 100$, $w_{col} = 10000$, $w_p = 1$.