Appendix: Energy-based Potential Games for Joint Motion Forecasting and Control

Anonymous Author(s)
Affiliation
Address
`email`

A List of Abbreviations

Abbreviation	**Description**
App.	Appendix
ADE	average displacement error
CDE	conditional density estimation
CV	Constant Velocity
EBM	Energy-based Model
EPOL	Energy-based Potential Game Layer
EPO	Energy-based Potential Game
Eq.	Equation
FDE	final displacement error
Fig.	Figure
GB	gradient-based
HiVT-M	Hierarchical Vector Transformer Modified
IFT	implicit function theorem
LB	learning-based
LSTM	long short-term memory
MLE	maximum likelihood estimation
MLP	multilayer perceptron
MB	model-based
NC	noise contrastive
NE	Nash equilibrium
NN	neural network
NLL	negative log likelihood
Nonlin.	nonlinear
Num.	Number
OCP	optimal control problem
OLNE	open-loop Nash equilibrium
OR	overlap rate
PDG	potential differential game
RPI	robot pedestrian interaction
SADE	scene average displacement error
SB	sampling-based
Sec.	Section
Tab.	Table
SDV	self-driving vehicle
SADE	scene average displacement error
SFDE	scene final displacement error
SOTA	state-of-the-art
UN	unrolling
VIBES	Vectorized Interaction-based Scene Prediction
V-LSTM	Vector-LSTM

B Extended Related Work

Data-driven Motion Forecasting. Deep learning motion forecasting approaches use different observation inputs. [1] uses birds-eye-view images, which have a high memory demand and can lead to discretization errors. [2] propose to use a vectorized environment representation instead, and [3] uses raw-sensor data. Many approaches utilize an encoder-decoder structure with convolutional neural networks [4], transformers [5], or graph neural networks [6] to model multi-agent interactions. In addition to deterministic models [5], various generative models, such as Generative Adversarial Networks (GANs) [7] and Conditional Variational Autoencoder formulations [8], as well as Diffusion Models [9], are used to produce multi-modal predictions. Predicting goals in hierarchical approaches like [10], can further increase the predictive performance using domain knowledge of the map information. Motion forecasting models can also be conditioned on the control [11] or future trajectory [8] of one agent. However, these conditional forecasts might lead to overly confident anticipation of how that agent may influence the predicted agents [12]. To include domain knowledge such as system dynamics into the learning process, it is also common practice [13, 1, 8] to first forecast the future control values of all agents and then to unroll a dynamics model to produce the future states.

Differentiable Optimization for Motion Planning. Differentiable optimization has also been applied in motion planning for SDVs. [14] and [15] imposes safety-constraints using differentiable control barrier functions or gradient-based optimization techniques in static environments. [16] and [17] couple a differentiable single-agent motion planning module with learning-based motion forecasting modules. In contrast, our work performs multi-agent joint optimizations in parallel, derived from a game-theoretic potential game formulation. Game-theoretic formulations can overcome overly conservative behavior when used for closed-loop control [18].

C Theorems

This section provides the full theorem of [19]:

Theorem C.1. *For a differential game $\Gamma_{\mathbf{x}_0}^T := \left(T, \{\mathbf{u}_i\}_{i=1}^N, \{C_i\}_{i=1}^N, f\right)$, if for each agent i, the running and terminal costs have the following structure $L_i(\mathbf{x}(t), \mathbf{u}(t), t) = p(\mathbf{x}(t), \mathbf{u}(t), t) + c_i\left(\mathbf{x}_{-i}(t), \mathbf{u}_{-i}(t), t\right)$ and*

$$S_i(\mathbf{x}(T)) = \bar{s}(\mathbf{x}(T)) + s_i\left(\mathbf{x}_{-i}(T)\right),$$

then, the open-loop control input $\mathbf{u}^* = (\mathbf{u}_1^*, \cdots, \mathbf{u}_N^*)$ that minimizes the following

$$\min_{u(\cdot)} \int_0^T p(\mathbf{x}(t), \mathbf{u}(t), t)dt + \bar{s}(\mathbf{x}(T))$$
$$\text{s.t. } \dot{x}_i(t) = f_i\left(\mathbf{x}_i(t), \mathbf{u}_i(t), t\right),$$

is an OLNE of the differential game $\Gamma_{\mathbf{x}_0}^T$, i.e., $\Gamma_{x_0}^T$ is a potential differential game.

Proof: See [19], with original proof provided by [20].

Here besides the potential functions p and $\bar{s}$, s_i and c_i are terms that are required to not depend on the state or control of agent i.

D Datasets

D.1 RPI

The RPI dataset is a synthetic dataset of simulated mobile robot pedestrian interactions. Multi-modal demonstrations are generated by approximately solving a two-player differential game ($N = 2$) with the iterative linear-quadratic game implementation of [21, 22] based on different start and goal configurations. Fig. 1 provides an illustration for the dataset construction. The robot's initial positions

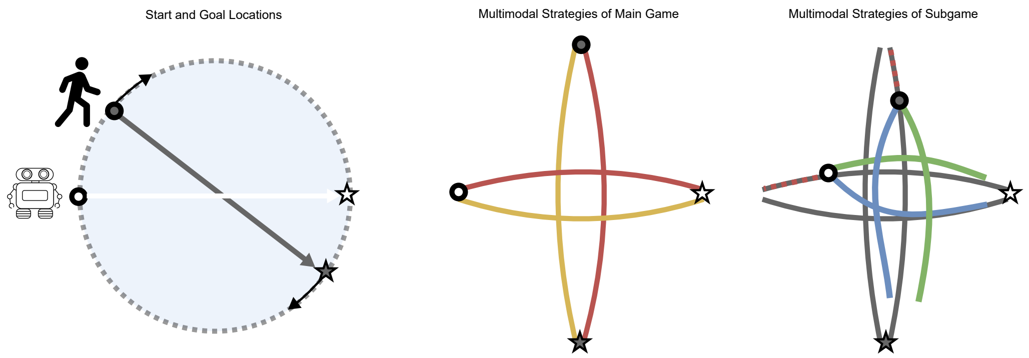
Start and Goal Locations
Multimodal Strategies of Main Game
Multimodal Strategies of Subgame
Figure 1: Dataset construction for RPI dataset. Left: First initial and goal states and game parameters are sampled. Middle: A *main game* is solved multiple times based on the sampled game configuration with subsequent result clustering. That leads to multimodal strategies (red and yellow). The agent moves according to the multimodal strategies of the main game. After a time step Δt, a *sub game* is solved. The results are multimodal strategies (blue and green) of the subgame. The histories (dotted red) and multi-modal strategies of the sub game build a demonstration for training and evaluation.

Figure 2: An exemplary highly-interactive scenario from the exiD dataset.

(white circle) and goal locations (white star) are the same in all solved games. In contrast, the initial state (dark grey circle) and goal location (dark grey stars) of the pedestrian move on a circle, as illustrated on the left graphic in Fig. 1. The agents are tasked to reach a goal location given an initial start state while avoiding collisions and minimizing control efforts. As solving the game once leads to a uni-modal local strategy, this work follows the implementation of Peters et al. [22]. It solves the game for a given initial configuration multiple times based on different sampled strategy initializations. Afterward the resulting strategies are clustered. The clustered strategies represent multi-modal strategies of the *main game*, and they are visualized in red and yellow in Fig. 1. The agents then execute the open-loop controls of the main game's initial strategies. After every time interval $\Delta t = 0.1$, the procedure of game-solving and clustering the results is repeated as long as the agents pass each other. The resulting strategies of the so-called *subgames* are visualized in green and blue on the right of Fig. 1. Based on the history (dotted red line) and the strategies of the subgame (blue and green), we then build a multi-modal demonstration for the dataset. Note that the main game and the corresponding subgames use the same cost function parametrizations, but the agents' preferences for collision avoidance differ between main games.

The resulting dataset is based on 20 main games and their corresponding subgame solutions. Here we draw collision cost parameters from a uniform distribution to enhance demonstration diversity. The resulting dataset contains 60338 samples, whereas we use 47822 ($\sim$80%) for training, 6228 for validation ($\sim$10%), and 6228 ($\sim$10%) for testing. The test set is constructed based on an unseen main game configurations. The goal is to predict $M = 2$ joint futures of $T = 4\,\mathrm{s}$ based on a history of $H = 1.8\,\mathrm{s}$ with a time interval of $\Delta t = 0.1$.

D.2 exiD

The exiD [23] dataset contains $19\,\mathrm{h}$ of real-world highly interactive highway data. Interactions between different types of vehicle classes are rich because the data was recorded by drones flying over seven locations of German highway entries and exits. Highway entries and exits, designed with acceleration and deceleration lanes and high-speed limits, promote interactive lane changes due to high relative speeds between on-ramping and remaining road users. In addition, the most common

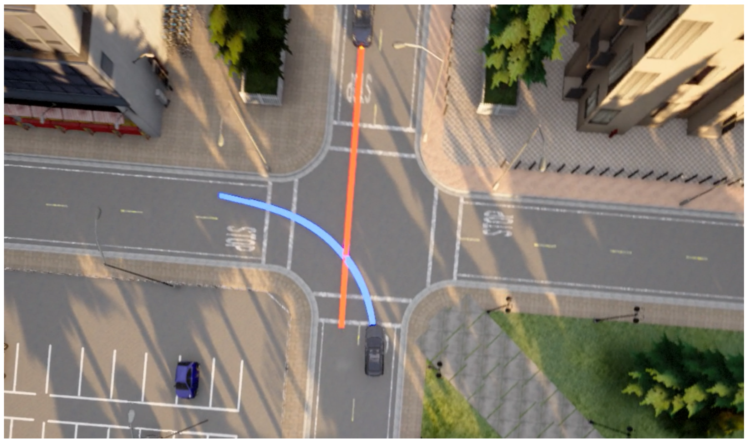

Figure 3: The CARLA left-turning scenario with planned joint trajectories.

cloverleaf interchange in Germany requires simultaneous observation of several other road users and gaps between them for safe entry or exit in a short time frame [23].

To further increase the interactivity, this work extracts scenarios with $N = 4$ agents in which at least one agent performs a lane change. The recordings are then sampled with a frequency of $\Delta t = 0.2\,\mathrm{s}$. The different networks are tasked to predict $M = 5$ joint futures of length $T = 4\,\mathrm{s}$ based on a history of $H = 1.8\,\mathrm{s}$. The resulting dataset contains 290735 samples, whereas we use 206592 ($\sim$72%) for training, 48745 for validation ($\sim$16%), and 35398 ($\sim$12%) for testing. To investigate the generalization capabilities of the different models, the test set contains *unseen scenarios from a different map* (map 0) than the training and validation scenarios. An exemplary scenario is visualized in Fig. 2.

D.3 CARLA

The experiment in the CARLA simulator (Version 9.11) use the implementation of [24] to construct interactive scenarios, whereas the SDV is tasked to perform an unprotected-left turn with another vehicle approaching the intersection ($N = 2$) as visualized in Fig. 3. To generate multi-modal demonstrations for training, the agents follow hand-crafted policies, to generate different outcomes. The SDV first decides whether to enter the intersection. The other vehicle decides to yield or allow the SDV to pass. Subsequently, the SDV re-evaluates the initial maneuver. This results in different interaction outcomes leading to multi-modal demonstrations.

This work uses four different intersections in Town 04. The approach is tasked to predict $M = 2$ joint futures of length $T = 6\,\mathrm{s}$ based on a history of $H = 1.8\,\mathrm{s}$. We only use five episodes of different interaction outcomes for training. In contrast to the exiD and RPI experiments, we use an agent-centric coordinate system with the SDV as the origin. To increase the training dataset size we perform the following data augmentations: 1) We add additional samples based on the original samples where all observations (history and map) are randomly rotated with an rotation angle drawn from a uniform distribution $\mathcal{U}[-\pi/8, \pi/8]$. In these samples we 2) add Gaussian noise $\mathcal{N}(0, 0.02)$ to the 2-D positions in the histories. Note that this augmentations are only performed for samples of the training dataset. The resulting dataset contains 2959 samples, whereas we use 1836 ($\sim$62%) for training, 361 ($\sim$12%) for validation and 762 ($\sim$26%) for testing. The training (intersection 1), validation (intersection 2) and test set (intersection 3 and 4) all differ in terms of the used intersection. During closed-loop control we evaluate on intersection 3.

In the closed-loop control experiment, the SDV follows the procedure described in Section 4.3 of the main paper to predict the SDV strategy $\mathbf{u}_{i=0}^{m^*}$ and the corresponding state trajectory $\mathbf{x}_{i=0}^{m^*}$. Two PID-controller are used for trajectory tracking. They compute a steering angle, braking or throttle signals to control the SDV in the simulator.

E Implementation Details

This section provides additional information for the used observation encoding backbones and the game parameter decoders. We further provide details for the used dynamics and energy features.

E.1 Network Architectures (Backbones and Baselines)

Lane Encoder. In all experiments, the lane encoders ϕ^{lane} of all backbones use a PointNet [25] like architecture as [2] with three layers and a width of 64. The polylines are constructed based on vectors that contain a 2-D start and 2-D goal position in a fixed-global coordinate system. Agent polylines also include time step information and are processed with different encoders depending on the used backbone.

Agent History Encoder. The V-LSTM (Vector-LSTM) [26] and VIBES (Vectorized Interaction-based Scene Prediction) backbones use an LSTM [27] for agent history encoding with depth three and width 64. Our modified HiVT-M (Hierarchical Vector Transformer Modified) [28] implementation uses a transformer [29] for the encoding of each agent individually. Note that this contrasts with the original implementation, where the encoding transformer already models local agent-to-agent and agent-to-lane interactions. We account for that in a modified global interaction graph as listed below. The transformer has a depth of three and a width of 64.

Global Interaction. The V-LSTM backbones update the polyline features in the global interaction graph with a single layer of attention [29] as described by [2]. The HiVT-M and VIBES models use a two-stage attention mechanism. First, one layer of attention between the map and agent polyline features, and afterwards a layer of attention between all updated agents features are applied. The global interaction graph has a width of 128.

Game Parameter and Initial Strategy Decoder. The agent weight, goal, and initial strategy decoders are implemented by a 3-layer MLP with a width of 64.

Goal Decoder. The goal decoder follows [10]. It takes as input the concatenation of an agent feature $\mathbf{z}_i$ and $G = 60$ possible goal points, denoted by $\mathbf{z}^{\text{goal}}$. The goal points are extracted from the centerlines of the current and neighboring lanes. If there exists no neighboring lane, we take the lane boundaries. The decoder ϕ^{goal} then predicts the logits of a categorical distribution per agent $\mathbf{l}_i^{\text{goal}} = \phi^{\text{goal}}(\mathbf{z}^{\text{goal}})$. During training and evaluation, the method takes the M most-likely goals $\mathbf{G}_i$ for all modes of a agent i. Probabilities for the goals per agent are computed by $\mathbf{PR}_i^{\text{goal}} = \text{softmax}(\mathbf{l}_i^{\text{goal}})$. The prediction of goals is made in parallel for all agents.

Scene Probability Decoder. The scene probability decoder also uses a 2-layer MLP with width $16 \times M$ and predicts logits $\mathbf{l}^{\text{prob}}$ for the M scene modes. The scene probabilities are derived by applying the softmax operations $\mathbf{PR} = \text{softmax}(\mathbf{l}^{\text{prob}})$.

The goal, agent weight and scene probability decoder use batch normalization. The interaction weight decoder, initial strategy decoder, and transformer agent encoder use layer normalization.

In the CARLA experiment we scaled the width of all networks by half to mitigate overfitting. We also experimented with downscaling the network with by a factor of four, but saw no increase in performance for the baselines and our method.

E.2 Dynamics

The discrete-time dynamically-extended unicycle dynamics [30, Chapter 13] are given by:

$$\begin{aligned}
x_{k+1} &= x_k + v_k \cos(\theta_t) \Delta t \\
y_{k+1} &= y_k + v_k \sin(\theta_t) \Delta t \\
v_{k+1} &= v_k + a_k \Delta t \\
\theta_{k+1} &= \theta_k + \omega_k \Delta t
\end{aligned} \tag{1}$$

x_k and y_k denote a 2-D position and θ_k the heading. In exiD and RPI the origin of the system is fixed. In CARLA the origin is the SDV, whereas the x axis is aligned with the SDV. v_k is the velocity, a_k the acceleration, ω_k the turnrate, and Δt a time interval. Hence, $n_x = 4 \times N$ and $n_u = 2 \times N$.

E.3 Energy Features and Optimization

Energy Features. The energy function in the RPI experiment uses the following agent-dependent features: $c(\cdot) = [c_{\text{goal}}, c_{\text{vel}}, c_{\text{acc}}, c_{\text{velb}}, c_{\text{accb}}, c_{\text{turnr}}, c_{\text{accb}}, c_{\text{turnrb}}]$. In the RPI experiments, the goal is given and not predicted. The agent-dependent energy features in the exiD experiments are given by $c(\cdot) = [c_{\text{goal}}, c_{\text{lane}}, c_{\text{vref}}, c_{\text{vel}}, c_{\text{acc}}, c_{\text{jerk}}, c_{\text{steer}}, c_{\text{turnr}}, c_{\text{turnacc}}]$. The agent-dependent energy features in the CARLA experiments are given by $c(\cdot) = [c_{\text{lane}}, c_{\text{vref}}, c_{\text{vel}}, c_{\text{acc}}, c_{\text{jerk}}, c_{\text{steer}}, c_{\text{turnr}}, c_{\text{turnacc}}]$. c_{goal} is a terminal cost penalizing the position difference of the last state to the predicted goal. c_{lane} minimizes the distance of the state trajectory to the reference lane to which the predicted goal point belongs. Note that different goal points can be predicted for the modes and as a result different lanes can be selected to better model multi-modality. c_{vref} is the difference between the predicted and and map-specific velocity limit. The other terms are running cost, evaluated for all timesteps and penalize high velocities (c_{vel}), accelerations (c_{acc}), jerks (c_{jerk}), as well as turn rates (c_{turnr}) and turn accelerations (c_{turnacc}). An index b marks a soft constraint implemented as a quadratic penalty, active when the bound is violated. Hence an inequality constraint $g(z) \leq 0$ with optimization variable z is implemented by a feature $\max(0, g(z))$. The interaction feature $d(\cdot)$ is also implemented as such a quadratic penalty. We evaluate the collision avoidance features at every discrete time step in the RPI experiments. In all experiments, agent geometries are approximated by circles of radius r_i, which is accurate for the mobile robot and pedestrian but an over-approximation for vehicles (CARLA, exiD) and especially for trucks in the highway exiD environment, where we use $r_i = L/2$. L is the length of a vehicle. Hence, we evaluate collision avoidance every fifth timestep in the exiD experiments. Future work could also use more accurate vehicle approximations (e.g., multiple circles [31]) to further evaluate collision avoidance at every time step to increase the predictive performance at a higher runtime and memory cost. In the RPI experiments, we set $r_i = 0.25\,\text{m}$.

Optimization. As the approach already predicts accurate initial strategies $\mathbf{U}^{\text{init}}$, our experiments only required a few optimization steps. Concretely, the results of Tab. 2 and 3 in the main paper are obtained with $s = 2$ optimization steps, rendering our approach real-time capable (see Fig. 9). Note while the approach also works, with a higher number of optimization steps (see Fig. 8), our experiments showed that fewer optimization steps lead to similiar results, with decreased runtime and memory requirements due to the predicted initialization. Both experiments use a stepsize of $\alpha = 0.3$. The experiments use a damping factor of $dp = 10$ in the Levenberg-Marquardt solver [32]. In the CARLA experiment we use $s = 20$. Especially in the low sample regime, a higher number of optimization steps is beneficial due to inductive bias from the game-theoretic optimization as the initialization performance is decreased, as also later shown in Tab. 1.

E.4 Training Details

Loss Functions. The imitation loss in our experiments is the minSADE [6, 33] given by:

$$\mathcal{L}^{\text{imit}} = \min_{m=1}^{M} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i^m - \mathbf{x}_{\text{GT}}\|^2 \tag{2}$$

It first calculates the average over all distances between agent trajectories $\mathbf{x}_i^m$ from agent i and mode m and the ground truth $\mathbf{x}_{\text{GT}}$. Then the minimum operator is applied to afterwards backpropagate the difference of the joint scene, which is closest to the ground truth. The second loss term $\mathcal{L}^{\text{goal}}$ computes the cross entropy (CE) for the goal locations averaged over all agents

$$\mathcal{L}^{\text{goal}} = \frac{1}{N} \sum_{i=1}^{N} \text{CE}\left(\mathbf{PR}_i^{\text{goal}}, \mathbf{g}_i^*\right), \tag{3}$$

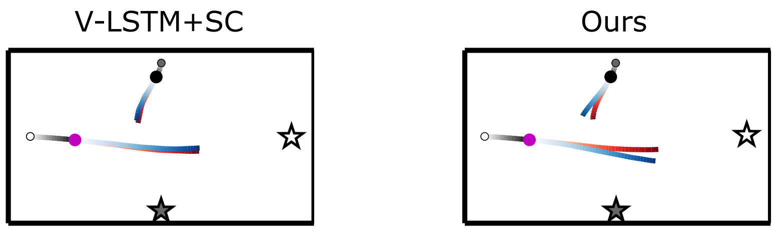

Figure 4: Qualitative comparison of the multi-modal ($M = 2$) joint predictions in the RPI environment. The start and end point of the pink agent are located on a circles with a radius of 3m . The start and endpoint of the black agent are visualized with a grey circle and star. The different modes are visualized in red and blue color.

whereas $\mathbf{g}_i^*$ is the target closest to the ground truth goal location. Lastly, $\mathcal{L}^{\text{prob}}$ computes the cross entropy for the joint futures

$$\mathcal{L}^{\text{prob}} = \text{CE}\left(\mathbf{PR}, \mathbf{x}^*\right), \tag{4}$$

whereas $\mathbf{x}^*$ is the joint prediction target closest to the ground truth joint future, estimated with the minSADE. We empirically set $\lambda_1 = 1, \lambda_2 = 0.1, \lambda_3 = 0.1$ in the multi-task loss described in the main paper.

In the RPI and exiD experiments, all approaches are trained with batch size 32, using the Adam optimizer [34]. Our models in the RPI and exiD environments use a learning rate of 0.00005 across all backbones. Note that the evaluation favors the baselines, as we performed grid searches for their learning rates, whereas our approach uses the same learning rate across all backbones (exiD). In CARLA we empirically set the batch size to 16 and the learning rate to 0.0005 for our method.

F Additional Experiments

F.1 Qualitative Results

This section provides extended qualitative results.

RPI. Fig. 4 visualizes an exemplary qualitative result of the RPI experiments. Both modes collapsed when using the V-LSTM+SC baseline (explicit strategy). In contrast, this work's implicit approach better models the multi-modality present in the demonstration. Since the dataset contains solutions of games solved with different collision-weight configurations, it can be seen that our proposed method accurately differentiates between different weightings of collisions. This finding aligns with these of [35], which discovered that implicit models could better represent the multi-modality of demonstrations.

exiD. Fig. 5 visualizes multi-modal predictions in a highly interactive scenario, where one car (green) and one truck (yellow) merge onto the highway. The green car performs a double-lane change. Note how our model in mode three accurately predicts the future scene evolution and also outputs reasonable alternative futures. For example, in mode one, the green car performs a single lane change, whereas the blue and red cars are also predicted to change lanes. Another multi-modal prediction is visualized in Fig. 6. Observe again how the ground truth is accurately predicted in this interactive scenario (mode 5), whereas, for example, also other plausible futures are generated. For instance, the yellow vehicle stays longer on the acceleration lane in mode one, whereas in mode three, the green vehicle performs a lane change.

CARLA. Fig 7 visualizes another qualitative joint prediction results in the CARLA environment. Observe how our method again predicts two reasonable joint futures, whereas the correct one (mode 1) has higher probability. In mode 1 the SDV (blue) goes first. That behavior is also observed when inspecting the feature weights. For example the weight w_{vref} of the SDV (blue) is higher in mode 1 than in mode 2, inducing a acceleration in mode 1. The same holds for the red agent in mode 2.

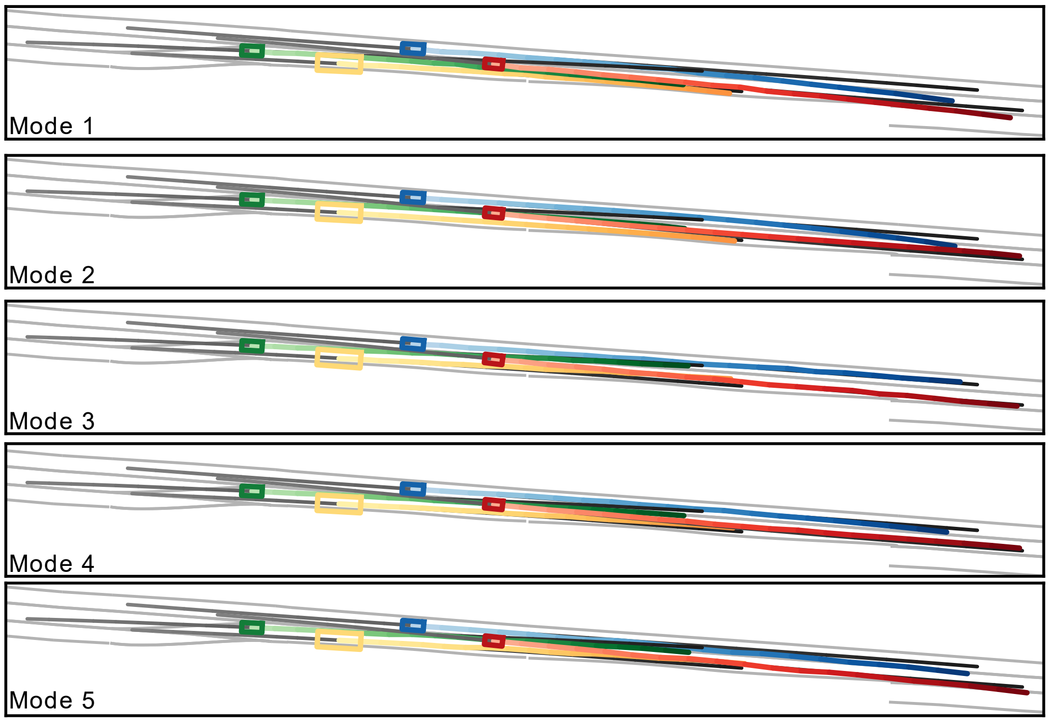

Figure 5: Multi-modal predictions in an interactive scenario, where the green and yellow perform on-ramp merges. The agent trajectories are visualized in different colors, whereas the color saturation increases the number of predicted steps. The ground truth (history and future) is shown with colors from dark grey to black and the map in light grey.

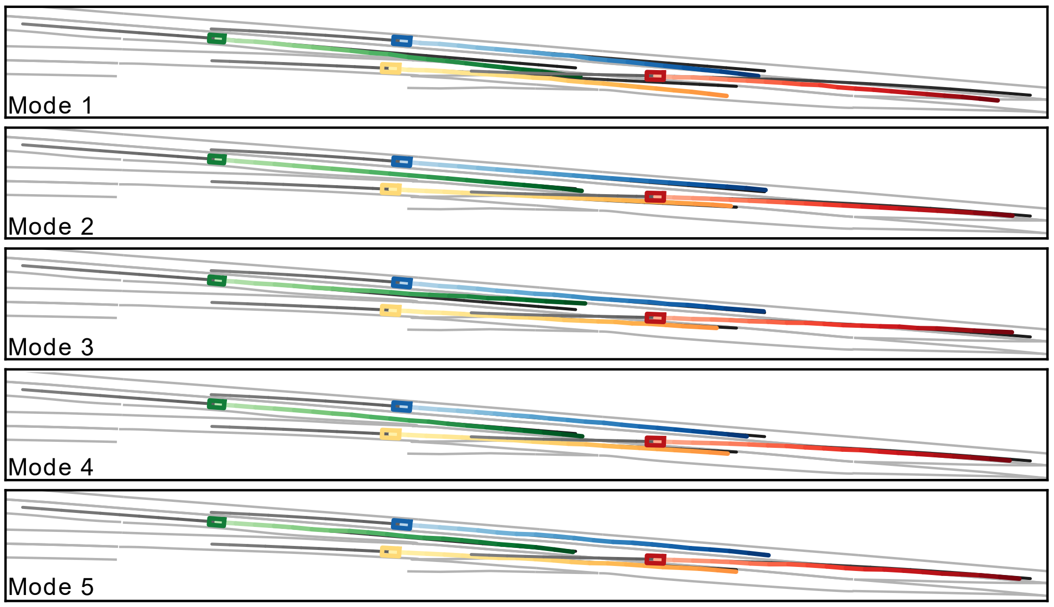

Figure 6: Multi-modal predictions in a interactive scenario, where the yellow and red agents perform lane changes. The agent trajectories are visualized in different colors, whereas the color saturation increases the number of predicted steps. The ground truth (history and future) is shown with colors from dark grey to black and the map in light grey.

Remember from the main paper, the weights are normalized with respect to the maximum weight of the individual feature in the sequence. *Hence, weights of the same feature are comparable between the two modes, but weights from different features are not comparable.*

F.2 Discussion of Interpretability

Following the definition of [36], interpretability in the context of SDV is achieved, among other things, by the input, output, and intermediate representations. While the input (historical trajectories and map information) and the output (future trajectories) are already human-interpretable, our approach can also output interpretable intermediate representations in the form of feature weights. For instance, visualization of these features can provide a consumer in the car another instance of insights about what the SDV will do in the future. For example, assume a feature that minimizes the distance to a stopping line. A high weight could indicate that the vehicle will stop at the line. Moreover, engineers could use these weights during debugging and algorithmic design. For instance, if a weight converges to always zero during training, it could indicates that the feature is unimportant, and hence, the engineer could discard the feature to reduce algorithmic complexity. Lastly, the feature weights could be used to design safety layers. For example, consider the scenario again in Fig 7 and assume another module indicating if a scenario is safety-critical. If this new module now classifies that the scenario is safety-critical, while our approach plans that the SDV should accelerate (e.g., indicated by a high weight w_{vref}), a third module could detect this conflict and overwrite the decision of our approach, to perform a braking maneuver instead.

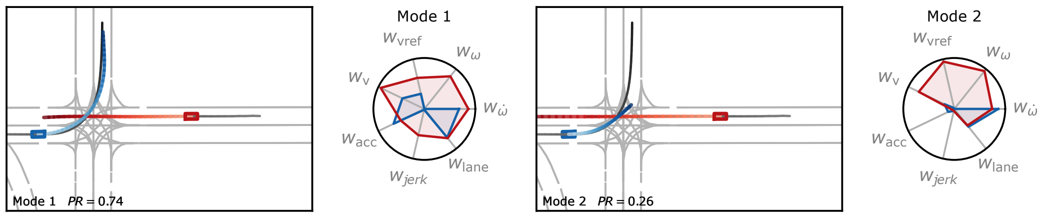

Figure 7: Qualitative joint predictions and feature weights for $M = 2$ modes. The self-dependent weights $\mathbf{W}_i^{\text{own}}$ normalized w.r.t. the maximum weight in the sequence. The ground truth is visualized with colors from dark grey to black and the map in light grey.

F.3 Demo Video

Please refer to the accompanying video for extensive visualization.

F.4 Quantitative Results

CARLA. Tab. 1 illustrates the predictive performance on the CARLA testset. We observed that the strongest baseline, which also uses the scene-consistent loss formulation with control prediction does not produce reasonable predictions, despite performing grid searches for different hyperparameters. Our method outperforms the baseline by a large margin. Nevertheless, although our method demonstrates practicality in this small-sample regime, the significance of the results is comparatively limited, as previously stated in the work of [37]. Especially in automated driving applications one has access to large datasets. However, in other robotics applications such as human-robot manipulator collaborations [38] datasets are fairly limited and we hypothesize that the approach could be beneficial here. We leave this studies for future work.

Runtime. Training and evaluation was performed using an AMD Ryzen 9 5900X and a Nvidia RTX 3090 GPU. Fig. 9 shows the runtime dependency by varying the number of optimization steps S, the modes M and the number of agents N. We observe, that our approach scales well with the number modes as all optimizations are parallelized on the GPU. The runtime increases with higher numbers of steps and more agents as commonly reported in the game-theoretic literature. *Note that*

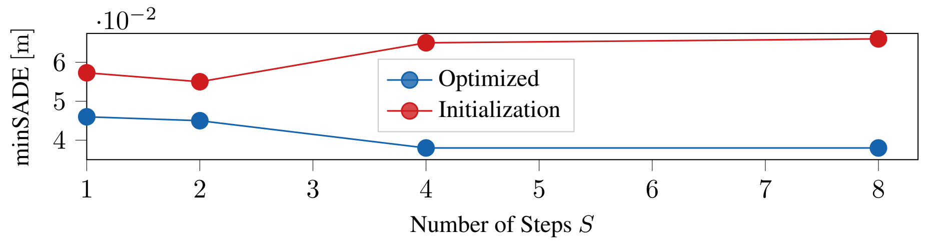

Figure 8: Predictive performance of the initial (red) and optimized strategy (blue) as a function of the number of optimization steps on the RPI validation dataset.

our multi-modal implementation scales better than common game-theoretic uni-modal solvers from the literature [39, Figure 4] for a higher number of agents.

Table 1: Predictive performance of different methods on the Carla test dataset. The metrics and formatting are the same as in Tab. 2 in the main paper, but $M = 2$.

	Marginal ↓		Joint ↓	
Method	ADE	FDE	SADE	SFDE
V-LSTM + SC	6.11	12.37	6.21	13.34
V-LSTM + Ours	**1.74**	**3.28**	**1.83**	**3.43**

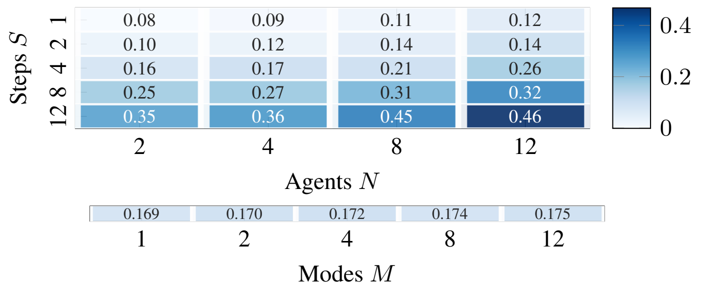

Figure 9: Runtime in [s] for different numbers of agents and optimization steps averaged over 100 exiD samples with $M = 4f4$. The experiments for the number of modes use $S = 4$ and $N = 4$.

F.5 Ablation Studies

Ablating the Number of Optimization Steps The experiments revealed that another influential hyperparameter is the number of steps S during optimization. Fig 8 visualizes the impact on the minSADE. Observe how the approach gets reasonable small metrics with all configurations and hence could be used with different numbers of steps. However, while the distance between the closest optimized joint future and the GT gets smaller with increasing optimization steps, the initialization gets slightly pushed away from the GT. Hence, with more steps, the approach gets less dependent on the initialization. [17] observes an similar effect for their differentiable single-agent optimization approach.

G Additional Limitations

In this section we name additional limitations. Our CARLA experiment are limited by the dataset size (see discussion in Sec. F.4) and can be regarded as a proof-of-concept for closed-loop control. Moreover, we observed that in many scenarios, besides the interacting vehicles, other vehicles in the exiD dataset performed a nearly constant velocity movement, which is also verified by the good results of the constant velocity baselines in Tab. 2 of the main paper. Here, future work should evaluate on larger urban datasets and simulators (e.g., [26]). Further, our approach assumes a object-based environment representation, with handcrafted input features (e.g., 2-D position information in agent histories) and low measurement uncertainties. However, raw-sensor data includes important information (e.g., the head movement of a pedestrian), which is relevant for downstream tasks such as motion forecasting and control. As our approach is fully differentiable, future work should explore joint perception and game-theoretic planning approaches. Doing so, would allow to propagate uncertainties trough the whole system architecture, which has proven to be effective in prior work such as [40].

References

[1] H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, J. Schneider, D. Bradley, and N. Djuric. Deep kinematic models for kinematically feasible vehicle trajectory predictions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10563–10569, 2020.

[2] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[3] K. M. K. S. L. Nicholas Rhinehart, Rowan McAllister. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of (ICCV) International Conference on Computer Vision*, pages 2821 – 2830, October 2019.

[4] N. Deo and M. M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1468–1476. Computer Vision Foundation / IEEE Computer Society, 2018.

[5] J. Ngiam, V. Vasudevan, B. Caine, Z. Zhang, H. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, D. J. Weiss, B. Sapp, Z. Chen, and J. Shlens. Scene transformer: A unified architecture for predicting future trajectories of multiple agents. In *The Tenth International Conference on Learning Representations, ICLR 2022, April 25-29, 2022*, 2022.

[6] S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *Computer Vision – ECCV 2020*, pages 624–641, Cham, 2020. Springer International Publishing.

[7] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[8] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 683–700, Cham, 2020. Springer International Publishing.

[9] T. Gu, G. Chen, J. Li, C. Lin, Y. Rao, J. Zhou, and J. Lu. Stochastic trajectory prediction via motion indeterminacy diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17113–17122, June 2022.

[10] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, C. Li, and D. Anguelov. Tnt: Target-driven trajectory prediction. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 895–904. PMLR, 16–18 Nov 2021.

[11] C. Diehl, T. S. Sievernich, M. Krüger, F. Hoffmann, and T. Bertram. Uncertainty-aware model-based offline reinforcement learning for automated driving. *IEEE Robotics and Automation Letters*, 8(2):1167–1174, 2023.

[12] C. Tang, W. Zhan, and M. Tomizuka. Interventional behavior prediction: Avoiding overly confident anticipation in interactive prediction. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11409–11415, 2022.

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

[14] W. Xiao, T.-H. Wang, R. Hasani, M. Chahine, A. Amini, X. Li, and D. Rus. Barriernet: Differentiable control barrier functions for learning of safe robot control. *IEEE Transactions on Robotics*, pages 1–19, 2023.

[15] C. Diehl, J. Adamek, M. Krüger, F. Hoffmann, and T. Bertram. Differentiable constrained imitation learning for robot motion planning and control. art. arXiv:2210.11796, 2022.

[16] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone. Diffstack: A differentiable and modular control stack for autonomous vehicles. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 2170–2180. PMLR, 14–18 Dec 2023.

[17] Z. Huang, H. Liu, J. Wu, and C. Lv. Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving. art. arXiv:2207.10422, 2023.

[18] X. Liu, L. Peters, and J. Alonso-Mora. Learning to play trajectory games against opponents with unknown objectives. art. arXiv:2211.13779, 2023.

[19] T. Kavuncu, A. Yaraneri, and N. Mehr. Potential ilqr: A potential-minimizing controller for planning multi-agent interactive trajectories. In *Robotics: Science and Systems XVII*, 07 2021.

[20] A. Fonseca-Morales and O. Hernández-Lerma. Potential Differential Games. *Dynamic Games and Applications*, 8(2):254–279, June 2018.

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

[22] L. Peters, D. Fridovich-Keil, C. J. Tomlin, and Z. N. Sunberg. Inference-based strategy alignment for general-sum differential games. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, page 1037–1045. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

[23] T. Moers, L. Vater, R. Krajewski, J. Bock, A. Zlocki, and L. Eckstein. The exid dataset: A real-world trajectory dataset of highly interactive highway scenarios in germany. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 958–964, 2022.

[24] N. Rhinehart, J. He, C. Packer, M. A. Wright, R. McAllister, J. E. Gonzalez, and S. Levine. Contingencies from observations: Tractable contingency planning with learned behavior models. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13663–13669, 2021.

[25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[26] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion

dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9710–9719, October 2021.

[27] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[28] Z. Zhou, L. Ye, J. Wang, K. Wu, and K. Lu. Hivt: Hierarchical vector transformer for multi-agent motion prediction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8813–8823, 2022.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[30] S. M. Lavalle. *Planning Algorithms*. Cambridge University Press, 2006.

[31] J. Ziegler and C. Stiller. Fast collision checking for intelligent vehicle motion planning. In *2010 IEEE Intelligent Vehicles Symposium*, pages 518 – 522, 07 2010.

[32] L. Pineda, T. Fan, M. Monge, S. Venkataraman, P. Sodhi, R. T. Q. Chen, J. Ortiz, D. DeTone, A. Wang, S. Anderson, J. Dong, B. Amos, and M. Mukadam. Theseus: A library for differentiable nonlinear optimization. In *Advances in Neural Information Processing Systems*, volume 35, pages 3801–3818. Curran Associates, Inc., 2022.

[33] E. Weng, H. Hoshino, D. Ramanan, and K. Kitani. Joint metrics matter: A better standard for trajectory forecasting. art. arXiv:2305.06292, 2023.

[34] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[35] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 158–168. PMLR, 08–11 Nov 2022.

[36] É. Zablocki et al. Explainability of vision-based autonomous driving systems: Review and challenges. *Interntional Journal Computer Vision*, 2022.

[37] P. Geiger and C.-N. Straehle. Learning game-theoretic models of multiagent trajectories using implicit layers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):4950–4958, May 2021.

[38] M. Krämer, C. Rösmann, F. Hoffmann, and T. Bertram. Model predictive control of a collaborative manipulator considering dynamic obstacles. *Optimal Control Applications and Methods*, 41(4):1211–1232, 2020.

[39] S. Le Cleac'h, M. Schwager, and Z. Manchester. Algames: a fast augmented lagrangian solver for constrained dynamic games. *Autonomous Robots*, 46, 01 2022.

[40] W. Zeng, S. Wang, R. Liao, Y. Chen, B. Yang, and R. Urtasun. Dsdnet: Deep structured self-driving network. In *Computer Vision – ECCV 2020*, pages 156–172, Cham, 2020. Springer International Publishing.