Interaction-Merged Motion Planning: Effectively Leveraging Diverse Motion Datasets for Robust Planning

Supplementary Material

A. Datasets

1. **Human-Human Interaction Dataset: ETH-UCY**
   The ETH-UCY dataset [7, 8] consists of five sub-datasets: ETH, Hotel, Univ, Zara1, and Zara2, each with distinct pedestrian densities and scene characteristics. We alternately select 4 out of 5 scenes to form the training and validation datasets, and train a separate model for each configuration. Additionally, since this dataset does not include a designated robot, we construct the dataset by alternately assuming each human agent in the scene as the robot.
2. **RL algorithm-based Robot Dataset: CrowdNav**
   The CrowdNav dataset [2] is a simulation-based dataset designed to enable collision-free navigation in crowded environments. To model human interactions, the dataset first generates human movements by employing the ORCA algorithm, allowing agents to reach their destinations while avoiding collisions. Subsequently, reinforcement learning is used to generate the robot's trajectory, ensuring it navigates without colliding with humans, thereby forming the complete dataset. Following the dataset composition approach from the previous study [6], we split the dataset into a 50:50 ratio for the training and validation sets.
3. **Human-Robot Interaction Dataset: THOR and SIT**
   The THOR dataset [9] is collected in an indoor environment with real humans and a robot. The data was gathered in an indoor space measuring 8.4 × 18.8 m, with various fixed obstacles placed throughout. In this setting, real humans navigate toward one of five designated destinations while avoiding the moving robot. Meanwhile, the robot follows a predetermined path to patrol the indoor space, regardless of nearby human presence and without considering human interaction. Unlike the CrowdNav dataset, where the robot takes human interaction into account, the THOR dataset captures scenarios where humans adjust their movement in response to the robot.
   The SIT dataset [1] contains indoor and outdoor scenes where both robots and humans move while considering their interactions. Data was collected from a total of 10 different scenes, each with varying crowd densities. Since the test set does not provide ground truth position information for surrounding agents, we conduct evaluations using the validation set.

B. Backbone Models

We conduct experiments using three different backbone models. Unlike the GameTheoretic model, DIPP and DTPP are designed for vehicle datasets, requiring some modifications to adapt them to the given robot dataset. (1) **GameTheoretic** [6] trains the forecaster to generate risky forecasts for the ego agent, encouraging the planner to produce safer plans. To extract a valid task vector from a consistent initial state across different datasets, we omit the step of training the forecaster to be risky. Instead, we ensure that the planner considers collisions during its training process. (2) **DIPP** [4] integrates the prediction module with the planning module, jointly training the prediction model to improve planning performance. Since the planner is trained after the predictor, the task vector is extracted from epochs after the planner's training begins. Additionally, weights are extracted at once from the unified module combining the planner and predictor. (3) **DTPP** [5] constructs a tree-structured planner and selects the optimal plan based on cost, achieving higher performance than single-step planning methods that directly generate plans. In this study, we use the model as is, incorporating it as a baseline.

The GameTheoretic uses both the past trajectory of surrounding agents, $\mathcal{X}_{surr}^{-T_{obs}:0}$, and the future predicted trajectory $\hat{\mathcal{Y}}_{surr}^{1:T_{fut}}$ generated by the forecaster to embed surrounding agents' feature h_{surr}. This feature is then used as the input to the interaction layer ψ_{inter}. In contrast, for DIPP and DTPP, h_{surr} is constructed solely based on the past trajectory of surrounding agents, $\mathcal{X}_{surr}^{-T_{obs}:0}$, and used as the input to the interaction layer. Instead, the loss for both the plan and prediction generated by the independent decoder from the transformer is provided to simultaneously train the forecaster and planner in a meaningful way.

C. Evaluation Metrics

1. **Average Displacement Error (ADE)**
   ADE is a metric for evaluating effectiveness by assessing how similar the generated ego agent's future plan is to the dataset's ground truth trajectory. ADE computes the L2 distance between every time step of the plan and the corresponding GT point, and then averages these distances. The detailed formula for the ADE metric used is as follows;

$$\text{ADE} = \frac{1}{NF} \sum_{i=1}^{N} \sum_{j=1}^{F} \|\mathbf{x}_i^j - \mathbf{g}_i^j\| \tag{1}$$

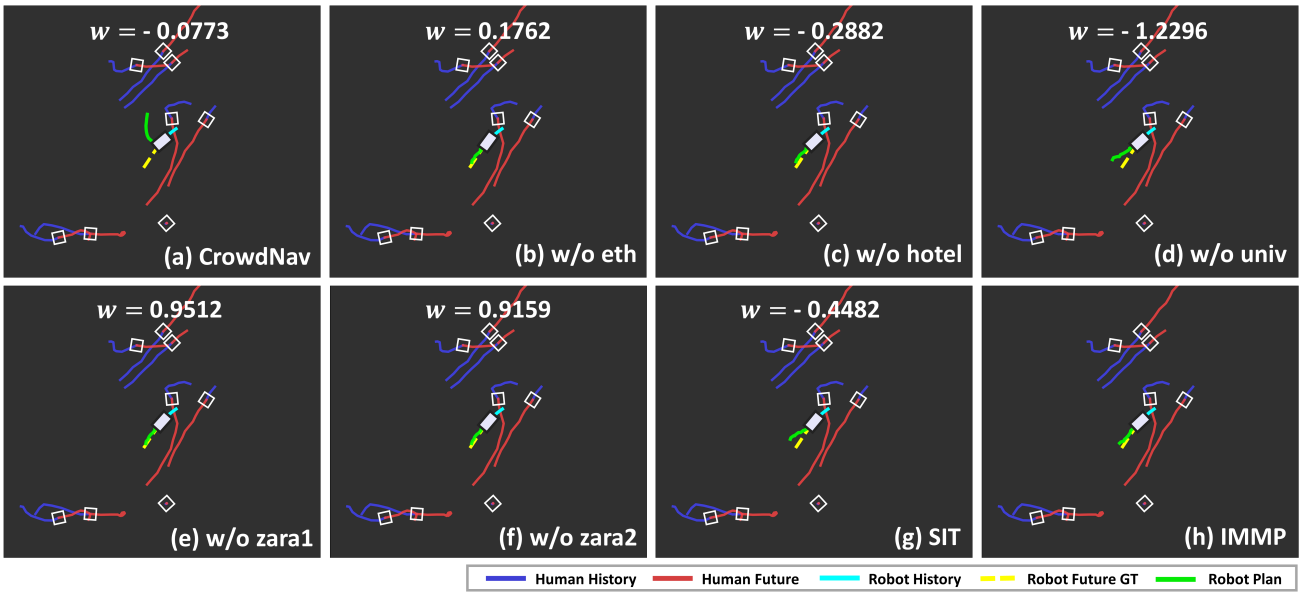

Figure 1. Qualitative results on the THOR dataset with GameTheoretic planning model. From (a) to (g) represent the inference results for each model trained on source domains. The values w indicate the average of weights of task vectors for each source domain in the IMMP planner.

where N is the number of samples, F is the number of future timesteps, $\mathbf{x}$ is the generated plan, and $\mathbf{g}$ is the ground truth plan.

2. **Collision Rate (CR)**
   Collision Rate is an important metric for evaluating safety in Motion Planning. It considers a collision to occur when the distance between certain waypoints in the generated plan and the ground truth plan is below a specified threshold. Following the approach in GameTheoretic [6], we use a threshold of 0.6. The formula for Collision Rate is as follows:

$$\text{CR} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left(\min_{j=1,\dots,F} \|\mathbf{x}_i^j - \mathbf{g}_i^j\| < \epsilon_c\right) \quad (2)$$

   where ϵ_c is the collision threshold.

3. **Final Displacement Error (FDE)**
   FDE is a metric for evaluating goal success. It calculates the L2 distance between the position at the final time step of the generated ego agent's plan and the destination. The formula for the FDE metric is as follows:

$$\text{FDE} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i^{T_f} - \mathbf{g}_i^{T_f}\| \quad (3)$$

   where T_f is the final timestep.

4. **Miss Rate (MR)**
   Miss rate is also a metric for evaluating goal success, assessing whether the position at the final time step of the generated plan deviates from the destination by more than a specified threshold. We compute the L2 distance between the endpoint of the plan and the destination, and here we use a threshold of 0.5 to determine a miss. The detailed formula for Miss Rate is as follows:

$$\text{MR} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(\|\mathbf{x}_i^{T_f} - \mathbf{g}_i^{T_f}\| > \epsilon_m) \quad (4)$$

   where ϵ_m is the miss threshold.

Table 1. Checkpoint pool $\mathcal{P}$ constructed for each planning model. The checkpoint interval C refers to the extraction interval of intermediate epoch points.

Model	Checkpoint Pool $\mathcal{P}$
GameTheoretic (forecaster)	ADE, C = 30
GameTheoretic (planner)	ADE, CR, FDE, MR, C = 5
DTPP	ADE, CR, FDE, MR, C = 1
DIPP	ADE, CR, FDE, MR, C = 5

D. Implementation Details

1. **Interaction-Conserving Pre-Merging**
   We extract key checkpoints from each of the three trained planning baseline models. Specifically, we select the checkpoints where ADE, CR, FDE, and MR achieve their best values and store them in the checkpoint pool $\mathcal{P}$. The selection of intermediate checkpoints during training is determined through multiple experiments. For GameTheoretic [6], unlike DTPP [5] and DIPP [4], a separate forecaster is used. Therefore, we also extract forecaster checkpoints following a similar methodology

as the planner. The checkpoints used for each planning model are in Tab. 1

2. **Interaction Transfer with Merging**
   Checkpoints in the checkpoint pool $\mathcal{P}$ are separated by module within the planning model. Specifically, we extract weights from the LSTM layers responsible for embedding the trajectories of the ego agent and surrounding agents, denoted as θ_{ego} and θ_{surr}, respectively. Additionally, we obtain θ_{inter} from the transformer model, which embeds interactions between the ego agent and other agents. Other parameters, excluding those related to embedding, are consolidated and stored together. When merging trainable parameters, we update them based on the loss of the target domain in the original planning model. We use Adam as the optimizer and set the learning rates to 1e-3, 1e-2, and 1e-3 for GameTheoretic, DTPP, and DIPP, respectively. For the scheduler, we use ReduceLROnPlateau across all three models.

E. Qualitative results

In Fig.3 of the main paper, the correlation between the merging weights of IMMP and the inference results of models trained on each source domain is shown when the SIT dataset is used as the target domain in the GameTheoretic model. Figure 1 presents the result when the THOR dataset is used as the target domain. In cases Fig. 1 (d) and (g), which show poor inference performance, low weights are assigned, while in cases Fig. 1 (e) and (f), which show good inference performance, higher weights are assigned. This demonstrates that even when the composition of the target dataset changes, IMMP effectively identifies the important source domains.

F. Further Analysis

1. **Need for Metric-wise Checkpoint Collection**
   To demonstrate that metric-wise checkpoints capture distinct characteristics, we evaluated metric-specific checkpoints, selected on the Univ dataset, across two target domains. As shown in Tab. 2, checkpoints from the same source (Univ) but selected by different metrics (e.g., Collision vs. Miss Rate) behave differently across target domains. Notably, the collision-optimized checkpoint performs best on THOR but worst on SIT, suggesting that each metric captures distinct aspects of the source domain.

2. **Hyperparameter Sensitivity: Checkpoint Interval**
   As shown in Tab. 3, when the GameTheoretic model targets the SIT domain, we evaluated performance across different checkpoint intervals C. Our method consistently outperforms the best-performing baseline in Tab. 1 (Ensemble-WTA, ADE: 0.3695), even under various settings of C. This demonstrates that our model requires minimal manual tuning and offers high practical utility.

Table 2. Target evaluation of metric-wise checkpoints from Univ.

Target	Chckpoint	ADE ↓	Col. Rate ↓	FDE ↓	Miss Rate ↓
SIT	ADE / FDE	0.4671	2.54E-04	1.0002	0.8043
	Col. Rate	0.8023	6.08E-04	1.6444	0.9489
	Miss Rate	0.3956	4.29E-04	0.8730	0.7294
THOR	ADE / FDE	0.3692	4.98E-04	0.6602	0.7470
	Col. Rate	0.2723	4.91E-04	0.5543	0.4481
	Miss Rate	0.3674	5.19E-04	0.6541	0.7488

Table 3. Effect of checkpoint interval C on Gametheoretic model performance in the SIT target domain (without finetuning).

C value	ADE ↓	Col. Rate ↓	FDE ↓	Miss Rate ↓
1	0.3234	4.69E-05	0.7471	0.6214
2	0.3219	4.28E-05	0.7424	0.6156
3	0.3220	4.28E-05	0.7433	0.6144
10	0.3243	3.62E-05	0.7548	0.6446
Ours (5)	0.3157	4.28E-05	0.7300	0.5934

3. **Ablation Study on Module Separation**
   As shown in Sec. A, the robot motion datasets differ in ego agent type and interaction mechanisms. Our module separation is designed to reflect these characteristics. To validate its effectiveness, we conduct an ablation comparing the original grouping with variants where two of the three modules are merged. As shown in Tab. 4, the results confirm that our original grouping is more effective.

Table 4. Ablation of Module Separation in DIPP on the SIT target domain (A: Robot, B: Human, C: Interaction)

Grouping	ADE ↓	Col. Rate ↓	FDE ↓	Miss Rate ↓
A + B	0.5666	**1.36E-04**	1.0744	0.8351
B + C	0.5608	2.18E-04	1.0631	0.8217
C + A	0.6108	1.83E-04	1.1462	0.8751
Seperate (origin)	**0.5112**	1.60E-04	**0.9358**	**0.7944**

4. **Correlation Between Domain Similarity and Merging Weights**
   We estimate the similarity between the source and target domains by measuring zero-shot performance, and visualize the correlation between dataset similarity and the merging weights. As shown in Fig. 2, there exists a proportional relationship between the merging weights and the domain similarity, indicating that more similar source domains are assigned higher weights.

5. **Comparison with Other Merging Techniques**
   We compare with recent merging baselines listed in Tab. 5, evaluated on the SIT dataset using the GameTheoretic planning model. Existing methods [3, 10] fail to preserve the hierarchical structure inherent in motion planning, resulting in notable performance degradation. In contrast, IMMP effectively transfers knowledge from

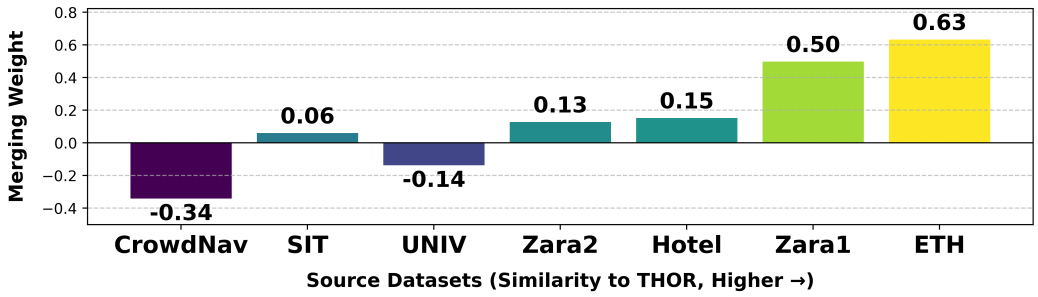

Figure 2. Merging weights with respect to similarity to THOR.

source datasets to the target domain, achieving superior performance.

Table 5. Comparison with recent merging methods on the SIT target domain.

Method	ADE ↓	Col. Rate ↓	FDE ↓	Miss Rate ↓
MuDSC [10]	1.6729	4.21E-03	2.9329	1.0
EMR-Merging [3]	1.5367	2.06E-03	2.8272	0.9941
IMMP	**0.3380**	**5.12E-05**	**0.9580**	**0.6976**

6. Experiments on a Larger Target Domain

We selected SIT and THOR as target domains because they are real-world datasets collected in actual robotic navigation environments, where data collection is relatively challenging. Such environments are likely to serve as realistic target domains in practical applications. We further evaluate IMMP on Zara2 (3× larger than THOR) in Tab. 6, proving robust on large-scale domains.

Table 6. IMMP performance on the Zara2 target domain

Method	ADE ↓	Col. Rate ↓	FDE ↓	Miss Rate ↓	Cost ↓
Domain Generalization [17]	0.5350	0.01802	1.1291	0.7641	X1
Domain Adaptation [16]	0.5378	0.01657	1.1441	0.7561	X1
Target Only	0.4850	0.01663	1.0047	0.7224	X1
Ensemble-WTA [2]	0.4781	**0.01614**	0.9917	0.7078	X7
IMMP + Finetune	**0.4687**	0.01688	**0.9580**	**0.6976**	X1

References

[1] Jong Wook Bae, Jungho Kim, Junyong Yun, Changwon Kang, Jeongseon Choi, Chanhyeok Kim, Junho Lee, Jungwook Choi, and Jun Won Choi. Sit dataset: socially interactive pedestrian trajectory dataset for social navigation robots. *Advances in Neural Information Processing Systems*, 36:24552–24563, 2023. 1

[2] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 international conference on robotics and automation (ICRA)*, pages 6015–6022. IEEE, 2019. 1

[3] Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. Emr-merging: Tuning-free high-performance model merging. *Advances in Neural Information Processing Systems*, 37:122741–122769, 2024. 3, 4

[4] Zhiyu Huang, Haochen Liu, Jingda Wu, and Chen Lv. Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving. *IEEE transactions on neural networks and learning systems*, 2023. 1, 2

[5] Zhiyu Huang, Peter Karkus, Boris Ivanovic, Yuxiao Chen, Marco Pavone, and Chen Lv. Dtpp: Differentiable joint conditional prediction and cost evaluation for tree policy planning in autonomous driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6806–6812. IEEE, 2024. 1, 2

[6] Kushal Kedia, Prithwish Dan, and Sanjiban Choudhury. A game-theoretic framework for joint forecasting and planning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6773–6778. IEEE, 2023. 1, 2

[7] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer graphics forum*, pages 655–664. Wiley Online Library, 2007. 1

[8] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th international conference on computer vision*, pages 261–268. IEEE, 2009. 1

[9] Andrey Rudenko, Tomasz P Kucner, Chittaranjan S Swaminathan, Ravi T Chadalavada, Kai O Arras, and Achim J Lilienthal. Thör: Human-robot navigation data collection and accurate motion trajectories dataset. *IEEE Robotics and Automation Letters*, 5(2):676–682, 2020. 1

[10] Zhengqi Xu, Ke Yuan, Huiqiong Wang, Yong Wang, Mingli Song, and Jie Song. Training-free pretrained model merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5915–5925, 2024. 3, 4