Proceedings of Machine Learning Research vol 168:1–14, 2022 4th Annual Conference on Learning for Dynamics and Control

Deep Interactive Motion Prediction and Planning: Playing Games with Motion Prediction Models

Jose L. Vazquez[1] VJOSE@ETHZ.CH
Alexander Liniger[1] ALEX.LINIGER@VISION.EE.ETHZ.CH
Wilko Schwarting[2] WILKOS@CSAIL.MIT.EDU
Daniela Rus[2] RUS@CSAIL.MIT.EDU
Luc Van Gool[1,3] VANGOOL@VISION.EE.ETHZ.CH
[1]*CVL - ETH Zurich,* [2]*CSAIL - MIT,* [3]*PSI - KU Leuven*

Editors: R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager, M. Kochenderfer

Abstract

In most classical Autonomous Vehicle (AV) stacks, the prediction and planning layers are separated, limiting the planner to react to predictions that are not informed by the planned trajectory of the AV. This work presents a module that tightly couples these layers via a game-theoretic Model Predictive Controller (MPC) that uses a novel interactive multi-agent neural network policy as part of its predictive model. In our setting, the MPC planner considers all the surrounding agents by informing the multi-agent policy with the planned state sequence. Fundamental to the success of our method is the design of a novel multi-agent policy network that can steer a vehicle given the state of the surrounding agents and the map information. The policy network is trained implicitly with ground-truth observation data using backpropagation through time and a differentiable dynamics model to roll out the trajectory forward in time. Finally, we show that our multi-agent policy network learns to drive while interacting with the environment, and, when combined with the game-theoretic MPC planner, can successfully generate interactive behaviors.
https://sites.google.com/view/deep-interactive-predict-plan

Keywords: Interactive Prediction, Model Predictive Control, Autonomous Vehicles

1. Introduction

Modern autonomy stacks, specifically those used for self-driving vehicles, consider prediction and planning different parts of the pipeline. However, this separation assumes a one-way interaction, where only the ego-vehicle is influenced by the other vehicles/agents. In reality, this assumption is not valid since the ego-vehicle also influences the other vehicles' decisions. Considering two-way interactions results in a game between all the agents, where all the vehicles plan simultaneously and find an equilibrium for the motion-planning game. Recent work developed methods to solve this problem (Liniger and Lygeros, 2020; Schwarting et al., 2021; Le Cleac'h et al., 2022). However, these methods require reward functions for all the agents, which limits these methods to specific problems like autonomous racing or toy examples. To overcome this limitation, several groups proposed to learn the reward function using inverse optimal control or inverse Reinforcement Learning (RL) (Sadigh et al., 2018; Zeng et al., 2019; Behbahani et al., 2019; Abbeel and Ng, 2004). However, this kind of approach has two disadvantages, first, inverse RL problems are often ambiguous and only work well with linear function approximators, shallow neural networks (Sadigh et al., 2018),

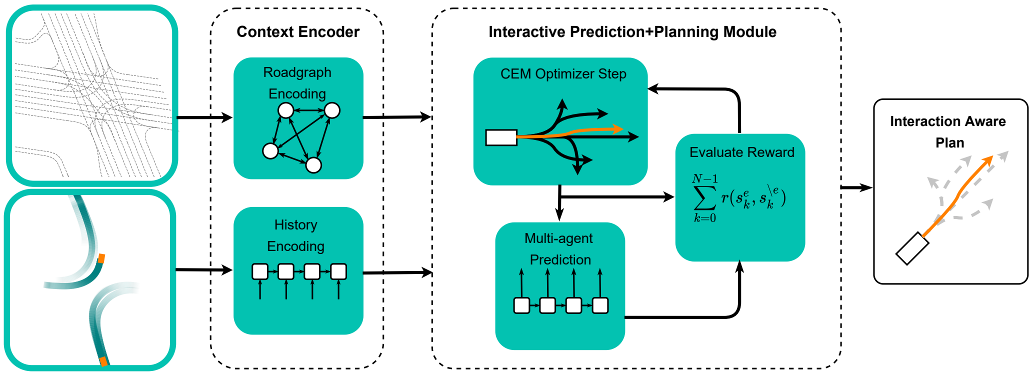

Figure 1: The joint prediction-planning module takes as inputs a past state buffer of all the vehicles in the scene and a HD-Map represented as a lane-graph. The Model Predictive Controller (MPC) inside the prediction-planning module uses the learned prediction module to internally simulate the multi-agent driving scenario.

or reward functions defined in the image space, which tend to be computationally and memory inefficient (Zeng et al., 2019). Second, after recovering the reward function, one still has to solve the computationally expensive game-theoretic planning problem (Sadigh et al., 2018; Schwarting et al., 2019). In our work, we propose an alternative route by directly learning a policy of the other agents, leveraging recent large-scale motion prediction datasets (Houston et al., 2020; Ettinger et al., 2021). Having access to a policy allows for best-response and leader-follower type algorithms to be used as a motion planner, drastically simplifying the interactive ego-motion planning problem.

By learning a policy, we can build on top of the ever-growing literature on deep motion prediction methods. However, as we will show, most of these approaches are highly optimized for accurate predictions but are not suited to reactively change to other vehicles in the prediction stage. Reactive prediction is a fundamental feature for our interactive planning approach, where the motion prediction policy of the other agents has to react to the proposed planned trajectory. Consequently, we propose using motion prediction tools such as road graph map integration and interaction layers but include them in an Interactive Multi-Agent Prediction (IMAP) policy. We further propose to train the policy in a model-based imitation learning approach. The core idea behind the IMAP policy is that at each step in the rollout/prediction, the model performs an interaction step both based on the recurrent state of the policy (intent) as well as the physical state of the agent. Additionally, the policy uses a map integration step where the relevant information is added from the HD-Map. Given all this information, the policy then computes actions for our differentiable dynamics model. Given the dynamics, we can roll out our policy and train it using an imitation loss on the physical states of the agents (position, heading, and velocity). We show that the policy can learn to predict and simulate a car on two large-scale motion prediction datasets, Lyft Level 5 (Houston et al., 2020) and the Waymo Open Motion Dataset (Ettinger et al., 2021). In summary, our contributions are:

- An interactive prediction-planning module to simultaneously predict and plan trajectories, using game-theoretic planning.
- Formulating motion prediction as a policy learning problem, which we tackled using a novel model-based imitation learning approach.

- Design of an interactive motion prediction policy, which captures different types of interaction between the multiple agents as well as the map.

2. Related Work

Prediction. The motion prediction or forecasting problem focuses on regressing the vehicles' future states conditioned on the historical states and possibly map information as a context variable. Recently, the prediction problem has been tackled extensively using deep neural networks (Ivanovic et al., 2018), but also model-based approaches like (Hu et al., 2019) are still used due to their interpretability and data efficiency. The availability of large-scale motion forecasting datasets is the driving force of deep learning models for prediction, but also the ability to encode interactive modeling directly in the network architecture as a form of inductive bias is fundamental. Modeling interactions between all the agents in the scene has been addressed by using multi-headed attention models (Mercat et al., 2020; Rella et al., 2021) or by using Graph Neural Networks (GNN) (Liang et al., 2020; Gao et al., 2020; Li et al., 2020; Kipf et al., 2018; Graber and Schwing, 2020). More closely related to our work in planning conditioned prediction models, we can find PiP (Song et al., 2020) where a Convolutional Neural Network (CNN) style architecture is used to encode the ground truth future of the vehicle. Although this model could react to a planned trajectory, it does not handle agent-to-agent interactions in the decoding stage. Trajectron++ (Salzmann et al., 2020) uses a similar approach to encode the ground truth future of the ego vehicle but using a Conditional Variational Autoencoder (CVAE) framework (Doersch, 2016) which better captures the distribution of possible paths an agent could follow conditioned on the ego-future but also lacks agent-to-agent interactions in the decoding stage. Trajectron++ was used for joint prediction and planning for human-robot interactions in (Nishimura et al., 2020). (Liu et al., 2021) presents a reactive prediction model that can be used for planning purposes, the approach uses pre-defined trajectory sets for each traffic participant, and it is trained via an augmented cross-entropy loss that penalizes collisions with other vehicles. Similar to our proposed method, TrafficSim (Suo et al., 2021) directly learns a policy in the decoder stage with the use of a differentiable simulation/dynamics model and a differentiable collision loss. Our approach aims to learn reactive behavior in continuous state-action spaces purely via imitation learning and without pre-defining a desired behavior in the loss function. Furthermore, we use a single recurrent cell that aims to capture the short and long term interactive behavior.

Planning. Planning in autonomous driving, especially in an imitation learning setting, is a difficult task, particularly due to different state distributions during training and evaluation (covariate-shift) (Ross et al., 2011). ChauffeurNet (Bansal et al., 2019) addresses this by utilizing a clever data-augmentation technique that re-generates a feasible trajectory subject to a perturbation in the initial state of the agent. In the interactive planning literature, Q-learning has been used to longitudinally control a vehicle using discrete accelerations (Leurent and Mercat, 2019). Game-theoretic trajectory optimization algorithms for non-cooperative scenarios have been addressed in (Spica et al., 2020; Wang et al., 2021; Williams et al., 2018; Fridovich-Keil et al., 2020; Di and Lamperski, 2019), but they require us to solve a full trajectory optimization problem for the other agents, which itself requires a pre-defined reward function. Reward functions have been successfully learned using inverse RL in (Sadigh et al., 2018; Schwarting et al., 2019; Peters et al., 2021; Mehr et al., 2021), but their structure is often limited to simple parameter vectors or too big image-based cost functions like in (Zeng et al., 2019). Our approach bypasses the reward function learning and uses a learned imitative policy that implicitly encodes the reward of the agents.

3. Interactive Motion Prediction

To train our IMAP policy, we employ a reinforcement and imitation learning-inspired approach. Specifically, we roll out our policy using a physical agent dynamics model. Given the rolled out agent trajectories, we can train the policy using a backpropagation trough time (BPTT) policy learning approach with an observation imitation loss. Thus, we can train a policy that generates physical actions only having access to observations. In the following, we will elaborate on this process, first introducing our model-based imitation learning approach given an abstract interactive policy and second introducing the exact design of our IMAP policy

3.1. Model-Based Imitation Learning

Theoretically, motion prediction can be posed as an imitation learning from observations problem (Torabi et al., 2019; Edwards et al., 2019), where we have observations of all the agents, and the goal is to find a policy that replicates the agents' decisions. The challenge with imitation learning from observations is that the actions are not available, rendering most imitation learning approaches unsuitable. However, in the case of motion forecasting, we have physical models that can predict the agents' motions since the main uncertainty does not come from the dynamics of the agents but the decision process. Considering that a physical model allows us to generate rollouts we can train the policies similarly to model-based RL methods (Hafner et al., 2020; Clavera et al., 2020), given that the model is differentiable with respect to the state and actions. To make this more concrete, let us first introduce our multi-agent notation. We use the subscript k to refer to a time step, and superscripts $i = 1, ..., I$ to refer to an agent, with I the number of agents. Furthermore, we use superscript $\neg i$ to refer to all agents except agent i. Second, we use a unicycle model as our motion model, as it well explains the motion of cars and other traffic agents. Thus, each agent is described by a state $s^i = [X, Y, \cos(\varphi), \sin(\varphi), v]$, where (X, Y) is the position (p), $\cos(\varphi), \sin(\varphi)$ is used as a smooth heading representation, and v is the forward velocity. Furthermore, an agent has the actions $a = [\alpha, r]$, where α is the acceleration and r the yaw rate. Hence, the agent's motion can be described by a discrete-time dynamical system of the form $s^i_{t+1} = f(s^i_t, a^i_t)$. Since the state s of the unicycle does not contain all information, such as the driving style, we use a recurrent policy, which has further access to a hidden state $h^i \in \mathbb{R}^H$. Finally, all agents have access to a map representation m. Therefore, we can define an abstract multi-agent policy $a = \pi(s^i, h^i, s^{\neg i}, h^{\neg i}, m)$. The policy gets as the input the physical ego state s^i and hidden state h^i, the physical state and hidden state of all the other agents ($s^{\neg i}$, $h^{\neg i}$), and finally the map representation m. In consequence, we can formulate the following model-based policy learning problem,

$$\begin{aligned}
\min_{\pi(\cdot)} \quad & \sum_{k=0}^{N} \sum_{i=1}^{I} \mathcal{L}(s^i_k, \hat{s}^i_k) \\
\text{s.t.} \quad & s^i_0 = \hat{s}^i_0 \\
& s^i_{k+1} = f(s^i_k, a^i_k) \\
& \{a^i_k, h^i_{k+1}\} = \pi(s^i_k, h^i_k, s^{\neg i}_k, h^{\neg i}_k, m) \\
& i = 1, ...I, \quad k = 0, ..., N
\end{aligned} \tag{1}$$

where, N is the training horizon, and $\mathcal{L}$ is an imitation cost function (in our case, the Huber loss) that penalizes differences between the state trajectory of the policy rollout and the ground truth measurements. Note that we encode the history of all the agents using the same policy, but using ground truth states (teacher forcing), according to $\{a_k^i, h_{k+1}^i\} = \pi(\hat{s}_k^i, h_k^i, \hat{s}_k^{\neg i}, h_k^{\neg i}, m)$, with $i = 1, ...I$, and $k = -K, ..., 0$ where K is the history length, finally, the hidden state is initialized with zero $h_{-K}^i = \mathbf{0}$. The resulting policy learning problem is fully differentiable since the forward model is differentiable. Thus, we can use a BPTT policy learning algorithm, which can efficiently learn with the use of the differentiable dynamics.

3.2. Interactive Policy

Given our model-based imitation learning problem (1), we have a method to train our IMAP policy. However, to achieve a good performance, the structure of the policy is fundamental. For our design, we recognize three fundamental interaction types that must be considered: first, long-term intention interaction, second, physical interactions, and finally, map interactions.

Policy Structure. As explained in the previous section, we use a recurrent policy structure, which we implement as a Gated Recurrent Unit (GRU) cell. The GRU cell takes the previous hidden state h as the recurrent input and additionally receives the concatenation of our three interaction modules, which will be explained next. Given the updated hidden state, we use two fully connected layers to produce the actions for the physical model. The output of our policy is a squashed Gaussian, which we sample from, this guarantees actions in physically reasonable bounds. However, it requires reparametrization (Doersch, 2016) of the sampled actions to train our policy.

Intention Interaction. The idea behind the intention interaction is to capture the long-term driving style, goal, and intention interaction between agents. We argue that these attributes are represented in the hidden state h of the policy. Hence, the interaction module considers the hidden representation of all agents. Inspired by (Mercat et al., 2020), we use a multi-headed dot-product attention mechanism (Vaswani et al., 2017) to model the interaction. Consequently, the embedding of an agent can be used in combination with linear layers to generate the queries, keys and values for the attention module. Finally, the output is separated into the individual agents and used within the policy as an input to the GRU cell.

Physical Interaction. The second type of interaction is considering interactions between the physical states s of agents. This interaction layer is intended for short-term behaviors and allows to learn collision avoidance and following skills. We formulate this layer as a GNN since it allows us to directly include state difference information in terms of edge features which is fundamental for this task. In our GNN, node features encode the physical state of each agent as well as agent metadata such as size and type, and edge features the difference in (x, y) coordinates. We consider a fully connected graph with self connections, do allow for a fast spread of information. We follow (Braso and Leal-Taixe, 2020) for the implementation details and use the mean as an aggregation function.

Map Interaction. The agents interact not only with each other but also with the environment/map. Thus, map interaction is a core building block of modern motion prediction networks. In our network, we use a VectorNet (Gao et al., 2020) based map encoder. The encoder uses a GNN on the polylines describing the map; note that we denote the graph of polylines as m. Given the encoded polylines, we use an attention mechanism to attend to the important parts of the map. We use a cross-attention mechanism where the queries for each agent are generated from the output of the physical interaction layer. This two-stage approach first lets the information be encoded into an embedded map graph, followed by a learnable extraction mechanism. Note that the approach is also computationally efficient since the map embedding is shared across all agents in the scene.

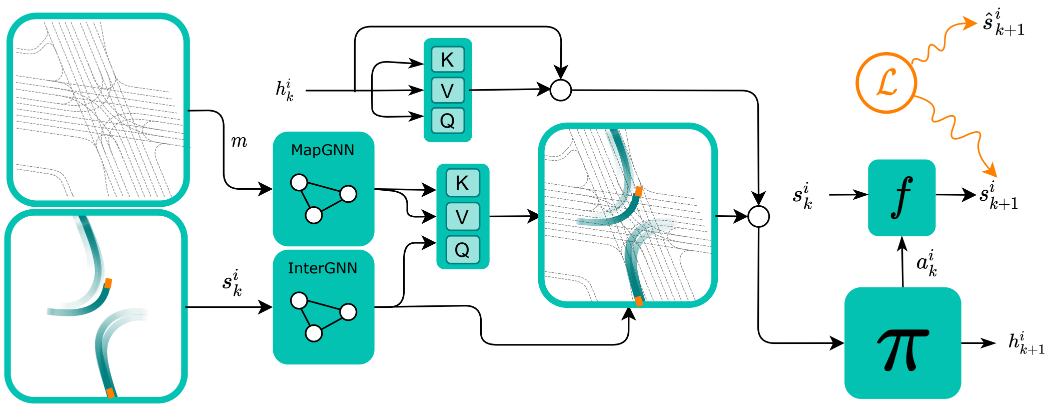

Figure 2: The recurrent IMAP policy π fuses the physical states s_i^k and map information m into a share embedding representation that is then used to to recursively control the dynamics model f. The single recurrent model is used for the encoding and decoding stage and is trained via BPTT using a direct loss on the states/observations $\mathcal{L}(\hat{s}^i_{k+1}, s^i_{k+1})$. IMAP also uses the hidden embeddings to generate the Keys, Values and Queries (K, V, Q) used in the intention and map interaction network.

The complete architecture of the policy is shown in Fig. 2, where we can see how the three interaction modules and the recurrent policy are set up.

Discussion. The core idea behind the training scheme and the policy architecture is to train a model that can react to changing driving behaviors of other traffic agents. Accordingly, overemphasis on the encoding phase has to be avoided since otherwise, the agents only extrapolate, which can result in catastrophic failures (Saadatnejad et al., 2021). We achieve this by performing closed-loop training, see (1). Thus, we do not learn to predict but learn to drive like the recorded agents. Note that the policy design is also essential, especially including the information about the other agents and the map at every step. This allows learning fundamental skills like collision avoidance and lane following. Hence, our method blurs the line between motion prediction and end-to-end driving, with our method being able to control traffic agents. One crucial difference is that our policy is not goal or route driven since we do not know the route of other agents. This is in contrast to end-to-end networks, which are normally conditioned on a route plan (Zhang et al., 2021; Bansal et al., 2019).

4. Interactive Motion Planning

Motion planning among other agents can be understood as a non-zero-sum game, where all agents plan trajectories considering their reward function. The reward function is agent-specific and captures the fundamentals of driving, such as collision avoidance, path following, and comfort. Solving this game has several drawbacks; first, the reward functions of other agents are generally unknown, and second, finding a Nash or other equilibrium of this game can be computationally demanding.

Given our multi-agent policy, we can drastically simplify the problem since other agents' reward functions and even the best responses are directly embedded in the policy. Therefore, we can formulate the multi-agent motion planning problem by only optimizing over the action sequence $\mathbf{a^e} = [a^e_0, ..., a^e_{N-1}]$ of the ego agent by relying on the learned policy for all other agents. Accordingly, we only need to specify the ego reward function $r(s^e, s^{\neg e})$ and the interactive policy π. The resulting motion planning problem can be formulated as follows,

$$
\begin{aligned}
\max_{\mathbf{a^e}} \quad & \sum_{k=0}^{N-1} r(s_k^e, s_k^{\neg e}) \\
\text{s.t.} \quad & s_0^e = \hat{s}_0^e, \ s_0^i = \hat{s}_0^i \\
& s_{k+1}^e = f(s_k^e, a_k^e) \\
& s_{k+1}^i = f(s_k^i, \pi(s_k^i, h_k^i, s_k^{\neg i}, h_k^{\neg i}, m)) \\
& i = 1, ...I, \quad k = 0, ..., N
\end{aligned} \tag{2}
$$

where, identical to (1), $f()$ is the discrete-time unicycle dynamics. Note that to simplify the notation, we omitted the recurrent output of the policy. Furthermore, $\neg i$ in this context still refers to all agents expect i, thus, the ego agent is considered inside the IMAP policy.

We can solve this problem using derivative-free optimization solvers such as the Cross-Entropy Method (CEM). However, as usual in game theory, the order of play can have a drastic influence. Thus, in the following, we will propose two possible approaches, one resulting in a leader-follower style equilibrium and the second in a Nash style equilibrium. Note that these equilibria are for games in a trajectory space, similar to (Liniger and Lygeros, 2020; Williams et al., 2018), where each trajectory is interpreted as a strategy of the player.

Both approaches are based on best responses iterations, where agents iteratively update the strategy by the best possible actions given the current actions of the other agents. Our IMAP policy naturally reacts with a “best” response to a other agents, even if the trajectory is fixed before hand. In the implementation we add the ego agent in the IMAP policy, and use the MPC trajectory with teacher forcing in the rollout. Note the we do not know the reward function of the IMAP policy.

4.1. Iterative Leader-Follower MPC (ILF-MPC)

Our first approach is a leader-follower setup inspired by the Stackelberg equilibrium (Başar and Olsder, 1998), where the leader is the ego agent. The ego agent computes a set of possible trajectories at each iteration using a CEM approach. The other agents compute their best response for all the ego trajectories by rolling out the IMAP policy. Given the best response trajectories, the reward of all the ego trajectories is computed, including collision penalties. The mean of the best trajectories (elites) is then used to initialize the next CEM iteration. Ideally, the CEM method is run until convergence. However, if the method is terminated early, the only drawback is that the ego trajectory could be further improved but the behavior of the other agents is still captured by the IMAP policy.

4.2. Iterative Best-Response MPC (IBR-MPC)

The leader-follower MPC proposed in the last section gives much power to the ego-agent, which potentially overestimates its influence. Therefore, we also propose an iterative best-response approach inspired by the Nash equilibrium (Başar and Olsder, 1998), since all players play a best response if the method converges, the resulting equilibrium is Nash. Therefore, we change the order of play and first compute trajectories for all agents with our IMAP policy. Given this starting point, the ego agent computes the best response with respect to the fixed agent trajectories using the CEM solver. Given this trajectory, all the other agents compute the best response using the IMAP policy with the ego trajectory fixed. This approach is repeated for several iterations or until convergence. Note that this approach needs fewer rollouts of the IMAP policy, but more CEM iterations since, at each outer-iteration, the ego agent needs to find the best trajectory requiring multiple CEM iterations.

5. Results

The proposed interactive policy is tested in both the prediction and planning tasks. In the prediction task, the models are ablated against our re-implementation of the Argoverse 2019/2022 competition winner SAMPP (Mercat et al., 2020) based on standard single modal prediction metrics. Additionally, we show that adding the Non-Linear Least Square (NLLS) perturbation method proposed in (Bansal et al., 2019) does not hurt the nominal performance of our models. In the planning task, we explore a lane merge scenario and show how IBP-MPC and ILF-MPC can plan a lane change behavior while also maximizing the margin to the approaching vehicle. Additionally we show how ILF-MPC can take advantage of having access to the reactive prediction model in the optimization step to plan highly interactive trajectories.

5.1. Interactive Motion Prediction

Essential to the success of our joint prediction-planning approach is the simultaneous prediction for all agents in the scene. We perform an ablation study to demonstrate that our information sharing mechanisms capture the underlying interactions of urban driving. For the ablation study we consider two of the most used metrics in trajectory prediction Final Displacement Error (ADE) and Average Displacement Error (ADE). Similar to other work in the field of trajectory prediction we perform inference with ego-centered scene information, but in contrast, we evaluate the model using the ego-agent metrics (ADE^e, FDE^e) as well as the rest of the agents' ($\text{ADE}^{\neg e}$, $\text{FDE}^{\neg e}$). Note that in our study, we do not consider multi-modal predictions but compare how different information sharing mechanisms can learn the underlying interactive behavior of the scene.

Lyft level 5. The Lyft level 5 motion prediction dataset (Houston et al., 2020) consists of several full-length driving logs sampled at 10Hz. The logs contain information about the tracks of the perceived agents and the state of the ego vehicle. The level 5 dataset is not designed to output multi-agent motion sequences, therefore, both the train and validation datasets were preprocessed by subsampling the logs every 5 seconds and extracting all the motion sequences of the agents in the vicinity of the AV. Preprocessing the data allows for quicker training and evaluation on a single GPU. The models for the ablation study are trained only using tracks from vehicles (pedestrians and bicycles are removed from the dataset). The road graph information was resampled to have equidistantly spaced points for each polyline, similar to the polylines included in the Waymo dataset.

Waymo. The Waymo Open Motion Dataset (WOMD) (Ettinger et al., 2021) consists of 9 second sequences sampled at 10Hz each sample contains multi-agent sequences of up to 8 actors that are probably interacting with each other. The dataset also contains a road graph per sample represented as a sequence of equidistantly sampled polylines. The models trained for the ablation studies on the Waymo dataset include the actor type (vehicle, pedestrian, cyclist) represented as a one-hot encoded embedding. Even though the dataset has sequences of 8s the models trained for the following ablation study are trained with a 3s horizon.

Ablation Study. All of the models were trained with a 3s prediction horizon, for the level 5 dataset, we used a 2s encoding buffer, and for the WOMD dataset its standard 1s encoding buffer. During

Table 1: IMAP Policy Ablation Studies

Dataset	Lyft @ 3s				Waymo @ 3s			
Metric	ADE^e	FDE^e	ADE$^{\neg e}$	FDE$^{\neg e}$	ADE^e	FDE^e	ADE$^{\neg e}$	FDE$^{\neg e}$
SAMPP*	0.27	0.68	0.39	0.83	0.88	2.41	1.26	2.80
SAMPP*(plan)	0.06*	0.08*	0.41	0.84	0.31*	0.43*	1.49	3.06
No Interact.	0.20	0.56	0.32	0.77	0.74	2.19	1.09	2.46
+Interact.	0.20	0.54	0.30	0.69	0.65	1.87	0.99	2.13
+Map	0.17	0.46	0.29	0.67	0.60	1.71	0.97	2.04
+NLLS	0.17	0.46	0.28	0.65	0.60	1.73	0.97	2.03

training, with a probability $p = 0.1$, the data is perturbed using random agent scene centering and random $SE(2)$ scene transformation with uniform distribution $(\Delta x, \Delta y) \sim \mathcal{U}[-10m, 10m]$ and $\Delta\varphi \sim \mathcal{U}[-0.8rad, 0.8rad]$. The perturbations are essential to mitigate the bias introduced by scene ego centering. The ablation study, performed on the level 5 and WOMD validation set, shows the same trends in both datasets, the SAMPP* baseline is outperformed by our proposed model with a differentiable unicycle propagation model and a stochastic policy. Conditioning SAMPP* on the ego ground truth trajectory SAMPP*(plan) similar to (Salzmann et al., 2020) does not improve the predictions of the other agents, showing that such an approach is not suited for our interactive planning. Once the interaction component is added to our method all the metrics improve, especially FDE$^{\neg e}$. The addition of the map encoder helps to improve the predictions of all agents but it particularly helps the ego predictions showing that the bias introduced due to scene centering is not fully mitigated via the data perturbations. Lastly, the NLLS perturbation needed for the planning module is added and we show that it does not hurt the nominal performance of our model. Note that we only train a single-mode model. As a result, our IMAP policy is not competitive in the multi-model evaluation metrics used in the WOMD-leaderboard.

5.2. Interactive Motion Planning

The motion planning component of the proposed architecture is tested in a qualitative fashion by analyzing a lane change scenario. Therefore, we take a scene from WOMD with seven agents and generate a plan for the ego agent, whereas the others are controlled by the WOMD trained IMAP policy. Lane changes are the most common type of interaction found in everyday driving and can be challenging in classical driving stacks. Often, lane changes require cooperation from cars on the merging lane. This interaction is not neglected in our approach compared to sequential prediction and planning. Thus, this scenario is used to explore the subtle differences between both proposed MPCs. Additionally, an adversarial reward term is used in the ILF-MPC method to showcase how privileged information about the reaction of other agents aids the optimization problem at hand.

Best Response MPC vs. Leader Follower MPC. To compare IBR-MPC and ILF-MPC, we solve the same lane-changing planning problem with a prediction horizon of 5s. The reward function uses a distance to target vehicle cost as a proxy for a collision penalty $-1/||p_k^e - p_k^{\neg e}||_2$, a terminal desired heading reward $-\text{anglediff}(\varphi_T^e - \varphi^*)$, a regularization term on the acceleration and yaw rate commands $-||a_k^e||_2^2$ and finally the orthogonal distance of each point in the trajectory to the

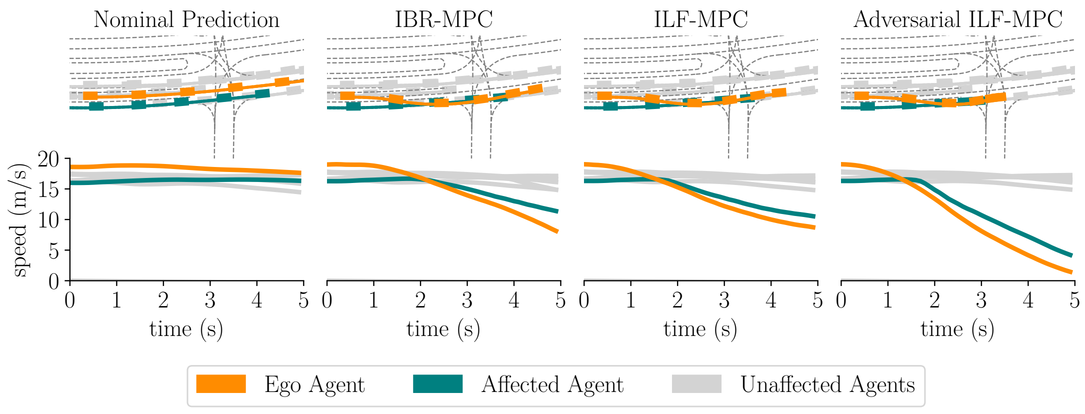

Figure 3: Planned trajectories and velocities, form left to right, nominal model prediction for all agents from a WOMD sample, lane-change with IBR-MPC, lane-change with ILF-MPC, adversarial stopping example with ILF-MPC.

reference lane $-\text{proj}(p_k^e)$. The combined reward allows the optimization problem to find a solution that completes the lane change, but it also emphasises a safe distance to other vehicles. In Fig. 3 we can see that IBC-MPC and ILF-MPC have almost identical outcomes, but in our experiments we found that ILF-MPC tends to converge significantly faster than IBR-MPC using the same cost function and hyperparameters for the CEM optimizer. Next, we show how the ILF-MPC exploits its privileged information on how the agents would react. We introduce an adversarial reward term of the form $-v_k^a$, this reward incentivizes the ego agent to adversarially decrease the speed of the target agent. In Fig. 3 we can see that the ego agent plans to position itself in front of the target agent while braking, the policy of the affected agent performs a braking maneuver to avoid collision with the ego agent. Note that the optimizer is pushing the prediction model outside of the training data distribution, nonetheless, the policy reacts meaningfully and avoids a rear-end collision. The target agent does not come to a complete stop at the end. One could use a longer prediction horizon to overcome this issue or consider terminal constraints in the MPC. Note that we implemented the same interactive planners with SAMPP*(plan), but the MPC was not able to perform a lane change.

6. Conclusion

We proposed a novel interactive motion prediction and planning framework, where the motion prediction model and the motion planner play a game with each other. Therefore, we proposed a novel way to formulate the motion prediction problem as a policy learning problem. The policy is learned using model-based imitation learning, which, combined with our IMAP policy, allows us to learn a highly interaction-aware prediction model/policy. Given this model, we propose two interactive motion planners based on best response iterations. One is inspired by a leader-follower structure and the other by the Nash equilibrium. Finally, we showed simulation results in realistic driving situations where our deep interactive motion planning and prediction framework can perform challenging lane changes and even adversarial tasks such as stopping another car. Both our proposed methods can plan challenging interactive motions, correctly predicting the influence of the own action on the other agents. However, ILF-MPC can, by design, plan more interactive trajectories but it should be studied if human drivers behave according to the modeled leader-follower structure.

References

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, 2004. doi: 10.1145/1015330.1015430.

Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst. In *Proceedings of Robotics: Science and Systems*, 2019. doi: 10.15607/rss.2019.xv.031.

Tamer Başar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory, 2nd Edition*. 1998. doi: 10.1137/1.9781611971132.

Feryal Behbahani, Kyriacos Shiarlis, Xi Chen, Vitaly Kurin, Sudhanshu Kasewa, Ciprian Stirbu, Joao Gomes, Supratik Paul, Frans A. Oliehoek, Joao Messias, and Shimon Whiteson. Learning from demonstration in the wild. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May:775–781, 2019. ISSN 10504729. doi: 10.1109/ICRA.2019.8794412.

Guillem Braso and Laura Leal-Taixe. Learning a Neural Solver for Multiple Object Tracking. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. doi: 10.1109/CVPR42600.2020.00628.

Ignasi Clavera, Violet Fu, and Pieter Abbeel. Model-Augmented Actor-Critic: Backpropagating Through Paths. *arXiv*, 2020. ISSN 23318422. URL https://arxiv.org/abs/2005.08068.

Bolei Di and Andrew Lamperski. Newton's Method and Differential Dynamic Programming for Unconstrained Nonlinear Dynamic Games. In *Proceedings of the IEEE Conference on Decision and Control*, volume 2019-December, 2019. doi: 10.1109/CDC40024.2019.9029237.

Carl Doersch. Tutorial on Variational Autoencoders. *arXiv*, 6 2016. URL http://arxiv.org/abs/1606.05908.

Ashley D. Edwards, Himanshu Sahni, Yannick Schroecker, and Charles L. Isbell. Imitating latent policies from observation. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, 2019.

Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large Scale Interactive Motion Forecasting for Autonomous Driving : The Waymo Open Motion Dataset. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 4 2021.

David Fridovich-Keil, Andrea Bajcsy, Jaime F. Fisac, Sylvia L. Herbert, Steven Wang, Anca D. Dragan, and Claire J. Tomlin. Confidence-aware motion prediction for real-time collision avoidance1. *International Journal of Robotics Research*, 39(2-3):250–265, 2020. ISSN 17413176. doi: 10.1177/0278364919859436.

Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. VectorNet: Encoding HD maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. doi: 10.1109/CVPR42600.2020.01154.

Colin Graber and Alexander G. Schwing. Dynamic Neural Relational Inference. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. doi: 10.1109/CVPR42600.2020.00854.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. *International Conference on Learning Representations*, 2020.

John Houston, Guido Zuidhof, Luca Bergamini, Yawei Ye, Ashesh Jain, Sammy Omari, Vladimir Iglovikov, and Peter Ondruska. One Thousand and One Hours: Self-driving Motion Prediction Dataset. *Conference on Robot Learning (CoRL)*, 2020. ISSN 23318422.

Yeping Hu, Liting Sun, and Masayoshi Tomizuka. Generic Prediction Architecture Considering both Rational and Irrational Driving Behaviors. In *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, 2019. doi: 10.1109/ITSC.2019.8917105.

Boris Ivanovic, Edward Schmerling, Karen Leung, and Marco Pavone. Generative Modeling of Multimodal Multi-Human Behavior. *IEEE International Conference on Intelligent Robots and Systems*, pages 3088–3095, 2018. ISSN 21530866. doi: 10.1109/IROS.2018.8594393.

Thomas Kipf, Ethan Fetaya, Kuan Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for Interacting systems. In *35th International Conference on Machine Learning, ICML 2018*, volume 6, 2018.

Edouard Leurent and Jean Mercat. Social Attention for Autonomous Decision-Making in Dense Traffic. *arXiv*, 2019. URL http://arxiv.org/abs/1911.12250.

Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. ALGAMES: a fast augmented Lagrangian solver for constrained dynamic games. *Autonomous Robots*, 46(1), 2022. ISSN 15737527. doi: 10.1007/s10514-021-10024-7.

Jiachen Li, Fan Yang, Masayoshi Tomizuka, and Chiho Choi. EvolveGraph: Multi-agent trajectory prediction with dynamic relational reasoning. In *Advances in Neural Information Processing Systems*, volume 2020-December, 2020.

Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning Lane Graph Representations for Motion Forecasting. *European Conference in Computer Vision (ECCV)*, 2020.

Alexander Liniger and John Lygeros. A Noncooperative Game Approach to Autonomous Racing. *IEEE Transactions on Control Systems Technology*, 28(3), 2020. ISSN 15580865. doi: 10.1109/TCST.2019.2895282.

Jerry Liu, Wenyuan Zeng, Raquel Urtasun, and Ersin Yumer. Deep Structured Reactive Planning. *International Conference on Robotics and Automation (ICRA)*, 2021.

Negar Mehr, Mingyu Wang, and Mac Schwager. Maximum-Entropy Multi-Agent Dynamic Games: Forward and Inverse Solutions. *arXiv*, 10 2021. URL http://arxiv.org/abs/2110.01027.

Jean Mercat, Thomas Gilles, Nicole El Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Pita Gil. Multi-Head Attention for Multi-Modal Joint Vehicle Motion Forecasting. In *Proceedings - IEEE International Conference on Robotics and Automation*, 2020. doi: 10.1109/ICRA40945.2020.9197340.

Haruki Nishimura, Boris Ivanovic, Adrien Gaidon, Marco Pavone, and Mac Schwager. Risk-sensitive sequential action control with multi-modal human trajectory forecasting for safe crowd-robot interaction. In *IEEE International Conference on Intelligent Robots and Systems*, 2020. doi: 10.1109/IROS45743.2020.9341469.

Lasse Peters, David Fridovich-Keil, Vicenç Rubies-Royo, Claire Tomlin, and Cyrill Stachniss. Inferring Objectives in Continuous Dynamic Games from Noise-Corrupted Partial State Observations. In *Robotics: Science and Systems*, 2021. doi: 10.15607/rss.2021.xvii.030.

Edoardo Mello Rella, Jan-Nico Zaech, Alexander Liniger, and Luc Van Gool. Decoder Fusion RNN: Context and Interaction Aware Decoders for Trajectory Prediction. *International Conference on Intelligent Robots and Systems (IROS)*, 8 2021.

Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. *Journal of Machine Learning Research*, 2011.

Saeed Saadatnejad, Mohammadhossein Bahari, Pedram Khorsandi, Mohammad Saneian, Seyed-Mohsen Moosavi-Dezfooli, and Alexandre Alahi. Are socially-aware trajectory prediction models really socially-aware? *arXiv*, 8 2021. URL http://arxiv.org/abs/2108.10879.

Dorsa Sadigh, Nick Landolfi, Shankar S. Sastry, Sanjit A. Seshia, and Anca D. Dragan. Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state. *Autonomous Robots*, 42(7):1405–1426, 10 2018. ISSN 15737527. doi: 10.1007/s10514-018-9746-1.

Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Multi-Agent Generative Trajectory Forecasting With Heterogeneous Data for Control. *European Conference on Computer Vision*, 1 2020.

Wilko Schwarting, Alyssa Pierson, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus. Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences of the United States of America*, 116(50):2492–24978, 2019. ISSN 10916490. doi: 10.1073/pnas.1820676116.

Wilko Schwarting, Alyssa Pierson, Sertac Karaman, and Daniela Rus. Stochastic Dynamic Games in Belief Space. *IEEE Transactions on Robotics*, 37(6), 2021. ISSN 19410468. doi: 10.1109/TRO.2021.3075376.

Haoran Song, Wenchao Ding, Yuxuan Chen, Shaojie Shen, Michael Yu Wang, and Qifeng Chen. PiP: Planning-informed Trajectory Prediction for Autonomous Driving. *European Conference on Computer Vision (ECCV)*, 2020.

Riccardo Spica, Eric Cristofalo, Zijian Wang, Eduardo Montijano, and Mac Schwager. A real-time game theoretic planner for autonomous two-player drone racing. *IEEE Transactions on Robotics*, 36(5), 2020. ISSN 19410468. doi: 10.1109/TRO.2020.2994881.

Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. TrafficSim: Learning to Simulate Realistic Multi-Agent Behaviors. *Conference on Computer Vision and Pattern Recognition*, 2021.

Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. In *IJCAI International Joint Conference on Artificial Intelligence*, volume 2019-August, 2019. doi: 10.24963/ijcai.2019/882.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 2017-December, 2017.

Mingyu Wang, Zijian Wang, John Talbot, J. Christian Gerdes, and Mac Schwager. Game-Theoretic Planning for Self-Driving Cars in Multivehicle Competitive Scenarios. *IEEE Transactions on Robotics*, 37(4), 2021. ISSN 19410468. doi: 10.1109/TRO.2020.3047521.

Grady Williams, Brian Goldfain, Paul Drews, James M. Rehg, and Evangelos A. Theodorou. Best Response Model Predictive Control for Agile Interactions between Autonomous Ground Vehicles. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2403–2410, 2018. ISSN 10504729. doi: 10.1109/ICRA.2018.8462831.

Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:8652–8661, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00886.

Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-End Urban Driving by Imitating a Reinforcement Learning Coach. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 8 2021.