PRECOG: PREdiction Conditioned On Goals in Visual Multi-Agent Settings

Nicholas Rhinehart[1] Rowan McAllister[2] Kris Kitani[1] Sergey Levine[2]
[1]Carnegie Mellon University [2]University of California, Berkeley
{nrhineha,kkitani}@cs.cmu.edu {rmcallister,svlevine}@berkeley.edu

Abstract

For autonomous vehicles (AVs) to behave appropriately on roads populated by human-driven vehicles, they must be able to reason about the uncertain intentions and decisions of other drivers from rich perceptual information. Towards these capabilities, we present a probabilistic forecasting model of future interactions of multiple agents. We perform both standard forecasting and conditional forecasting with respect to the AV's goals. Conditional forecasting reasons about how all agents will likely respond to specific decisions of a controlled agent. We train our model on real and simulated data to forecast vehicle trajectories given past positions and LIDAR. Our evaluation shows that our model is substantially more accurate in multi-agent driving scenarios compared to existing state-of-the-art. Beyond its general ability to perform conditional forecasting queries, we show that our model's predictions of all agents improve when conditioned on knowledge of the AV's intentions, further illustrating its capability to model agent interactions.

1. Introduction

Autonomous driving requires reasoning about the future behaviors of agents, *e.g.* at stop signs, roundabouts, crosswalks, or when parking. In multi-agent settings, each agent's behavior affects the behavior of others. Motivated by people's ability to reason in these settings, we present a method to forecast multi-agent interactions from perceptual data, such as images and LIDAR. Beyond forecasting the behavior of all agents, we want our model to *conditionally forecast* how other agents are likely to respond to different decisions each agent could make. When planning a robot to a goal, we want to forecast what other agents would likely do in response. This reasoning is essential for agents to make good decisions in multi-agent environments: they must reason how their future decisions could affect the multi-agent system around them. Examples of forecasting and conditioning forecasts on robot goals are shown in Fig. 1 and Fig. 2. Videos of the outputs of our approach are available at https://sites.google.com/view/precog.

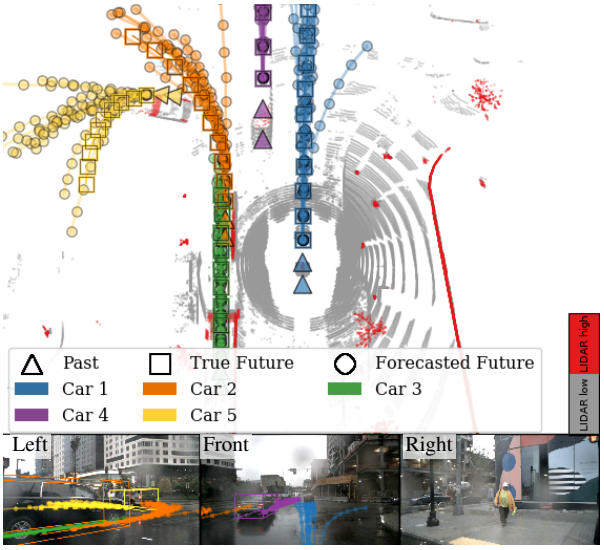


Figure 1: Forecasting on nuScenes [4]. The input to our model is a high-dimensional LIDAR observation, which informs a distribution over all agents' future trajectories.

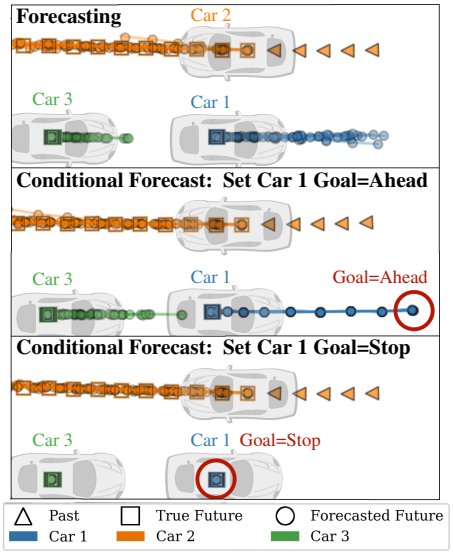


Figure 2: Conditioning the model on different Car 1 goals produces different predictions: here it forecasts Car 3 to move if Car 1 yields space, or stay stopped if Car 1 stays stopped.

To achieve accurate conditional forecasting, we propose a factorized flow-based generative model that forecasts the joint state of all agents. Our model reasons probabilistically about plausible future interactions between agents given rich observations of their environment. It uses latent variables to capture the uncertainty in other agents' intentions. Our key idea is the use of *factorized* latent variables to model decoupled agent intentions even though agent dynamics are coupled. Factorization across agents and time enable us to *query* the effects of changing an arbitrary agent's decision at an arbitrary time step.

Our contributions follow:

1. **State-of-the-art multi-agent forecasting:** We develop a multi-agent forecasting model called Estimating Social-forecast Probabilities (ESP) that uses *exact* likelihood inference (unlike VAEs or GANs) to outperform three state-of-the-art forecasting methods in real (nuScenes [4]) and simulated (CARLA [8]) datasets.
2. **Goal-conditioned multi-agent forecasting:** We present the first generative multi-agent forecasting method to condition on agent intent, called PREdicting Conditioned on Goals (PRECOG). After modelling agent interactions, conditioning on one agent's goal alters the predictions of other agents.
3. **Multi-agent imitative planning objective:** We derive a data-driven objective for motion planning in multi-agent environments. It balances the likelihood of reaching a goal with the probability that expert demonstrators would execute the same plan. We use this objective for offline planning to known goals, which improves forecasting performance.

2. Related Work

Multi-agent modeling and forecasting is a challenging problem for control applications such as autonomous driving. Safe control requires faithful models of reality to anticipate dangerous situations before they occur. Multi-agent forecasting and planning is particularly difficult, since all agents react to (and affect) each other concurrently. Modeling co-dependency between agents is especially critical in tightly-coupled scenarios such as intersections.

Game-theoretic planning: Traditionally, multi-agent planning and game theory approaches explicitly model multiple agents' policies or internal states, usually by generalizing Markov decision process (MDP) to multiple decisions makers [5, 33]. These frameworks facilitate reasoning about collaboration strategies, but suffer from "state space explosion" intractability except when interactions are known to be sparse [24] or hierarchically decomposable [11].

Multi-agent Forecasting: Data-driven approaches have been applied to forecast complex interactions between multiple pedestrians [1, 3, 10, 14, 21], vehicles [6, 19, 26], and athletes [9, 18, 20, 32, 34, 35]. These methods attempt to generalize from previously observed interactions to predict multi-agent behavior in new situations. Forecasting is related to Imitation Learning [25], which learns a model to mimic demonstrated behavior. Since forecasting approaches generally do not interact with the environment, they are essentially non-interactive Imitation Learning for forecasting. In contrast to some Imitation Learning methods, *e.g.* behavior cloning [28], behavior forecasting models are not executed in the environment of the observed agent – they are instead predictive models of the agent.

Forecasting methods that make Markovian assumptions typically treat the joint state over individual agents as a single "state" of the Markov process [34]. By forecasting the joint multi-agent state, such methods inherently model interactions at each time step. While these data-driven methods forecast multi-agent scenarios as observers, in situations where one or more of the agents is controlled, conditional forecasting is necessary to predict how the controls will affect the multi-agent system.

Forecasting for control and planning: Generative models for multi-agent forecasting and control have been proposed. In terms of multi-agent forecasting, our work is related to [31] which uses a conditional VAE [17] encoding of the joint states of multiple agents together with recurrent cells to predict future human actions. However, our work differs in three crucial ways. First, we model continual co-influence between agents, versus "robot-only" influence, where an agent's responses to the human are not modeled. Second, our method uses contextual visual information useful for generalization to many new scenes. Third, we model interactions between more than two vehicles *jointly*. While [15] assumes conditional independencies for computational reasons, we do not, as they impose minimal overhead.

We consider scenarios in which the model may control one of the agents (a "robot"). In terms of planned control, our method generalizes imitative models [30]. In [30], single-agent forecasting models are used for deterministic single-agent planning. Our work instead considers multi-agent forecasting, and therefore must plan over a distribution of possible paths: from our robot's perspective, the future actions of other human drivers are uncertain. By modeling co-influence, our robot's trajectory are conditional on the (uncertain) future human trajectories, and therefore future robots states are necessarily uncertain. Thus, our work proposes a nontrivial extension for imitative models: we consider future path planning uncertainty induced by the uncertain actions of other agents in a multi-agent setting. While [30] could implicitly model other agents through its visual conditioning, we show explicit modeling of other agents yields better forecasting results, in addition to giving us the tools to predict responses to agent's plans.

3. Deep Multi-Agent Forecasting

Now we describe our likelihood-based model for contextual multi-agent forecasting. We describe how we can condition our forecasts on decisions made by a subset of the agents. We describe how we can plan decisions according to an agent's intentions using our likelihood function as part of a planning objective. We use these planned decisions to perform intention-conditioned forecasting.

3.1. Notation

First, we define our notation and terminology for different types of predictive models applicable to autonomous driving. We treat our multi-agent system as a continuous-space, discrete-time, partially-observed Markov process, composed of A agents (vehicles) that interact over T time steps. We model all agent positions at time t as $\mathbf{S}_t \in \mathbb{R}^{A\times D}$, where $D=2$. $\mathbf{S}_t^a$ represents agent a's (x, y) coordinates on the ground plane. We assume there is one "robot agent" (*e.g.* the autonomous vehicle that our model can control) and A–1 "human agents" (*e.g.* human drivers that our model cannot control). For convenience, we define $\mathbf{S}_t^r \doteq \mathbf{S}_t^1 \in \mathbb{R}^D$ to index the robot state, and $\mathbf{S}_t^h \doteq \mathbf{S}_t^{2:A} \in \mathbb{R}^{(A-1)\times D}$ to index the human states. We distinguish variables in bold from functions (not bold). Random variables are capitalized. We define $t = 0$ to be the current time. Finally, a lack of time subscript denotes *all* future time steps, e.g. $\mathbf{S} \doteq \mathbf{S}_{1:T}^{1:A} \in \mathbb{R}^{T\times A\times D}$.

Each agent has access to environment perception $\phi \doteq \{\mathbf{s}_{-\tau:0}, \boldsymbol{\chi}\}$, where τ is the number of past multi-agent positions we condition on and $\boldsymbol{\chi}$ is a high-dimensional observation of the scene. $\boldsymbol{\chi}$ might represent LIDAR or camera images, and is the robot's observation of the world. In our setting, LIDAR is provided as $\boldsymbol{\chi} = \mathbb{R}^{200\times 200\times 2}$, with $\boldsymbol{\chi}_{ij}$ representing a 2-bin histogram of points above and at ground level in $0.5\mathrm{m}^2$ cells. Although our environment perception is centered on the robot, each agent is modeled to have access to $\boldsymbol{\chi}$.

3.2. Estimating Social-forecast Probability (ESP)

We propose a data-driven likelihood-based generative model of multi-agent interaction to probabilistically predict T-step dynamics of a multi-agent system: $\mathbf{S} \sim q(\mathbf{S}|\phi;\mathcal{D})$, where $\mathcal{D}$ is training data of observed multi-agent state trajectories. Our model is generative, and learns to map latent variables $\mathbf{Z}$ via an invertible function f to generate multi-agent state trajectories conditioned on ϕ. f's invertibility induces $q(\mathbf{S}|\phi)$, a *pushforward distribution* [23], also known as an *invertible generative model* [7, 12, 16, 29, 13]. Invertible generative models can efficiently and exactly compute probabilities of samples. Here, it means we can compute the probability of joint multi-agent trajectories, which is critical to our goal of *planning* with the model. Hence, we name the model Estimating Social-forecast Probabilities (ESP). $\mathbf{S}$ is sampled from q as follows:

$$\mathbf{S} = f(\mathbf{Z};\phi), \qquad \mathbf{S} \in \mathbb{R}^{T\times A\times D}, \tag{1}$$

$$\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \qquad \mathbf{Z} \in \mathbb{R}^{T\times A\times D}. \tag{2}$$

Our latent variables $\mathbf{Z} \doteq \mathbf{Z}_{1:T}^{1:A}$ factorize across agents and time, which allows us to *decide* agent a's reaction at time t by setting $\mathbf{Z}_t^a \leftarrow \mathbf{z}_t^a$, discussed later. Our model is related to the R2P2 single-agent generative model [29], which constructs a deep likelihood-based generative model for single-agent vehicle forecasting. For multi-step prediction, we generalize R2P2's recursive one-step single-agent prediction for the multi-agent setting, and assume a one-step time delay for agents to react to each other:

$$\mathbf{S}_t^a \;=\; \mu_\theta^a(\mathbf{S}_{1:t-1}, \phi) + \sigma_\theta^a(\mathbf{S}_{1:t-1}, \phi)\cdot \mathbf{Z}_t^a \;\in \mathbb{R}^D, \tag{3}$$

where $\mu_\theta^a(\cdot)$ and $\sigma_\theta^a(\cdot)$ are neural network functions (with trainable weights θ) outputting a one-step mean prediction $\boldsymbol{\mu}_t^a \in \mathbb{R}^D$ and standard-deviation matrix $\boldsymbol{\sigma}_t^a \in \mathbb{R}^{D\times D}$ of agent a, defining the system's transition function q:

$$q(\mathbf{S}_t|\mathbf{S}_{1:t-1}, \phi) = \prod_{a=1}^{A} \mathcal{N}(\mathbf{S}_t^a; \boldsymbol{\mu}_t^a, \boldsymbol{\Sigma}_t^a), \tag{4}$$

where $\boldsymbol{\Sigma}_t^a = \boldsymbol{\sigma}_t^a \boldsymbol{\sigma}_t^{a\top}$. Note that (3) predicts the a^{th} agent's state $\mathbf{S}_t^a$ given the previous *multi-agent* states $\mathbf{S}_{1:t-1}$. We can see that given $\mathbf{S}_{1:t-1}$, the one-step prediction in (3) is unimodal Gaussian. However, multi-step predictions are generally multimodal given the recursive nonlinear conditioning of neural network outputs $\boldsymbol{\mu}_t^a$ and $\boldsymbol{\sigma}_t^a$ on previous predictions. The final joint of this model can be written as

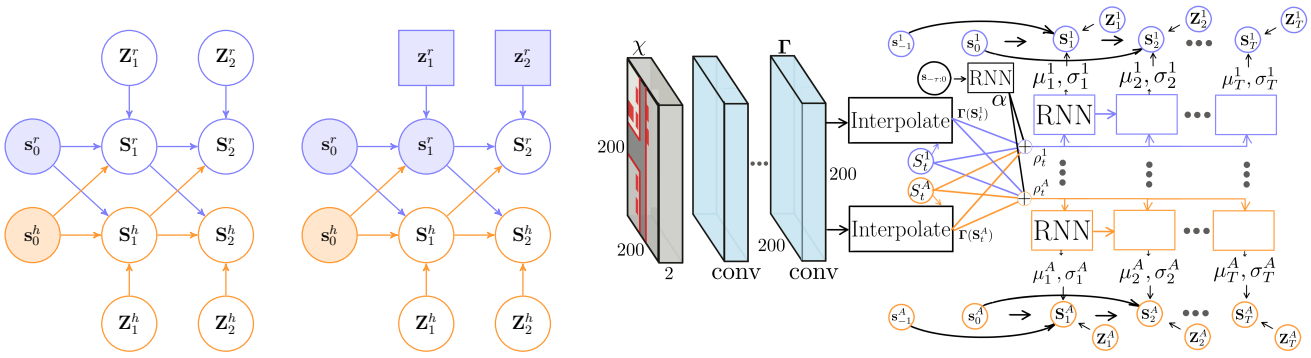
$$q(\mathbf{S}|\phi) = \prod_{t=1}^{T} q(\mathbf{S}_t|\mathbf{S}_{1:t-1}, \phi). \tag{5}$$

3.3. Model Implementation

To implement our model $q(\mathbf{S}|\phi)$, we design neural networks that output $\boldsymbol{\mu}_t^a$ and $\boldsymbol{\sigma}_t^a$. Similar to [29], we expand μ_θ^a to represent a "Verlet" step, which gives a constant-velocity mean prediction when network output $\mathbf{m}_t^a$ is 0:

$$\mathbf{S}_t^a = \underbrace{2\mathbf{S}_{t-1}^a - \mathbf{S}_{t-2}^a + \overbrace{m_\theta^a(\mathbf{S}_{1:t-1}, \phi)}^{\mathbf{m}_t^a}}_{\boldsymbol{\mu}_t^a} + \underbrace{\sigma_\theta^a(\mathbf{S}_{1:t-1}, \phi)}_{\boldsymbol{\sigma}_t^a}\cdot \mathbf{Z}_t^a. \tag{6}$$

A high-level diagram of our implementation shown in Fig. 3c. Recall the context $\phi \doteq \{\mathbf{s}_{-\tau:0}, \boldsymbol{\chi}\}$, containing the past positions of all agents, $\mathbf{s}_{-\tau:0}$, and a feature map $\boldsymbol{\chi}$, implemented as LIDAR is mounted on the first agent. We encode $\mathbf{s}_{-\tau:0}$ with a GRU. A CNN processes $\boldsymbol{\chi}$ to



(a) ESP forecasting (b) PRECOG planning (c) ESP model implementation

Figure 3: Our factorized latent variable model of forecasting and planning. In Fig. 3a our model uses latent variable $\mathbf{Z}^a_{t+1}$ to represent variation in agent a's *plausible* scene-conditioned reactions to all agents $\mathbf{S}_t$, causing uncertainty in every agents' future states $\mathbf{S}$ because they interact. Variation exists because of unknown driver goals and different driving styles observed in the training data. Beyond forecasting, our model admits *planning* robot decisions by *deciding* $\mathbf{Z}^r = \mathbf{z}^r$ (Fig. 3b). Shaded nodes represent observed or determined variables, and square nodes represent robot decisions (Barber's notation [2]). Note $\mathbf{Z}$ *factorizes* across agents, isolating the robot's reaction variable $\mathbf{z}^r$. Human goals and reactions remain uncertain ($\mathbf{Z}^h$ is unobserved) and are not controllable (the robot cannot decide $\mathbf{Z}^h$), and yet the robot's decisions $\mathbf{z}^r$ will still *influence* human drivers $\mathbf{S}^h_{2:T}$ (and vice-versa). Fig. 3c shows our implementation, with details in Appendix C.

$\mathbf{\Gamma}$ at the same spatial resolution as $\boldsymbol{\chi}$. Features for each agent's predicted position $\mathbf{S}^a_t$ are computed by interpolating into $\mathbf{\Gamma}$. "Social features" for agent a are computed: $\mathbf{S}^a_t - \mathbf{S}^b_t \;\forall\; b \in A \setminus \{a\}$. Then, the social features, past encoding, and CNN features are passed into a per-agent GRU, which produces $\mathbf{m}^a_t$ and $\boldsymbol{\sigma}^a_t$ in (6). We train our model with recordings of expert multi-agent interaction $\mathbf{S}^* \sim p(\mathbf{S}^*|\phi)$ by maximizing likelihood with respect to our model parameters θ. We used shared parameters to produce $\mathbf{\Gamma}$ and the past encoding, and independent parameters in the MLPs and GRUs after observing a performance boost by doing so. Further details are provided in the supplement.

3.4. PREdiction Conditioned On Goals (PRECOG)

A distinguishing feature of our generative model for multi-step, multi-agent prediction is its latent variables $\mathbf{Z} \doteq \mathbf{Z}^{1:A}_{1:T}$ that factorizes over agents and time. Factorization makes it possible to use the model for highly flexible conditional forecasts. Conditional forecasts enable the controlled (robot) agent to predict how other agents would likely respond to different robot decisions at different moments in time. Since robots are not merely passive observers, but one of potentially many agents, the ability to anticipate how they affect others is critical to their ability to plan useful, safe, and effective actions, critical to their utility within a planning and control framework [22].

Human drivers can appear to take highly stochastic actions in part because we cannot observe their intentions. In our model, the source of this uncertainty comes from the latent variables $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In practical scenarios, the robot knows its own intentions, can choose its own actions, and can plan a course of action to achieve a desired goal. Recall from (3) that *one-step* agent predictions are conditionally independent from each other give the previous multi-agent states. Therefore, certainty in the latent state $\mathbf{Z}^a_t$ corresponds to certainty of the a^{th} agent's *reaction* to the multi-agent system at time t. Different values of $\mathbf{Z}^a_t$ correspond to different ways of reacting to the same information. Deciding values of $\mathbf{Z}^a_t$ corresponds to controlling the agent a. We can therefore implement control of the robot via assigning values to its latent variables $\mathbf{Z}^r \leftarrow \mathbf{z}^r$. In contrast, human reactions $\mathbf{Z}^h_t$ cannot be decided by the robot, and so remain uncertain from the robot's perspective and can only be influenced by their conditioning on the robot's previous states in $\mathbf{S}_{1:t-1}$, as seen Fig. 3b. Therefore, to generate conditional-forecasts, we simply decide $\mathbf{z}^r$, sample $\mathbf{Z}^h$, concatentate $\mathbf{Z} = \mathbf{z}^r \oplus \mathbf{Z}^h$, and warp $\mathbf{S} = f(\mathbf{Z}, \phi)$.

This factorization of latent variables easily facilitates conditional forecasting. To forecast $\mathbf{S}$ with closed-loop control of the robot, we can fix $\mathbf{z}^r$ while sampling the human agents' reactions from their distribution $p(\mathbf{Z}^h) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, which are warped via (1).

3.5. Multi-Agent Planning

We discussed how forecasting can condition on some value of $\mathbf{z}^r$, but not yet how to find *desirable* values of $\mathbf{z}^r$, *e.g.* values that would safely direct the robot towards its goal location. We perform multi-agent planning by optimizing an objective $\mathcal{L}$ w.r.t. the control variables $\mathbf{z}^r$, which allows us to produce the "best" forecasts under $\mathcal{L}$.

While many valid objectives can be adopted, we take inspiration from imitative models (IM), which estimate the likeliest state trajectory an expert driver "would have taken" to reach a goal location, based on prior expert demonstra-

tions [30]. IM modeled single-agent environments where robot trajectories are planned without consideration of other agents. Multi-agent planning is different, because future robot states are uncertain (states $\mathbf{S}^r_{t>1}$ in Fig. 3b), even when conditioned on control variables $\mathbf{z}^r$, because of the uncertainty in surrounding human drivers $\mathbf{Z}^h$.

We generalize IM to multi-agent environments, and plan w.r.t. the uncertainty of human drivers close by. First, we chose a "goal likelihood" function that represents the likelihood that a robot reaches its goal $\mathcal{G}$ given state trajectory $\mathbf{S}$. For instance, the likelihood could be a waypoint $\mathbf{w} \in \mathbb{R}^D$ the robot should approach: $p(\mathcal{G}|\mathbf{S}, \phi) = \mathcal{N}(\mathbf{w}; \mathbf{S}^r_T, \epsilon\mathbf{I})$. Second, we combine the goal likelihood with a "prior probability" model of safe multi-agent state trajectories $q(\mathbf{S}|\phi)$, learned from expert demonstrations. Note that unlike many other generative multi-agent models, we can compute the probability of generating $\mathbf{S}$ from $q(\mathbf{S}|\phi)$ exactly, which is critical to our planning approach. This results in a "posterior" $p(\mathbf{S}|\mathcal{G}, \phi)$. Finally, we seek the value of $\mathbf{z}^r$ that maximizes the posterior probability. This corresponds to the robot planning a goal-seeking path that is within the learned distribution of demonstrated multi-agent behavior. Since this posterior is random due to unobserved $\mathbf{Z}^h$, we marginalize it out:

$$\log \mathbb{E}_{\mathbf{Z}^h}[p(\mathbf{S}|\mathcal{G}, \phi)] \;\geq\; \mathbb{E}_{\mathbf{Z}^h}[\log p(\mathbf{S}|\mathcal{G}, \phi)] \tag{7}$$

$$= \mathbb{E}_{\mathbf{Z}^h}[\log q(\mathbf{S}|\phi) \cdot p(\mathcal{G}|\mathbf{S}, \phi)] - \log p(\mathcal{G}|\phi) \tag{8}$$

$$\mathcal{L}(\mathbf{z}^r, \mathcal{G}) \doteq \mathbb{E}_{\mathbf{Z}^h}[\log q(\mathbf{S}|\phi) \cdot p(\mathcal{G}|\mathbf{S}, \phi)] \tag{9}$$

$$= \mathbb{E}_{\mathbf{Z}^h}[\log \underbrace{q(f(\mathbf{Z})|\phi)}_{\text{multi-agent prior}} + \log \underbrace{p(\mathcal{G}|f(\mathbf{Z}), \phi)}_{\text{goal likelihood}}], \tag{10}$$

where (7) follows by Jensen's inequality, which we use to avoid the numerical issue of a single sampled $\mathbf{Z}^h$ dominating the batch. (8) follows from Bayes' rule and uses our learned model q as the prior. In (9), we drop $p(\mathcal{G}|\phi)$ because it is constant w.r.t. $\mathbf{z}^r$. Recall $\mathbf{Z} = \mathbf{z}^r \oplus \mathbf{Z}^h$ is the concatenation of robot and human control variables. The robot can plan using our ESP model by optimizing (10):

$$\mathbf{z}^{r*} = \underset{\mathbf{z}^r}{\operatorname{argmax}}\, \mathcal{L}(\mathbf{z}^r, \mathcal{G}). \tag{11}$$

A "selfish" robot might instead seek to maximize the posterior probability of just its own trajectories. However, such an objective may place human agents in unusual, precarious driving situations, outside the prior distribution of "usual driving interaction" previously demonstrated. By optimizing (10), the robot avoids actions that would put either it or the other agents in unexpected situations.

4. Experiments

We first compare our forecasting model against existing state-of-the-art multi-agent forecasting methods, including SocialGAN [14], DESIRE [19]. We also include a baseline model: R2P2-MA (adapted from R2P2 [29] to instead handle multiple agent inputs), which does not model how agents will react to each others' future decisions. Second, we investigate the novel problem of *conditional* forecasting. To quantify forecasting performance, we study scenarios where we have samples of the robot's true intention and the human reactions to it. Knowledge of these intentions should enable our model to better predict what the robot and each agent could do. Third, we ablate the high-dimensional contextual input $\boldsymbol{\chi}$ from our model to determine its relevance to forecasting. Finally, we evaluate our model's test-time sensitivity to the robot's localization noise, and observation noise of other agents' states, and how much this sensitivity is mitigated with train-time noise injection.

4.1. Datasets

CARLA dataset: We generated a realistic dataset for multi-agent trajectory forecasting and planning with the CARLA simulator [8]. We ran the autopilot in `Town01` for over 900 episodes of 100 seconds each in the presence of 100 other vehicles, and recorded the trajectory of every vehicle and the autopilot's LIDAR observation. We randomized episodes to either train, validation, or test sets. We created sets of 60,701 train, 7586 validation, and 7567 test scenes, each with 2 seconds of past and 4 seconds of future position information at 5Hz. See Appendix E for details and https://sites.google.com/view/precog for datasets.

nuScenes dataset: We used the recently-released full nuScenes dataset [4], a real-world dataset for multi-agent trajectory forecasting, in which 850 episodes of 20 seconds of driving were recorded and labelled at 2Hz with the positions of all agents, and synced with many sensors, including LIDAR. We processed each of the examples to train, val, and test splits. Each example has 2 seconds of past and 4 seconds of future position information interpolated to 5Hz and accompanied by a LIDAR map composited from 10 previous scans at 10Hz. We also experimented with concatenating χ, which normally contains just featurized LIDAR, with a binary mask of *road* presence that nuScenes provides, indicated as "+Road" in our evaluation.

Didactic Benchmark: We also constructed a tightly-controlled scenario to illustrate a fundamental difference between the R2P2-MA and ESP models. The scene represents an intersection where a robot driver and a human driver cooperate to avoid crashing.

4.2. Metrics

Log-likelihood: As our models can perform exact likelihood inference (unlike GANs or VAEs), we can precisely evaluate how likely held-out samples are under each model. Test log-likelihood is given by the forward cross-entropy

Table 1: CARLA and nuScenes multi-agent forecasting evaluation. All CARLA-trained models use `Town01 Train` only, and are tested on `Town02 Test`. No training data is collected from `Town02`, and thus `Town02 Test` evaluates generalizability to new towns. Mean scores (and their standard errors) of sample quality $\hat{m}$ (13), and log likelihood $\hat{e}$ (12), are shown. The "–" symbol indicates if an approach cannot compute likelihoods. The R2P2-MA model generalizes the single-agent forecasting approach of [29]. Variants of our ESP method (highlighted gray) mostly outperform prior work in the multi-agent CARLA and nuScenes settings. For additional `Town01 Test` and single agent evaluations see Appendix F.

Approach	Test $\hat{m}_{K=12}$	Test $\hat{e}$	Test $\hat{m}_{K=12}$	Test $\hat{e}$	Test $\hat{m}_{K=12}$	Test $\hat{e}$	Test $\hat{m}_{K=12}$	Test $\hat{e}$
CARLA Town02 Test	2 agents		3 agents		4 agents		5 agents	
DESIRE [19]	1.943 ± 0.033	–	1.587 ± 0.020	–	2.234 ± 0.023	–	2.422 ± 0.017	–
SocialGAN [14]	0.977 ± 0.016	–	0.812 ± 0.013	–	1.098 ± 0.014	–	1.141 ± 0.015	–
R2P2-MA [29]	0.540 ± 0.009	0.625 ± 0.002	0.387 ± 0.008	0.645 ± 0.002	0.690 ± 0.009	0.621 ± 0.002	0.770 ± 0.008	$\mathbf{0.618 \pm 0.002}$
Ours: ESP, no LIDAR	0.724 ± 0.013	0.688 ± 0.003	0.719 ± 0.011	0.640 ± 0.002	0.919 ± 0.011	0.650 ± 0.002	1.102 ± 0.011	0.652 ± 0.002
Ours: ESP	$\mathbf{0.311 \pm 0.008}$	$\mathbf{0.615 \pm 0.002}$	$\mathbf{0.385 \pm 0.007}$	$\mathbf{0.585 \pm 0.002}$	$\mathbf{0.509 \pm 0.007}$	$\mathbf{0.599 \pm 0.002}$	$\mathbf{0.675 \pm 0.007}$	0.630 ± 0.001
nuScenes Test	2 agents		3 agents		4 agents		5 agents	
DESIRE [19]	3.473 ± 0.102	–	4.421 ± 0.130	–	5.957 ± 0.162	–	6.575 ± 0.198	–
SocialGAN [14]	2.119 ± 0.087	–	3.033 ± 0.110	–	3.484 ± 0.129	–	3.871 ± 0.148	–
R2P2-MA [29]	1.336 ± 0.062	0.951 ± 0.007	2.055 ± 0.093	0.989 ± 0.008	2.695 ± 0.100	1.020 ± 0.011	3.311 ± 0.166	$\mathbf{1.050 \pm 0.012}$
Ours: ESP, no LIDAR	1.496 ± 0.069	$\mathbf{0.920 \pm 0.008}$	2.240 ± 0.084	$\mathbf{0.955 \pm 0.008}$	3.201 ± 0.113	1.033 ± 0.012	3.442 ± 0.139	1.107 ± 0.018
Ours: ESP	1.325 ± 0.065	0.933 ± 0.008	1.705 ± 0.089	1.018 ± 0.011	2.547 ± 0.095	1.053 ± 0.015	3.266 ± 0.155	1.082 ± 0.013
Ours: ESP+Road	$\mathbf{1.081 \pm 0.053}$	0.929 ± 0.008	$\mathbf{1.505 \pm 0.070}$	1.016 ± 0.011	$\mathbf{2.360 \pm 0.093}$	$\mathbf{1.013 \pm 0.012}$	$\mathbf{2.892 \pm 0.162}$	1.114 ± 0.024

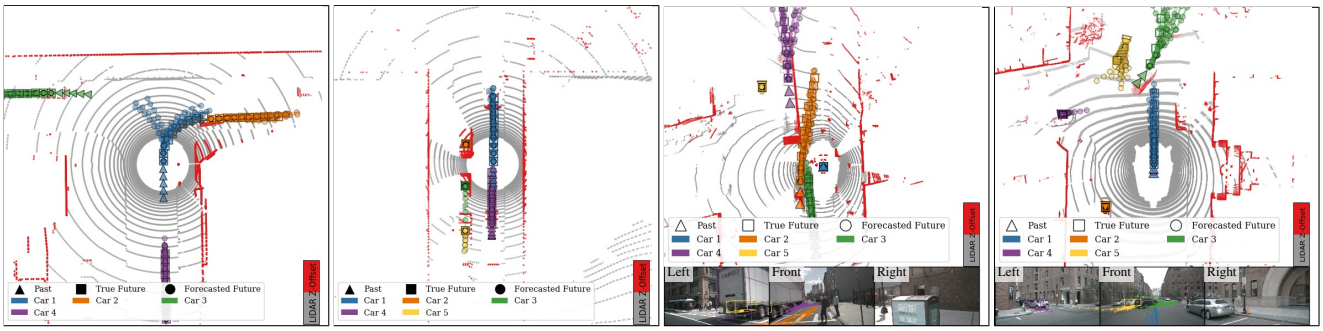


Figure 4: Examples of multi-agent forecasting with our learned ESP model. In each scene, 12 joint samples are shown, and LIDAR colors are discretized to near-ground and above-ground. *Left:* (CARLA) the model predicts Car 1 could either turn left or right, while the other agents' future maintain multimodality in their speeds. *Center-left:* The model predicts Car 2 will likely wait (it is blocked by Cars 3 and 5), and that Cars 3 and 5 sometimes move forward together, and sometimes stay stationary. *Center-right:* Car 2 is predicted to overtake Car 1, which itself is forecasted to continue to wait for pedestrians and Car 2. *Right:* Car 4 is predicted to wait for the other cars to clear the intersection, and Car 5 is predicted to either start turning or continue straight.

$H(p, q) = -\mathbb{E}_{\mathbf{S}^* \sim p(\mathbf{S}^*|\phi)} \log q(\mathbf{S}^*|\phi)$, which is unbounded for general p and q. However, by perturbing samples from $p(\mathbf{S}^*|\phi)$ with noise drawn from a known distribution η (*e.g.* a Gaussian) to produce a perturbed distribution p', we can enforce a lower bound [29]. The lower bound is given by $H(p', q) \geq H(p') \geq H(\eta)$. We use $\eta = \mathcal{N}(\mathbf{0}, 0.01 \cdot \mathbf{I})$, whose $H(\eta)$ is known analytically. For our final likelihood statistic we use:

$$\hat{e} \doteq \big[H(p', q) - H(\eta)\big]/_{(TAD)} \geq 0, \qquad (12)$$

which has $^{\text{nats}}/_{\text{dim.}}$ units. We call $\hat{e}$ "extra nats" because it represents the (normalized) extra nats above the lower bound of 0. Normalization enables comparison across models of different dimensionalities.

Sample quality: For sample metrics, we must take care not to penalize the distribution when it generates plausible samples different than the expert trajectory. We extend the "minMSD" metric [19, 26, 29] to measure quality of *joint trajectory samples*. The "minMSD" metric samples a model and computes the error of the best sample in terms of MSD. In contrast to the commonly-used average displacement error (ADE) and final displacement error (FDE) metrics that computes the mean Euclidean error from a batch of samples to a *single* ground-truth sample [1, 6, 10, 14, 27], minMSD has the desirable property of not penalizing plausible samples that correspond to decisions the agents could have made, but did not. *This prevents erroneously penalizing models that make diverse behavior predictions*. We hope other methods that make predictions on multimodal data will also measure the quality of joint samples with min-

MSD, given by:

$$\hat{m}_K \doteq \mathbb{E}_{\mathbf{S}^*} \min_{k \in \{1..K\}} ||\mathbf{S}^* - \mathbf{S}^{(k)}||^2/(TA), \quad \mathbf{S}^{(k)} \overset{\text{iid}}{\sim} q(\mathbf{S}|\phi), \tag{13}$$

which has square meter units, and $\mathbf{S}^* \sim p(\mathbf{S}^*|\phi)$. We denote the per-agent error of the best *joint* trajectory with

$$\hat{m}^a_K \doteq \mathbb{E}_{\mathbf{S}^* \sim p(\mathbf{S}^*|\phi)} ||\mathbf{S}^{*a} - \mathbf{S}^{a,(k^\dagger)}||^2/T, \quad k^\dagger \doteq \operatorname*{argmin}_{k \in \{1..K\}} ||\mathbf{S}^* - \mathbf{S}^{(k)}||^2. \tag{14}$$

4.3. Baselines

DESIRE [19] proposed a conditional VAE model that observes past trajectories and visual context. We followed the implementation as described. Whereas DESIRE is trained with a single-agent evidence lower bound (ELBO), our model jointly models multiple agents with an exact likelihood. As DESIRE does not compute multi-agent likelihoods, we cannot compute its $\hat{e}$.

SocialGAN [14] proposed a conditional GAN multi-agent forecasting model that observes the past trajectories of all modeled agents, but not $\boldsymbol{\chi}$. We used the authors' public implementation. In contrast to SocialGAN, we model joint trajectories and can compute likelihoods (and therefore $\hat{e}$).

R2P2 [29] proposed a likelihood-based conditional generative forecasting model for single-agents. We extend R2P2 to the multi-agent setting and use it as our R2P2-MA model; R2P2 does not jointly model agents. We otherwise followed the implementation as described. We trained it and our model with the forward-cross entropy loss. We can compute R2P2's likelihood, and therefore $\hat{e}$, by assuming independence across agents: $q(\mathbf{S}|\phi) = \prod_{a=1}^{A} q^a(\mathbf{S}^a|\phi)$.

4.4. Multi-Agent Forecasting Experiments

We build 4 datasets from CARLA and nuScenes data, corresponding to modeling different numbers of agents (2..5). Agents are sorted by their distances to the autopilot, at $t = 0$. When 1 agent is included, only the autopilot is modeled. When A agents are included, the autopilot and the $A-1$ closest vehicles are modeled.

For each method, we report its best test-set score at the best val-set score. In R2P2 and our method, the val-set score is $\hat{e}$. In DESIRE and SocialGAN, the val-set score is $\hat{m}$, as they cannot compute $\hat{e}$. Tab. 1 shows the multi-agent forecasting results. Across all 10 settings, our model achieves the best $\hat{m}$ scores, and achieves the best $\hat{e}$ score in $^8/_{10}$ settings. We also ablated our model's access to $\boldsymbol{\chi}$ ("ESP, no LIDAR"), which puts it on equal footing with SocialGAN, in terms of model inputs. Visual context provides a uniform improvement in every case.

Qualitative examples of our forecasts are shown in Fig. 4. We observe three important types of multimodality: 1) multimodality in speed along a common specific direction, 2) the model properly predicts diverse plausible paths at intersections, and 3) when the agents are stopped, the model predicts sometimes the agents will stay still, and sometimes they will accelerate forward. The model also captures qualitative social behaviors, such as predicting that one car will wait for another before accelerating. See Appendix G for additional visualizations.

4.5. PRECOG Experiments

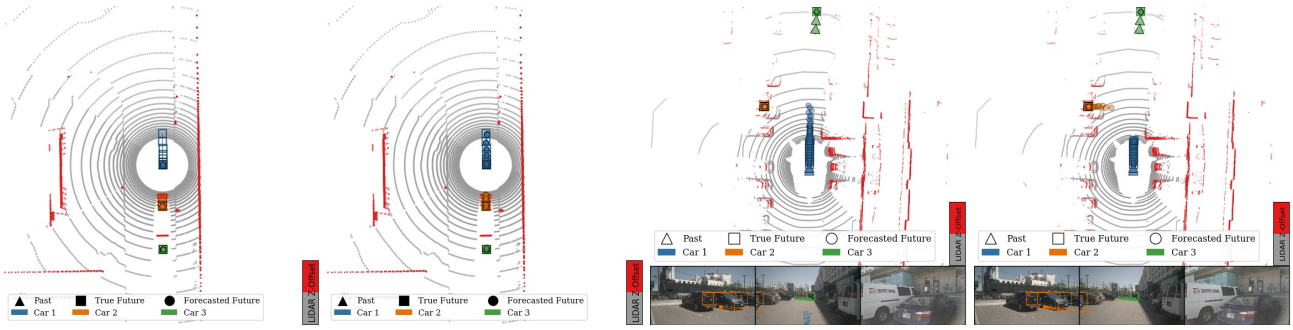
Now we perform our second set of evaluations. We investigate if our planning approach enables us to sample more plausible joint futures of all agents. Unlike the previous unconditional forecasting scenario, when the robot is using the ESP model for planning, it knows its own goal. We can simulate planning offline by assuming the goal was the state that the robot actually reached at $t = T$, and then planning a path from the current time step to this goal position. We can then evaluate the quality of the agent's path and the stochastic paths of other agents under this plan. While this does not test our model in a full control scenario, it does allow us to evaluate whether conditioning on the goal provides more accurate and higher-confidence predictions. We use our model's multi-agent prior (5) in the stochastic latent multi-agent planning objective (10), and define the goal-likelihood $p(\mathcal{G}|\mathbf{S}, \phi) = \mathcal{N}(\mathbf{S}^r_T; \mathbf{S}^{*r}_T, 0.1 \cdot \mathbf{I})$, *i.e.* a normal distribution at the controlled agent's last true future position, $\mathbf{S}^{*r}_T$. As discussed, this knowledge might be available in control scenarios where we are confident we can achieve this positional goal. Other goal-likelihoods could be applied to relax this assumption, but this setup allows us to easily measure the quality of the resulting joint samples. We use gradient-descent on (10) to approximate $\mathbf{z}^{r*}$ (see supplement for details). The resulting latent plan yields highly likely joint trajectories under the uncertainty of other agents and approximately maximizes the goal-likelihood. Note that since we planned in latent space, the resulting robot trajectory is not fully determined – it can evolve differently depending on the stochasticity of the other agents. We next illustrate a scenario where joint modeling is critical to accurate forecasting and planning. Then, we conduct planning experiments on the CARLA and nuScenes datasets.

4.5.1 Didactic Example

In the didactic example, a robot (blue) and a human (orange) both navigate in an intersection, the human has a latent intention: with 0.5 probability they will turn left, and otherwise they will drive straight. The human always travels straight for 4 time steps, and then reveals its latent intention by either going straight or left. The robot attempts

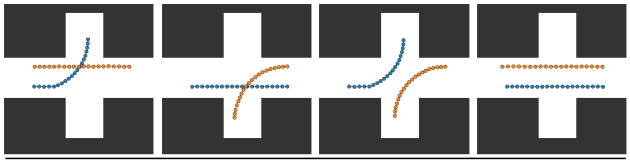
Table 2: Forecasting evaluation of our model on CARLA Town01 Test data. Planning the robot to a goal position (PRECOG) enables better predictions for all agents. Means and their standard errors are reported.

Data	Approach	Test $\hat{m}_{K=12}$	Test $\hat{m}^{a=1}_{K=12}$	Test $\hat{m}^{a=2}_{K=12}$	Test $\hat{m}^{a=3}_{K=12}$	Test $\hat{m}^{a=4}_{K=12}$	Test $\hat{m}^{a=5}_{K=12}$
CARLA 2	ESP	0.337 ± 0.013	0.196 ± 0.009	0.478 ± 0.024	–	–	–
	PRECOG	$\mathbf{0.241 \pm 0.012}$	$\mathbf{0.055} \pm 0.003$	$\mathbf{0.426} \pm 0.024$	–	–	–
CARLA 3	ESP	0.426 ± 0.013	0.204 ± 0.009	0.556 ± 0.027	0.519 ± 0.021	–	–
	PRECOG	$\mathbf{0.355 \pm 0.012}$	$\mathbf{0.052 \pm 0.003}$	$\mathbf{0.519 \pm 0.025}$	$\mathbf{0.493 \pm 0.020}$	–	–
CARLA 4	ESP	0.537 ± 0.011	0.236 ± 0.009	0.615 ± 0.021	0.656 ± 0.023	0.643 ± 0.023	–
	PRECOG	$\mathbf{0.478 \pm 0.011}$	$\mathbf{0.054 \pm 0.003}$	$\mathbf{0.583 \pm 0.021}$	$\mathbf{0.637 \pm 0.022}$	$\mathbf{0.638 \pm 0.023}$	–
CARLA 5	ESP	0.718 ± 0.012	0.340 ± 0.011	0.759 ± 0.024	0.809 ± 0.025	0.851 ± 0.023	0.828 ± 0.024
	PRECOG	$\mathbf{0.640 \pm 0.011}$	$\mathbf{0.066 \pm 0.003}$	$\mathbf{0.741 \pm 0.024}$	$\mathbf{0.790 \pm 0.024}$	$\mathbf{0.804 \pm 0.022}$	$\mathbf{0.801 \pm 0.024}$
nuScenes 2	ESP	1.094 ± 0.053	0.955 ± 0.057	1.233 ± 0.078	–	–	–
	PRECOG	$\mathbf{0.514 \pm 0.037}$	$\mathbf{0.158 \pm 0.016}$	$\mathbf{0.871 \pm 0.070}$	–	–	–
nuScenes 3	ESP	1.511 ± 0.077	1.128 ± 0.061	1.543 ± 0.118	1.862 ± 0.147	–	–
	PRECOG	$\mathbf{1.016 \pm 0.062}$	$\mathbf{0.121 \pm 0.005}$	$\mathbf{1.320 \pm 0.105}$	$\mathbf{1.606 \pm 0.122}$	–	–
nuScenes 4	ESP	2.200 ± 0.090	1.604 ± 0.099	1.940 ± 0.123	2.405 ± 0.149	2.851 ± 0.213	–
	PRECOG	$\mathbf{1.755 \pm 0.083}$	$\mathbf{0.133 \pm 0.006}$	$\mathbf{1.804 \pm 0.126}$	$\mathbf{2.319 \pm 0.141}$	$\mathbf{2.764 \pm 0.231}$	–
nuScenes 5	ESP	2.921 ± 0.175	1.861 ± 0.109	2.369 ± 0.188	2.812 ± 0.188	3.201 ± 0.254	4.363 ± 0.652
	PRECOG	$\mathbf{2.508 \pm 0.152}$	$\mathbf{0.149 \pm 0.021}$	$\mathbf{2.324 \pm 0.187}$	$\mathbf{2.654 \pm 0.190}$	$\mathbf{3.157 \pm 0.273}$	$\mathbf{4.254 \pm 0.586}$



(a) CARLA, ESP (b) CARLA, PRECOG (c) nuScenes, ESP (d) nuScenes, PRECOG

Figure 5: Examples of *planned* multi-agent forecasting (PRECOG) with our learned model in CARLA and nuScenes. By using our planning approach and conditioning the robot on its true final position, our predictions of the other agents change, our predictions for the robot become more accurate, and sometimes our predictions of the other agent become more accurate.



Model	Test $\hat{m}_{K=12}$	Test $\hat{e}$	Forecasting crashes	Planning crashes
R2P2-MA	0.331	0.085	50.8%	49.5%
ESP	**0.000**	**0.031**	**1.17%**	**0.00%**

Figure 6: Evaluation of our models on our "Social Cross" environment. *Left plots:* The R2P2-MA model cannot model agent interaction, and generates joint behaviors not present in the data. *Right plots:* The ESP model allows agents to influence each other, and does not generate undesirable joint behaviors. *Bottom*: Model performances.

to drive straight, but will acquiesce to the human if the human turns in front of the robot. We trained our models and evaluate them in Fig. 6. Each trajectory has length $T = 20$. While both models closely match the training distribution in terms of likelihood, their sample qualities are significantly different. The R2P2-MA model generates samples that crash 50% of the time, because it does not condition future positions for the robot on future positions of the human, and vice-versa. In the ESP model, the robot is able to react to the human's decision during the generation process by choosing to turn when the human turns.

4.5.2 CARLA and nuScenes PRECOG

We use the trained ESP models to run PRECOG on the test-sets in CARLA and nuScenes. Here, we use both $\hat{m}_K$ and $\hat{m}^a_K$ to quantify joint sample quality in terms of all agents

and each agent individually. In Tab. 2 and Fig. 5, we report results of our planning experiments. We observe that our planning approach significantly improves the quality of the joint trajectories. As expected, the forecasting performance improves the most for the planned agent ($\hat{m}_K^1$). Notably, the forecasting performance of the other agents improves across all datasets and all agents. We see the non-planned-agent improvements are usually greatest for Car 2 ($\hat{m}_K^2$). This result conforms to our intuitions: Car 2 is the *closest* agent to the planned agent, and thus, it the agent that Car 1 influences the most. Qualitative examples of this planning are shown in Fig. 5. We observe trends similar to the CARLA planning experiments – the forecasting performance improves the most for the planned agent, with the forecasting performance of the unplanned agent improving in response to the latent plans. See Appendix G for additional visualizations.

4.6. Robustness to Agent Localization Errors

In real-world data, there may be error in the localization of the other agents ($\mathbf{s}_{-\tau:0}$). We can simulate this error in our test-set by perturbing $\mathbf{s}^a_{-\tau:0}$ with a random vector $v^a \sim \mathcal{N}(\mathbf{0}, \epsilon I_{D\times D})$. We also train a model by injecting noise generated similarly. In Fig. 7 we compare nuScenes A2 ESP models trained without ($M_{\epsilon=0.0}$) and with ($M_{\epsilon=0.1}$) noise injection. We observe that $M_{\epsilon=0.0}$ is much more sensitive to test-time noise than $M_{\epsilon=0.1}$ at all perturbation scales, which shows noise injection is an effective strategy to mitigate the effects of localization error. We also note $M_{\epsilon=0.1}$ improves performance even when the test-data is not perturbed.

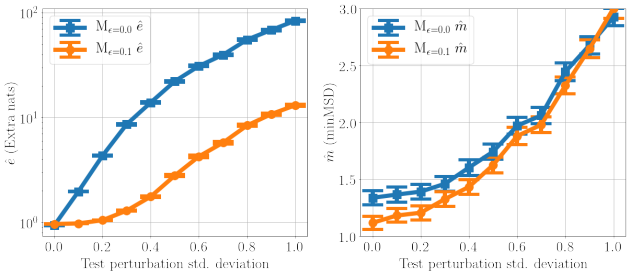


Figure 7: Evaluating the effects of noisy localization.

5. Conclusions

We present a multi-agent forecasting method, ESP, that outperforms state-of-the-art multi-agent forecasting methods on real (nuScenes) and simulated (CARLA) driving data. We also developed a novel ability, PRECOG, to condition forecasts on agent intentions. We showed conditional forecasts improve joint-agent and per-agent predictions, compared to unconditional forecasts used in prior work. Conditional forecasting can be used for planning, which we demonstrated with a novel multi-agent imitative planning objective. Future directions include conditional forecasting w.r.t. multiple agent intentions, useful for multi-AV coordination via communicated intent.

Acknowledgements

We thank Kate Rakelly, Angelos Filos, and Anca Dragan for their helpful feedback. This work was sponsored in part by IARPA (D17PC00340).

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2, 6

[2] D. Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012. 4

[3] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo. Context-aware trajectory prediction. *arXiv preprint arXiv:1705.02503*, 2017. 2

[4] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 1, 2, 5

[5] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998:746–752, 1998. 2

[6] N. Deo and M. M. Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based LSTMs. *arXiv preprint arXiv:1805.05499*, 2018. 2, 6

[7] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016. 3, 11

[8] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning (CoRL)*, pages 1–16, 2017. 2, 5, 12, 13

[9] P. Felsen, P. Lucey, and S. Ganguly. Where will they go? Predicting fine-grained adversarial multi-agent motion using conditional variational autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 732–747, 2018. 2

[10] T. Fernando, S. Denman, S. Sridharan, and C. Fookes. Soft + hardwired attention: An LSTM framework for human trajectory prediction and abnormal event detection. *Neural networks*, 108:466–478, 2018. 2, 6

[11] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan. Hierarchical game-theoretic planning for autonomous vehicles. *arXiv preprint arXiv:1810.05766*, 2018. 2

[12] W. Grathwohl, R. T. Chen, J. Betterncourt, I. Sutskever, and D. Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018. 3, 11

[13] J. Guan, Y. Yuan, K. M. Kitani, and N. Rhinehart. Generative hybrid representations for activity forecasting with no-regret learning. *arXiv preprint arXiv:1904.06250*, 2019. 3

[14] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 5, 6, 7, 14

[15] B. Ivanovic, E. Schmerling, K. Leung, and M. Pavone. Generative modeling of multimodal multi-human behavior. *arXiv preprint arXiv:1803.02015*, 2018. 2

[16] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10236–10245, 2018. 3, 11

[17] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[18] H. M. Le, Y. Yue, P. Carr, and P. Lucey. Coordinated multi-agent imitation learning. In *International Conference on Machine Learning*, pages 1995–2003, 2017. 2

[19] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker. DESIRE: Distant future prediction in dynamic scenes with interacting agents. In *Computer Vision and Pattern Recognition (CVPR)*, pages 336–345, 2017. 2, 5, 6, 7, 14

[20] N. Lee and K. M. Kitani. Predicting wide receiver trajectories in american football. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016. 2

[21] W.-C. Ma, D.-A. Huang, N. Lee, and K. M. Kitani. Forecasting interactive dynamics of pedestrians with fictitious play. In *Computer Vision and Pattern Recognition (CVPR)*, pages 4636–4644. IEEE, 2017. 2

[22] R. McAllister, Y. Gal, A. Kendall, M. Van Der Wilk, A. Shah, R. Cipolla, and A. V. Weller. Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2017. 4

[23] R. J. McCann et al. Existence and uniqueness of monotone measure-preserving maps. *Duke Mathematical Journal*, 1995. 3

[24] F. S. Melo and M. Veloso. Decentralized MDPs with sparse interactions. *Artificial Intelligence*, 175(11):1757–1789, 2011. 2

[25] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018. 2

[26] S. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. *arXiv preprint arXiv:1802.06338*, 2018. 2, 6

[27] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 261–268. IEEE, 2009. 6

[28] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989. 2

[29] N. Rhinehart, K. M. Kitani, and P. Vernaza. R2P2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *European Conference on Computer Vision (ECCV)*, September 2018. 3, 5, 6, 7, 11, 12, 13, 14

[30] N. Rhinehart, R. McAllister, and S. Levine. Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*, 2018. 2, 5, 13

[31] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018. 2

[32] C. Sun, P. Karlsson, J. Wu, J. B. Tenenbaum, and K. Murphy. Stochastic prediction of multi-agent interactions from partial observations. *arXiv preprint arXiv:1902.09641*, 2019. 2

[33] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993. 2

[34] E. Zhan, S. Zheng, Y. Yue, and P. Lucey. Generative multi-agent behavioral cloning. *arXiv preprint arXiv:1803.07612*, 2018. 2

[35] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu. Multi-agent tensor fusion for contextual trajectory prediction. *arXiv preprint arXiv:1904.04776*, 2019. 2

A. Planning and Forecasting Algorithms

To execute planning, we perform gradient ascent to approximately solve the optimization problem (11). Recall the latent joint behavior is $\mathbf{Z} \doteq \mathbf{Z}^{1:A}_{1:T}$, the latent human behavior is $\mathbf{Z}^h \doteq \mathbf{Z}^{2:A}_{1:T}$, and the robot behavior is $\mathbf{z}^r \doteq \mathbf{z}^1_{1:T}$. We approximate the expectation in (10) with a sample expectation over K samples from $p(\mathbf{Z}^h) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, denoted ${}^{1:K}\mathbf{z}^h$, where the k^{th} sample is ${}^k\mathbf{z}^h$. Each of these samples for the latent human behavior is combined with the single latent robot plan, each denoted ${}^k\mathbf{z} = [\mathbf{z}^r, {}^k\mathbf{z}^h]$. This batch is denoted ${}^{1:K}\mathbf{z}$. The approximation of (10) is then given as

$$\hat{L}({}^{1:K}\mathbf{z}, \mathcal{G}, \phi) = \frac{1}{K} \sum_{k=1}^{K} \log(q(f({}^k\mathbf{z})|\phi) p(\mathcal{G}|f({}^k\mathbf{z}), \phi)), \quad (15)$$

with $\hat{L}$ parameterized by (q, f, p), and f's dependence on ϕ dropped for notational brevity. The ${}^{1:K}\mathbf{z}^h$ is redrawn before each gradient ascent step on (15). This procedure is illustrated in Alg. 1.

Algorithm 1 MULTIIMITATIVEPLAN(q, f, p, ϕ, K)

Define $\hat{L}$ with q, f, p
Initialize $\mathbf{z}^r_{1:T} \sim \mathcal{N}(0, I)$
while not converged **do**
  ${}^{1:K}\mathbf{z}^h \overset{\text{iid}}{\sim} N(0, I)$
  $\mathbf{z}^r_{1:T} \leftarrow \mathbf{z}^r_{1:T} + \nabla_{\mathbf{z}^r_{1:T}} \hat{L}({}^{1:K}\mathbf{z}, \mathcal{G}, \phi)$
end while
return $\mathbf{z}^r_{1:T}$

In our implementation, we use $K = 12$, track the $\mathbf{z}^r_{1:T}$ that achieved the best $\hat{L}$ score, terminate the ascent if the best $\mathbf{z}^r_{1:T}$ has not improved in 10 steps, and return the corresponding best $\mathbf{z}^r_{1:T}$. To initialize $\mathbf{z}^r_{1:T}$ more robustly, we sample a full ${}^{1:K}\mathbf{z}$ multiple times (15), and use the $\mathbf{z}^r$ corresponding to the best $\hat{L}$. We can also run the planning over multiple initial samples of $\mathbf{z}^r_{1:T}$.

Now, we further detail how we can use this planning to perform goal-conditioned forecasting. As described in Sec. 4.5, we model intentions in our experiments by defining our goal-likelihood $p(\mathcal{G}|\mathbf{S}_{1:T}, \phi) = \mathcal{N}(\mathbf{S}^r_T; \mathbf{S}^{*r}_T, 0.1 \cdot \mathbf{I})$, *i.e.* a normal distribution at the controlled agent's last true future position, $\mathbf{S}^{*r}_T$. In general, we can pass any final desired robot position, $\mathbf{s}^{\dagger r}_T$ as the mean of this distribution. Then, we perform intention-conditioned forecasting on a specific scene ϕ to a specific robot goal $\mathbf{s}^{\dagger r}_T$, with our trained multi-agent density q, defined by f. This forecasting is performed by first planning $\mathbf{z}^r$ according to Alg. 1, then sampling $\mathbf{Z}^h$ again to generate stochastic joint outcomes, conditioned on the robot's intentions. This procedure is illustrated in Algs. 2 and 3.

Algorithm 2 PRECOG$(q, f, p, \mathbf{s}^{\dagger r}_T, \phi, K)$

$\mathbf{z}^r \leftarrow$ MULTIIMITATIVEPLAN(q, f, p, ϕ, K)
Sample ${}^{1:K}\mathbf{z}^h_{1:T} \overset{\text{iid}}{\sim} N(0, I)$
Forecast ${}^{1:K}\mathbf{s}^{1:A}_{1:T} \leftarrow f({}^{1:K}\mathbf{z}^{1:A}_{1:T}, \phi)$
return ${}^{1:K}\mathbf{s}^{1:A}_{1:T}$

Algorithm 3 POSPRECOG$(q, f, \mathbf{s}^{\dagger r}_T, \phi, K)$

Define $p(\mathcal{G}|\mathbf{S}_{1:T}, \phi) = \mathcal{N}(\mathbf{S}^r_T; \mathbf{s}^{\dagger r}_T, 0.1 \cdot I)$
return PRECOG$(q, f, p, \mathbf{s}^{\dagger r}_T, \phi, K)$

B. Alternate Joint PDF forms

The original joint can be expanded over each agent:

$$q(\mathbf{S}|\phi) = \prod_{t=1}^{T} q(\mathbf{S}_t|\mathbf{S}_{1:t-1}, \phi) = \prod_{t=1}^{T} \prod_{a=1}^{A} \mathcal{N}(\mathbf{S}^a_t; \boldsymbol{\mu}^a_t, \boldsymbol{\Sigma}^a_t).$$

Additionally, the change-of-variables rule yields an equivalent density [7, 12, 16, 29]:

$$q(\mathbf{S}|\phi) = \mathcal{N}(f^{-1}(\mathbf{S}; \phi); 0, I) \left| \det \frac{\mathrm{d}f}{\mathrm{d}\mathbf{Z}} \Big|_{\mathbf{Z} = f^{-1}(\mathbf{S};\phi)} \right|^{-1},$$

We can derive expressions of these terms with the rollout equation (6), reproduced here as (16), which implicitly defines f:

$$\mathbf{S}^a_t = \underbrace{2\mathbf{S}^a_{t-1} - \mathbf{S}^a_{t-2} + \overbrace{m^a_\theta(\mathbf{S}_{1:t-1}, \phi)}^{\mathbf{m}^a_t}}_{\boldsymbol{\mu}^a_t} + \underbrace{\sigma^a_\theta(\mathbf{S}_{1:t-1}, \phi)}_{\boldsymbol{\sigma}^a_t} \cdot \mathbf{Z}^a_t. \quad (16)$$

The full Jacobian is given as:

$$\frac{\partial f}{\partial \mathbf{Z}} = \begin{bmatrix} \frac{\partial \mathbf{S}_1}{\partial \mathbf{Z}_1} & 0 & \dots & 0 \\ \frac{\partial \mathbf{S}_2}{\partial \mathbf{Z}_1} & \frac{\partial \mathbf{S}_2}{\partial \mathbf{Z}_2} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \frac{\partial \mathbf{S}_T}{\partial \mathbf{Z}_1} & \frac{\partial \mathbf{S}_T}{\partial \mathbf{Z}_2} & \dots & \frac{\partial \mathbf{S}_T}{\partial \mathbf{Z}_T} \end{bmatrix},$$

where

$$\frac{\partial \mathbf{S}_t}{\partial \mathbf{Z}_t} = \begin{bmatrix} \sigma^1_t & 0 & \dots & 0 \\ \frac{\partial \mathbf{S}^2_t}{\partial \mathbf{Z}^1_t} & \sigma^2_t & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \frac{\partial \mathbf{S}^A_t}{\partial \mathbf{Z}^1_t} & \frac{\partial \mathbf{S}^A_t}{\partial \mathbf{Z}^2_t} & \dots & \sigma^A_t \end{bmatrix}$$

Due to the block triangular nature of the Jacobian and

applying Laplace expansion along the diagonal:

$$\det \frac{\mathrm{d}f}{\mathrm{d}\mathbf{Z}} = \prod_t \det \frac{\partial \mathbf{S}_t}{\partial \mathbf{Z}_t} = \prod_{t=1}^{T} \prod_{a=1}^{A} \det \sigma_t^a(\mathbf{S}_{1:t-1}, \phi).$$

Finally, $\mathbf{Z} = f^{-1}(\mathbf{S}; \phi)$ is given by computing each $\mathbf{Z}_t^a = (\sigma_t^a(\mathbf{S}_{1:t-1}, \phi))^{-1} (\mathbf{S}_t^a - \mu_t^a(\mathbf{S}_{1:t-1}, \phi))$. Algorithmically, the functions f and f^{-1} are implemented separately, each with a double for-loop over agents and time. We use the following checks to ensure f is a bijection: $|\mathbf{Z} - f^{-1}(f(\mathbf{Z}), \phi), \phi)|_\infty < \epsilon$, $|\mathbf{S} - f(f^{-1}(\mathbf{S}, \phi), \phi)|_\infty < \epsilon$.

C. Architecture and Training Details

Both past and future trajectories for each agent are represented in each agent's own *local* coordinate frame at $t = 0$, with agent's forward axis pointing along the agent's yaw at $t = 0$. Each agent a observes positions of the other agents in the coordinate frame of agent a. We use a 9-layer fully-convolutional network with stride 1 and 32 channels per layer, and kernel sizes of 3×3, to process χ into a feature grid $\mathbf{\Gamma}$ at the same spatial resolution as χ. The LIDAR is mounted on the first agent, thus it is generally more informative about nearby agents. This enables the prediction to be learned *relative* to the agent, with global context obtained by feature map interpolation. At each time step, each agent's predicted future position $\mathbf{s}_t^a$ is bilinearly-interpolated into $\mathbf{\Gamma}$: $\mathbf{\Gamma}(\mathbf{s}_t^a)$, which ensures $\mathrm{d}\mathbf{\Gamma}(\mathbf{s}_t^a)/\mathrm{d}\mathbf{s}_t^a$ exists. The "SocialMapFeat" component performs this interpolation by converting the positions (in meters) to feature grid coordinates (in 0.5 $^{\text{meters}}/_{\text{cell}}$), and bilinearly-interpolating each into the feature map $\mathbf{\Gamma}$. The interpolation is performed by retrieving the features at the corners of the nearest unit square to the current continuous position.

We also experimented with an additional featurization scheme on the nuScenes data. Instead of interpolating only at $\mathbf{s}_t^a$, we interpolated at nearby positions subsampled from arcs relative to $\mathbf{s}_t^a$ at various radii. By letting $\mathbf{s}_t^a - \mathbf{s}_{t-1}^a$ define the predicted orientation, the arcs were generated by evenly sampling 7 points along arcs of length $5\pi/4$ at radii $[1, 2, 4, 8, 16, 32]$ meters, which loosely simulates the forecasted agent's future field-of-view. The midpoint of each arc lied along the ray from $\mathbf{s}_{t-1}^a$ through $\mathbf{s}_t^a$. Without this scheme, the SocialMapFeat is of size $8A$. With it, the SocialMapFeat is of size $8(A + 7 \cdot 6)$ (8 is the size of the last dimension of Γ, 7 is the number of points per arc, and 6 is the number of arcs). We found this approach to yield superior performace in nuScenes, and employed it in the R2P2 baseline, as well as all of our methods. The full details of the architecture are provided in Tab. 3.

Finally, we performed additional featurization in the nuScenes setting (for all methods) by replacing χ with a *signed-distance transform*, similar to [29]. It provides a spatially-smoother input to the convolutional network, which we found augmented performance. The signed distance transform (SDT) of $\chi_c \in \mathbb{R}^{H \times W}$ can be computed by first binarizing to $\chi_c \in \{0, 1\}^{H \times W}$ and using the Euclidean distance transform (DT), which is commonly provided (*e.g.* in `scipy`). We compute it by binarizing with threshold τ: $\mathrm{SDT}(\chi_c, \tau) = \mathrm{DT}(\chi_c \geq \tau) - \mathrm{DT}(\chi_c < \tau)$, then clipping the result to $[-10, 1]$, and finally normalizing to $[0, 1]$. For LIDAR channels, we use $\tau = 5$. When we use the already-binarized road prior, binarization is unnecessary.

Figure 8: Images from the CARLA simulator [8]. *Left:* frontal view. *Right:* overhead view.

In Fig. 9, we illustrate various forms of the probabilistic graphical models corresponding to our main model (ESP), a baseline model (R2P2-MA), and how the assignment of latent variables ($\mathbf{Z}$) in these models affects the production of the states ($\mathbf{S}$).

We trained our model with stochastic gradient descent using the Adam optimizer with learning rate $1 \cdot 10^{-4}$, with minibatch size 10, until validation-set performance (of $\hat{e}$, as discussed in the main paper) showed no improvement for a period of 10 epochs.

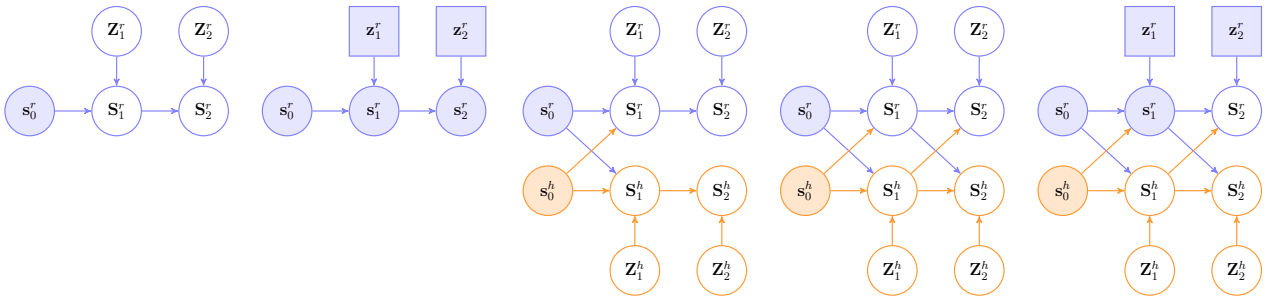
D. Baseline Implementations

SocialGAN We used the public implementation available at `https://github.com/agrimgupta92/sgan`. We found the default `train.py` parameters yielded poor performance in our domain. We achieved significantly better SocialGAN performance by using the network parameters in the `run_traj.sh` script. In contrast to SocialGAN, we model joint trajectories, and can compute likelihoods for planning (and for $\hat{e}$).

DESIRE We re-implemented DESIRE following the authors' description in the paper and supplement. In our domain, χ is purely LIDAR-based, whereas their model combines image-based semantic segmentation features into the same coordinate frame. We found most provided parameters to work well, except those related to the re-ranking component. The re-ranking often did not improve the trajectories. The best results were obtained with 1 re-ranking

Table 3: Detailed Co-Influence Architecture that implements $\mathbf{s}_{1:T}^{1:A} = f(\mathbf{z}_{1:T}^{1:A}, \phi)$. Typically, $T = 20$, $A = 5$, $D = 2$. In CARLA, $H = W = 100$. In nuScenes, $H = W = 200$.

Component	Input [dimensionality]	Layer or Operation	Output [dimensionality]	Details
Static featurization of context: $\phi = \{\chi, \mathbf{s}_{-\tau:0}^{1:A}\}$. *Shared parameters for each agent.*				
MapFeat	$\chi\ [H, W, 2]$	2D Convolution	${}^{1}\chi\ [H, W, 32]$	3×3 stride 1, ReLu
MapFeat	${}^{i-1}\chi\ [H, W, 32]$	2D Convolution	${}^{i}\chi\ [H, W, 32]$	3×3 stride 1, ReLu, $i \in [2, \ldots, 8]$
MapFeat	${}^{8}\chi\ [H, W, 32]$	2D Convolution	$\mathbf{\Gamma}\ [H, W, 8]$	3×3 stride 1, ReLu
PastRNN	$\mathbf{s}_{-\tau:0}^{1:A}\ [\tau + 1, AD]$	RNN	$\alpha\ [32]$	GRU across time dimension
Dynamic generation via double loop: for $t \in \{0, \ldots, T-1\}$, for $a \in \{1, \ldots, A\}$. *Separate parameters for each agent.*				
SocialFeat	$\mathbf{s}_t^{1:A}\ [AD]$	$\mathbf{s}_t^a - \mathbf{s}_t^b, b \in \{1..A\} \setminus a$	${}^{0}\eta_t^a\ [AD - D]$	Agent displacements
SocialFeatMLP	${}^{0}\eta_t^a\ [AD - D]$	Affine (FC)	${}^{1}\eta_t^a\ [200]$	Tanh activation
SocialFeatMLP	${}^{1}\eta_t^a\ [200]$	Affine (FC)	${}^{2}\eta_t^a\ [50]$	Identity activation
SocialMapFeat	$\mathbf{s}_t^{1:A}\ [AD]$	Interpolate	$\gamma_t^a = \mathbf{\Gamma}(\mathbf{s}_t^1) \oplus \cdots \oplus \mathbf{\Gamma}(\mathbf{s}_t^A)\ [8A]$	Differentiable interpolation, concat. ($\oplus$)
JointFeat	$\gamma_t^a, \mathbf{s}_0^{1:A}, {}^{2}\eta_a, \alpha$	$\gamma_t^a \oplus \mathbf{s}_0^{1:A} \oplus {}^{2}\eta_a \oplus \alpha$	$\rho_t^a\ [8A + AD + 50 + 32]$	Concatenate ($\oplus$)
FutureRNN	$\rho_t^a\ [8A + AD + 50 + 32]$	RNN	${}^{1}\rho_t^a\ [50]$	GRU
FutureMLP	${}^{1}\rho_t^a\ [50]$	Affine (FC)	${}^{2}\rho_t^a\ [200]$	Tanh activation
FutureMLP	${}^{2}\rho_t^a [200]$	Affine (FC)	$m_t^a\ [D],\ \xi_t^a\ [D, D]$	Identity activation
MatrixExp	$\xi_t^a\ [D, D]$	$\mathrm{expm}(\xi_t^a + \xi_t^{a,\mathrm{transpose}})$	$\boldsymbol{\sigma}_t^a\ [D, D]$	Differentiable Matrix Exponential [29]
VerletStep	$\mathbf{s}_t, \mathbf{s}_{t-1}, m_t^a, \boldsymbol{\sigma}_t^a, \mathbf{z}_t^a$	$2\mathbf{s}_t - \mathbf{s}_{t-1} + m_t^a + \boldsymbol{\sigma}_t^a \mathbf{z}_t^a$	$\mathbf{s}_{t+1}^a\ [D]$	(Eq. 6)



(a) R2P2 forecast [29] (b) DIM planning [30] (c) R2P2-MA forecast (d) ESP forecast (e) PRECOG planning

Figure 9: Graphical model comparison between prior work (Fig. 9a, Fig. 9b); a baseline we used (Fig. 9c); and our proposed methods (Fig. 9d, Fig. 9e). Shaded nodes represent observed or determined variables, and square nodes represent robot decisions. The intent of vehicles is realted to their latent variables $\mathbf{Z}$ which determines the range of reactions one vehicles will have given the state of other vehicles $\mathbf{S}$. Future reactions are always unknown in the case of the human drivers ("h" superscript), but can be decided in the case of robot ("r" superscipt) planning. How vehicles react affects—and induces uncertianty into—the multi-agent system state $\mathbf{S}$.

step. Whereas DESIRE is trained with a single-agent evidence lower bound (ELBO), our model jointly models multiple agents with an exact likelihood. As DESIRE does not compute multi-agent likelihoods, we cannot compute its $\hat{e}$, nor use it for planning in a multi-agent setting.

R2P2 We re-implemented R2P2 following the authors' description in the paper and supplement. We extended R2P2 to the multi-agent setting and use it as our R2P2-MA model; R2P2 does not jointly model agents. We can compute R2P2's likelihood, and therefore $\hat{e}$, by assuming independence across agents: $q(\mathbf{S}|\phi) = \prod_{a=1}^{A} q^a(\mathbf{S}^a|\phi)$. Note that since this joint likelihood does not model agent's future actions to influence each other, R2P2 cannot be used for planning in a multi-agent setting. Fig. 9 compares the R2P2 baseline to our Co-Influence model.

E. CARLA Dataset Details

To remind the reader, we generated a realistic dataset for multi-agent trajectory forecasting and planning with the CARLA simulator [8]. Images from the simulator are shown Fig. 8. We ran the autopilot in `Town01` for over 900 episodes of 100 seconds each in the presence of 100 other vehicles, and recorded the trajectory of every vehicle and the autopilot's LIDAR observation. We randomized episodes to either train, validation, or test sets. We created sets of 60,701 train, 7586 validation, and 7567 test scenes, each with 2 seconds of past and 4 seconds of future position information at 5Hz. The dataset also includes 100 episodes obtained by following the same procedure in `Town02`. We used this data to further evaluate our Co-Influence model. We applied our saved models (the same models used to report results in the paper) to this data. We generated the CARLA data using version 0.8.4. We used the default ve-

Table 4: CARLA multi-agent forecasting evaluation. All CARLA-trained models use `Town01 Train` only, and are tested on `Town01 Test`. Mean scores (and their standard errors) of sample quality $\hat{m}$ (13), and log likelihood $\hat{e}$ (12), are shown. The "–" symbol indicates if an approach cannot compute likelihoods. The R2P2-MA generalizes the single-agent forecasting approach of [29]. Variants of our Co-Influence method (highlighted gray) mostly outperform prior work in the multi-agent CARLA setting. For single agent evaluations, see Tab. 5.nt

Approach	Test $\hat{m}_{K=12}$ (minMSD)	Test $\hat{e}$ (extra nats)	Test $\hat{m}_{K=12}$ (minMSD)	Test $\hat{e}$ (extra nats)	Test $\hat{m}_{K=12}$ (minMSD)	Test $\hat{e}$ (extra nats)	Test $\hat{m}_{K=12}$ (minMSD)	Test $\hat{e}$ (extra nats)
CARLA Town01 Test	2 agents		3 agents		4 agents		5 agents	
DESIRE [19]	1.656 ± 0.038	–	1.684 ± 0.031	–	2.425 ± 0.038	–	2.599 ± 0.029	–
SocialGAN [14]	0.842 ± 0.024	–	1.037 ± 0.030	–	1.386 ± 0.041	–	1.464 ± 0.028	–
R2P2-MA [29]	0.430 ± 0.016	0.669 ± 0.005	0.594 ± 0.015	0.645 ± 0.004	0.753 ± 0.015	0.649 ± 0.008	0.843 ± 0.014	$\mathbf{0.630 \pm 0.003}$
Ours: Co-Influ., no LIDAR	0.783 ± 0.022	0.714 ± 0.006	0.815 ± 0.020	0.668 ± 0.005	1.096 ± 0.020	0.684 ± 0.004	1.213 ± 0.019	0.670 ± 0.004
Ours: Co-Influence	$\mathbf{0.335 \pm 0.013}$	$\mathbf{0.634 \pm 0.005}$	$\mathbf{0.430 \pm 0.013}$	$\mathbf{0.613 \pm 0.004}$	$\mathbf{0.659 \pm 0.013}$	$\mathbf{0.647 \pm 0.004}$	$\mathbf{0.716 \pm 0.012}$	0.643 ± 0.003

hicle. We used a LIDAR position of $(0.0, 0.0, 2.5)$, with 32 channels, a range of 50, 100,000 points per second, a rotation frequency of 10, an upper FOV limit of 10, and a lower FOV limit of -30. We will release the 100GB of collected data.

Table 5: Performance in CARLA in the single-agent setting. For single agent forecasting, our model is identical to [29] (denoted by a $*$).

Approach	Test $\hat{m}_{K=12}$ (minMSD)	Test $\hat{e}$ (extra nats)
CARLA Town01 Test	1 agent	
DESIRE [19]	1.067 ± 0.040	–
SocialGAN [14]	0.921 ± 0.031	–
R2P2-MA [29]	$*$	$*$
Ours: Co-Influ., no LIDAR	0.496 ± 0.024	0.699 ± 0.006
Ours: Co-Influence	$\mathbf{0.136 \pm 0.010}$	$\mathbf{0.634 \pm 0.006}$

F. Additional Evaluation

We show additional evaluations on CARLA in Tab. 4. Table 4 shows the `Town01` of the models trained on `Town01` (on separate episodes). Models perform slightly better on the held-out `Town02` data. We suspect this is due to several factors: the `Town02` scene is physically smaller than `Town01`, so it is more easily congested when simulated with the same number of vehicles (100). Congestion leads to more stationary futures, which our model predicts well. Secondly, the LIDAR representation of roads and vehicles usually generalizes well. We show single-agent CARLA forecasting results in Tab. 5.

G. Additional Visualizations

We display additional visualization of our results in Figures 10, 11, 12, 13, and 14. In Fig. 10, we show additional forecasting results on the nuScenes dataset. In Fig. 11, we show additional forecasting results on the CARLA dataset. In Fig. 12, we show additional planning results on the CARLA dataset. In Fig. 13, we show additional planning results on the nuScenes dataset. In Fig. 14, we visualize the planning criterion ($\hat{L}$) across many different spatio-temporal goal positions in CARLA, which gives a qualitative interpretation of where the model prefers goal. In Fig. 15, we visualize the same criterion on nuScenes.

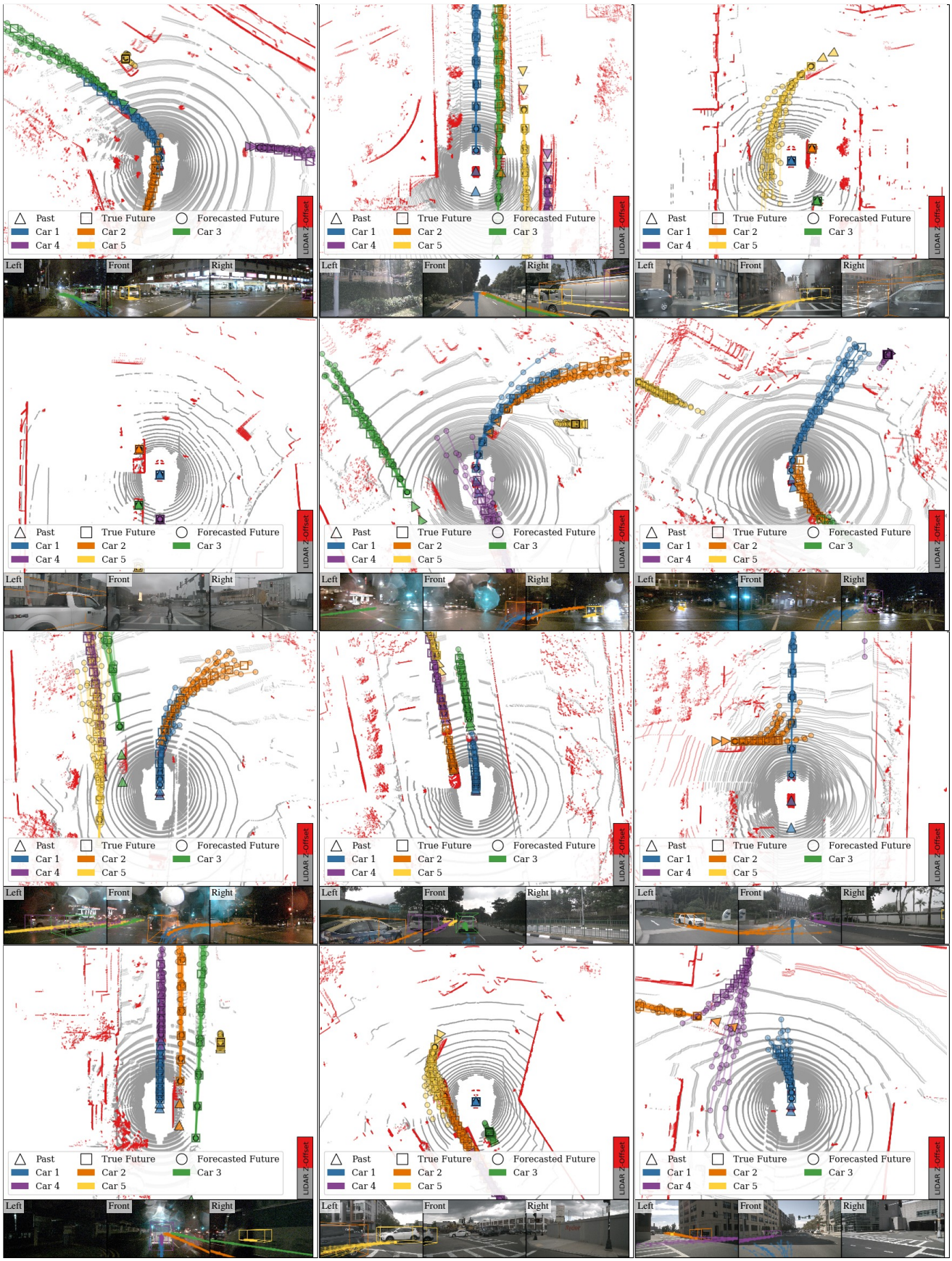


Figure 10: Example forecasting results on held-out nuScenes data with our learned ESP model. In each scene, 12 joint samples are shown, and LIDAR colors are discretized to near-ground and above-ground

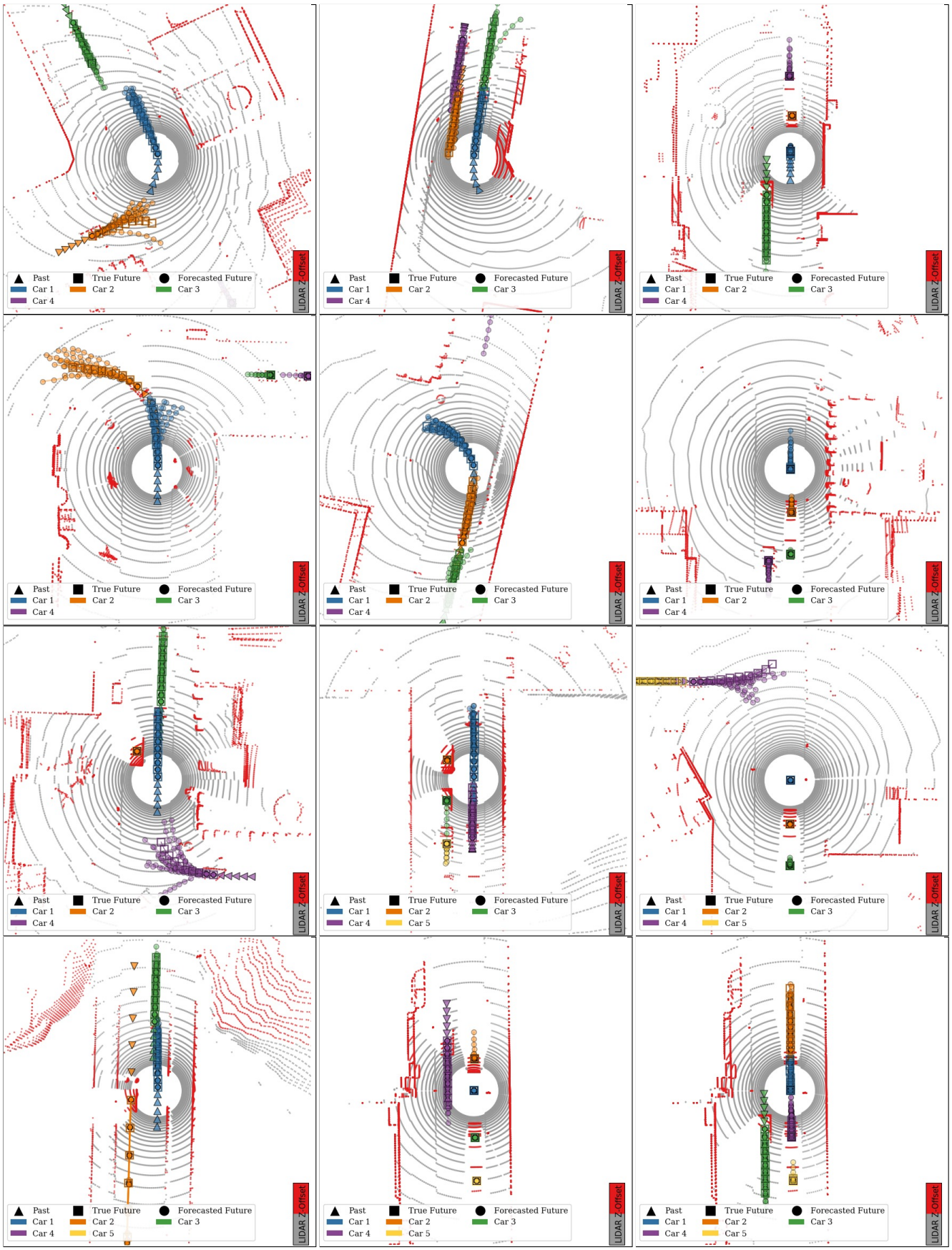
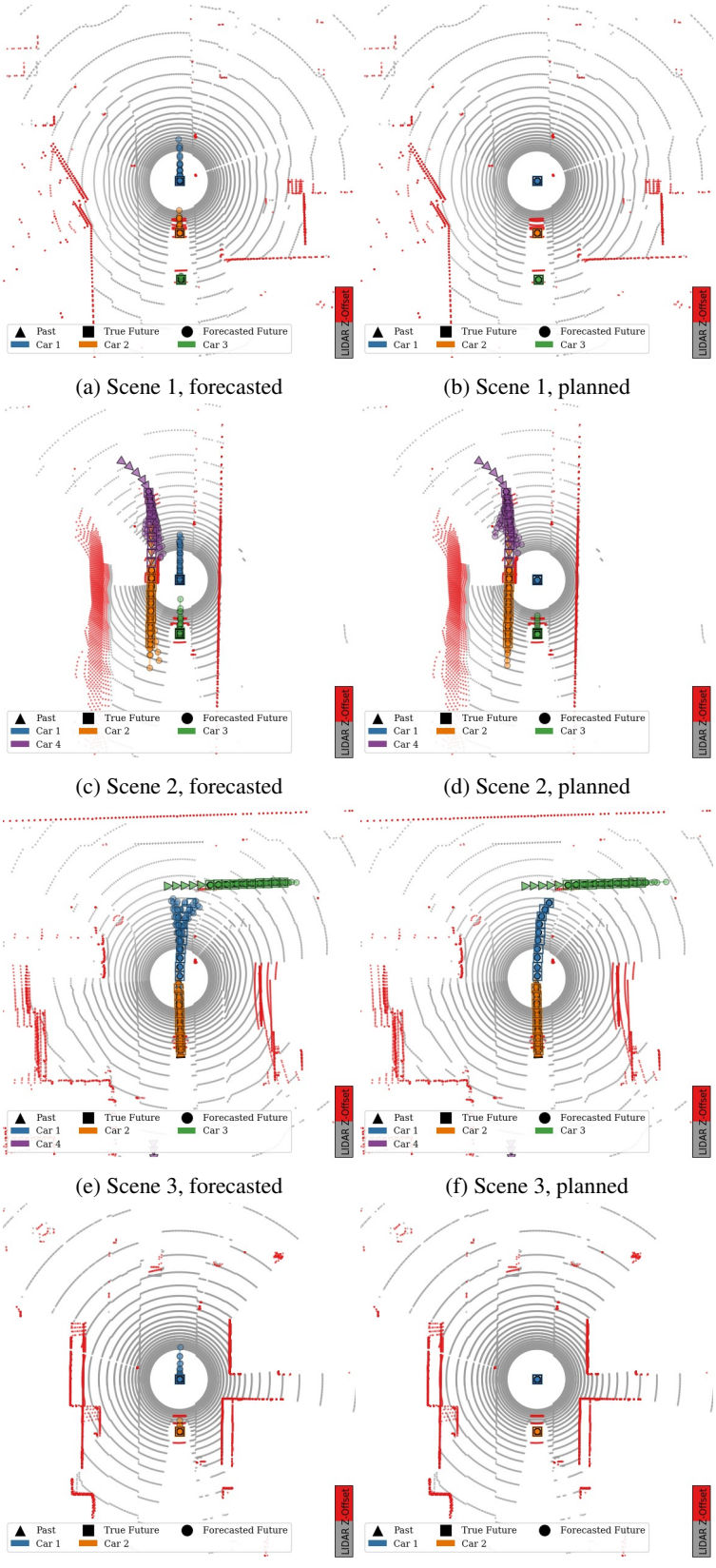


Figure 11: Example forecasting results on held-out CARLA data with our learned ESP model. In each scene, 12 joint samples are shown, and LIDAR colors are discretized to near-ground and above-ground.



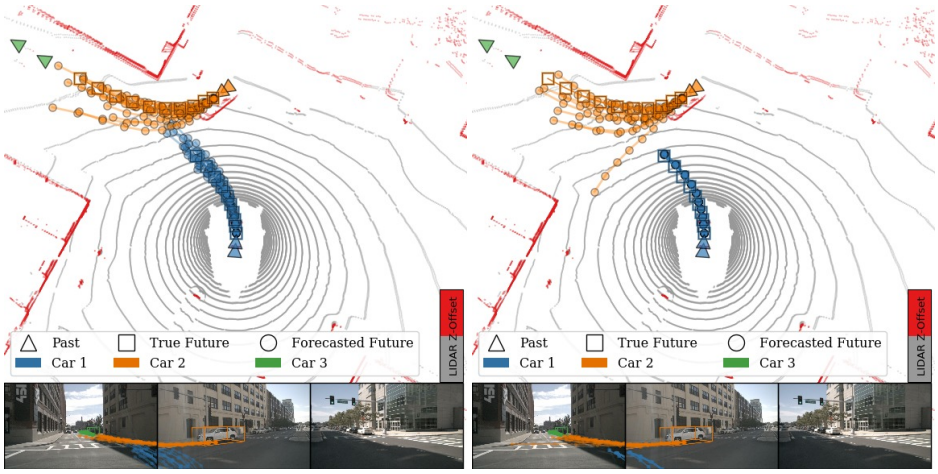
(a) Scene 1, forecasted (b) Scene 1, planned

(c) Scene 2, forecasted (d) Scene 2, planned

(e) Scene 3, forecasted (f) Scene 3, planned

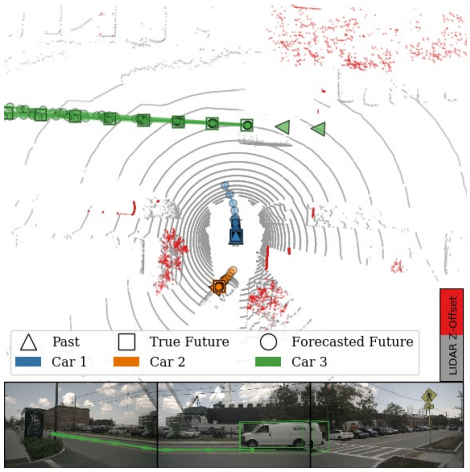
(g) Scene 4, forecasted (h) Scene 4, planned

Figure 12: Additional examples of *planned* multi-agent forecasting (PRECOG) with our learned model in CARLA. By using our planning approach and conditioning the robot on its true final position, our predictions for the robot become more accurate, and often our predictions of the other agent become more accurate.

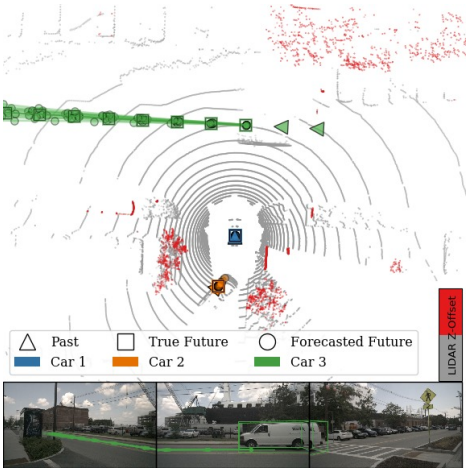


(a) Scene 1, forecasted

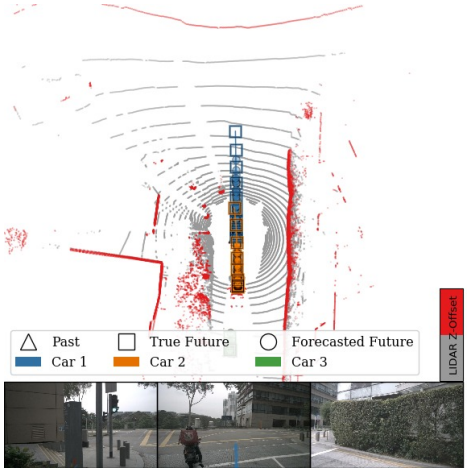
(b) Scene 1, planned



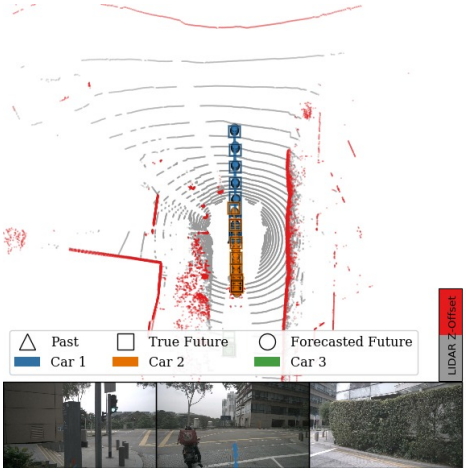
(c) Scene 2, forecasted



(d) Scene 2, planned



(e) Scene 3, forecasted



(f) Scene 3, planned

Figure 13: Additional examples of *planned* multi-agent forecasting (PRECOG) with our learned model in nuScenes. By using our planning approach and conditioning the robot on its true final position, our predictions for the robot become more accurate, and often our predictions of the other agent become more accurate.

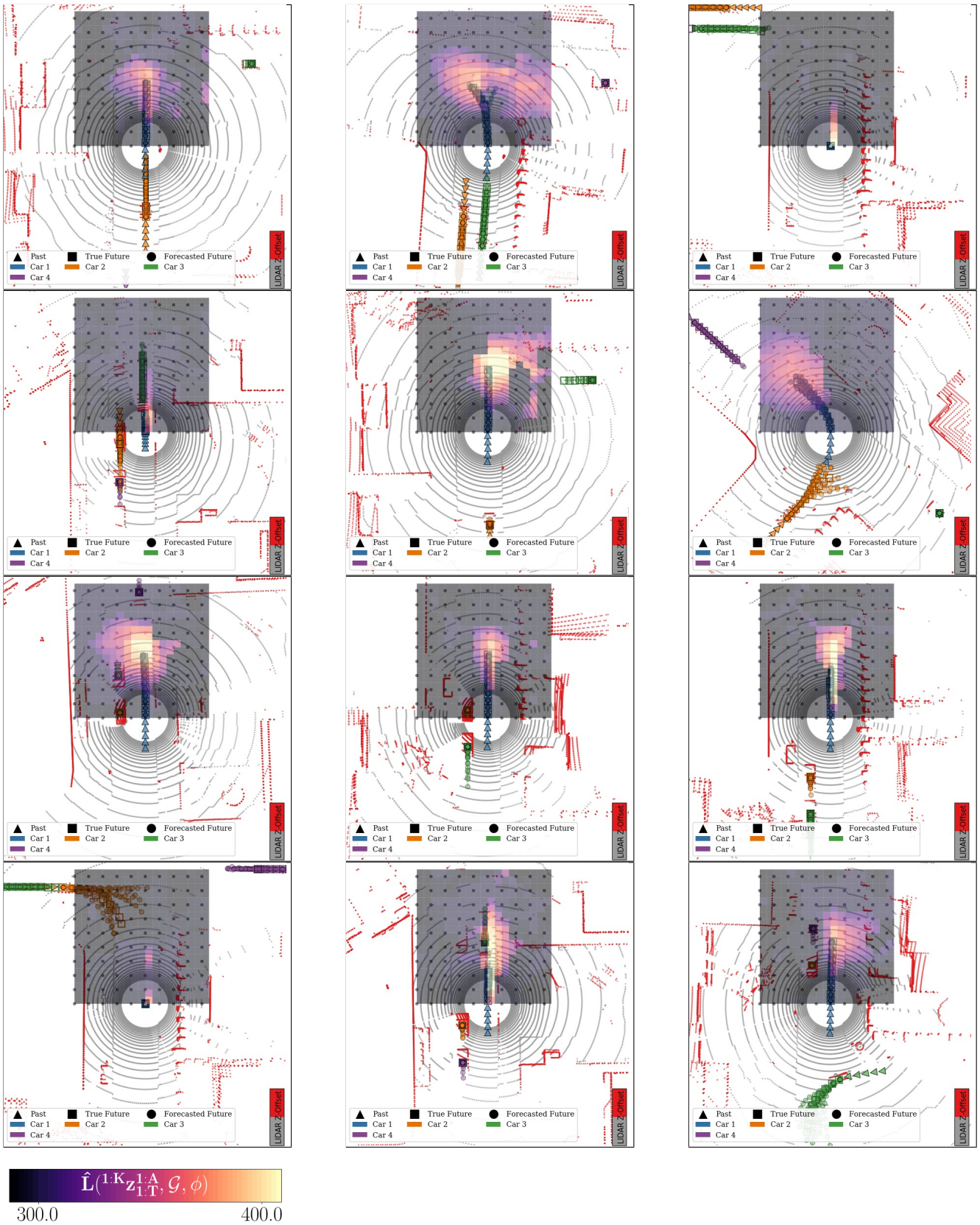


Figure 14: Plotting the planning criterion, $\hat{L}$, after planning to various positions (small circular dots in each plot) input to Alg. 3, with values interpolated between each position, in CARLA. The planning criterion input corresponds to a spatio-temporal goal at $T = 20$ in the future (4 seconds). The planning criterion prefers locations within its lane, unless it is uncertain about the possibility of turning. When the vehicle was stationary in the past, the planning criterion is highest at positions at or close in front of the vehicle.

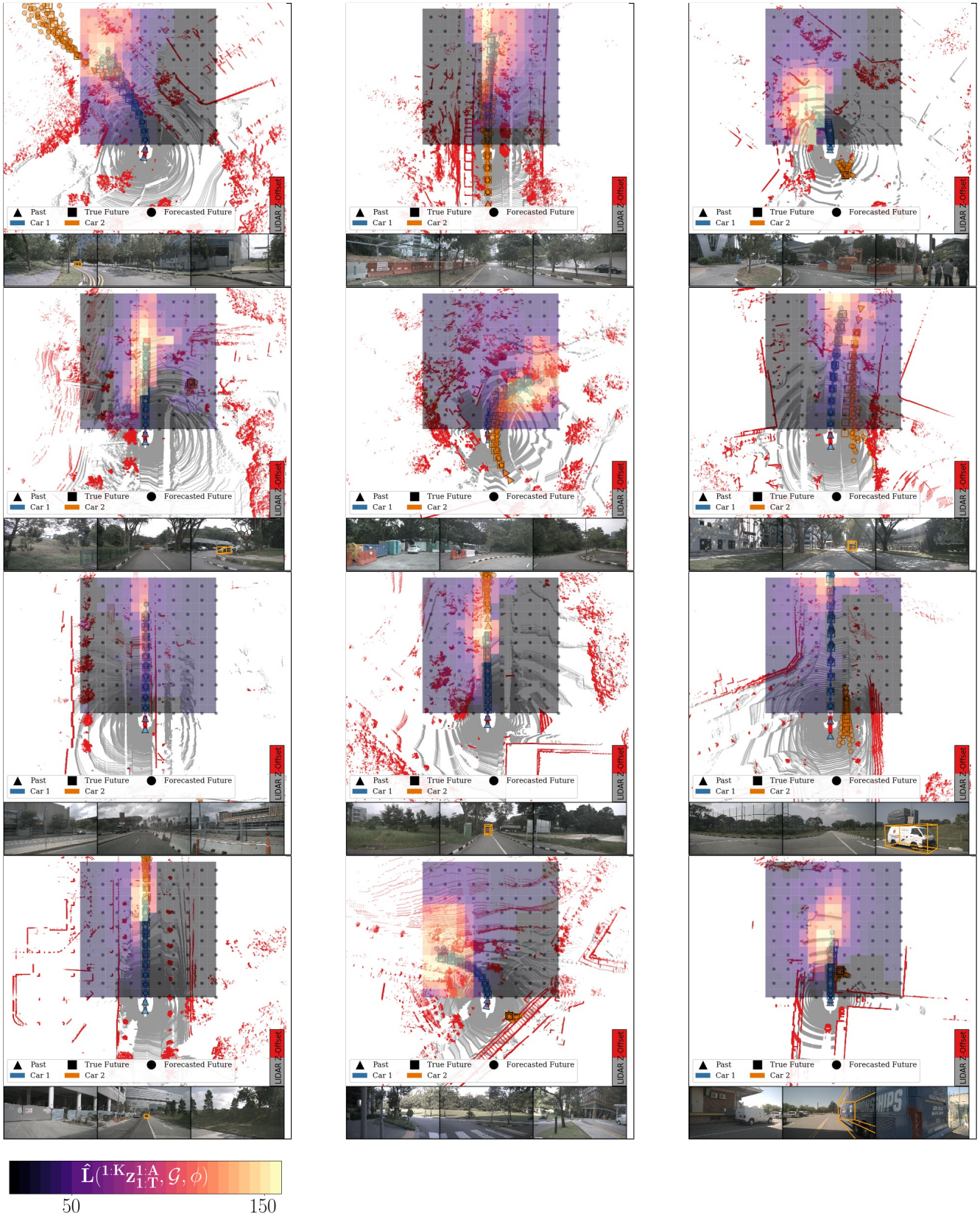


Figure 15: Plotting the planning criterion, $\hat{L}$, after planning to various positions (small circular dots in each plot) input to Alg. 3, with values interpolated between each position, in nuScenes. The planning criterion input corresponds to a spatio-temporal goal at $T = 20$ in the future (4 seconds). The planning criterion prefers locations within its lane, unless it is uncertain about the possibility of turning. When the vehicle was stationary in the past, the planning criterion is highest at positions at or close in front of the vehicle.