2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Relative Position Matters: Trajectory Prediction and Planning with Polar Representation

Bozhou Zhang[1] Nan Song[1] Bingzhao Gao[2] Li Zhang[1*]

https://github.com/LogosRoboticsGroup/Polaris

***Abstract*— Trajectory prediction and planning in autonomous driving are highly challenging due to the complexity of predicting surrounding agents' movements and planning the ego agent's actions in dynamic environments. Existing methods encode map and agent positions and decode future trajectories in Cartesian coordinates. However, modeling the relationships between the ego vehicle and surrounding traffic elements in Cartesian space can be suboptimal, as it does not naturally capture the varying influence of different elements based on their relative distances and directions. To address this limitation, we adopt the Polar coordinate system, where positions are represented by radius and angle. This representation provides a more intuitive and effective way to *model spatial changes and relative relationships*, especially in terms of distance and directional influence. Based on this insight, we propose Polaris, a novel method that operates entirely in Polar coordinates, distinguishing itself from conventional Cartesian-based approaches. By leveraging the Polar representation, this method explicitly models distance and direction variations and captures relative relationships through dedicated encoding and refinement modules, enabling more structured and spatially aware trajectory prediction and planning. Extensive experiments on the challenging prediction (Argoverse 2) and planning benchmarks (nuPlan) demonstrate that Polaris achieves state-of-the-art performance. We will release our code.**

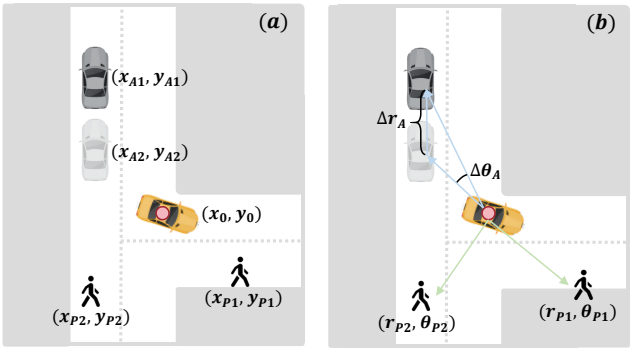

Fig. 1: **Comparison between our Polaris and previous methods.** As illustrated in **(a)**, previous methods encode agents and lanes to predict future trajectories in Cartesian coordinates. In contrast, our Polaris, shown in **(b)**, operates entirely in the polar coordinate system, *explicitly representing radius and angle to intuitively model relative distances, directions, and their variations*. This leads to more accurate trajectory prediction and planning.

I. Introduction

Trajectory prediction and planning, which involve predicting the future trajectories of surrounding vehicles and pedestrians and planning the ego agent's future trajectory using HD maps and historical trajectory data, are crucial for autonomous driving [1]. In recent years, significant progress has been made in this area. Some studies focus on improving the encoding of map and historical trajectory data to enhance scene context features [2], [3], while others explore better methods for decoding future trajectories [4], [5], [6], [7].

Existing methods [4], [8], [5], [7] follow a paradigm based on the Cartesian coordinate system. In these approaches, the scene context is encoded in Cartesian coordinates, and future trajectory positions are regressed accordingly, as illustrated in Figure 1 (a). These methods use (x, y) positions to model the relationships between the ego vehicle and surrounding traffic elements. The varying importance of these elements—determined by their relative distances and directions to the ego vehicle—is learned implicitly. Some approaches attempt to capture such relationships using $(\Delta x, \Delta y)$; however, this form still does not directly encode relative distance and direction, instead relying on the model to infer them. Such implicit modeling may lead to suboptimal performance. For example, as shown in Figure 1, pedestrian 1 (on the right) is directly ahead of the vehicle, while pedestrian 2 (on the left) is positioned to the side. Pedestrian 1 should have a greater influence on the ego vehicle's motion estimation than pedestrian 2. In Cartesian-based methods, such differences are not explicitly modeled but instead must be inferred through learning. In contrast, the Polar coordinate system provides an explicit representation of relative distances and directions. This enables the model to directly capture the varying influence of traffic elements in a more structured manner, leading to more accurate motion estimation. This observation motivates the design of new approaches that can better encode the variations and relative relationships among traffic elements by leveraging the advantages of Polar representation.

Motivated by the above insights, we propose a novel framework, **Polaris**, for trajectory prediction and planning based on Polar representation. As shown in Figure 1 (b), both the input and output are represented in the Polar coordinate system. By leveraging the intuitive nature of Polar coordinates—expressed as (r, θ)—our approach effectively models the relative distances and directions between the

[1]School of Data Science, Fudan University
[2]Tongji University
*Corresponding author (lizhangfd@fudan.edu.cn).

target agent and surrounding traffic elements, as well as their temporal variations. The proposed framework consists of two main components: Polar Scene Context Encoding and Polar Relationship Refinement. The Polar Scene Context Encoding module encodes the motion states (e.g., position, velocity, acceleration) of agents, as well as lane geometry and lane-change information from HD maps. It captures the relative relationships and spatial dynamics within the scene context. The Polar Relationship Refinement module refines the proposal output trajectories by interacting with surrounding agents and the map, further modeling the relative relationships within the scene. This refinement enhances the accuracy of prediction and planning. To support this process, we design a Relative Embedding Transformer, which explicitly embeds relative distances and angles, enabling more effective relationship modeling within the Polar representation.

Our **contributions** are summarized as follows: **(i)** We are the first to perform trajectory prediction and planning entirely in the Polar coordinate. **(ii)** We propose Polaris, a Polar-based framework that incorporates Polar Scene Context Encoding and Polar Relationship Refinement, both equipped with Relative Embedding Transformer to model relative spatial relationships. **(iii)** Extensive experiments on the Argoverse 2 and nuPlan benchmarks demonstrate that Polaris achieves state-of-the-art performance.

II. Related work

a) Trajectory prediction: Recent advancements in autonomous driving highlight the importance of accurately predicting agent behavior through effective scene representation. Early methods [9], [10] typically converted driving scenarios into image-based formats and applied convolutional networks for context encoding, but often struggled to capture fine-grained structural details. This limitation motivated a transition to vectorized representations [2], [11], [12], enabling more structured and geometry-aware modeling of the environment. Building upon these representations, various frameworks have been proposed to predict multi-modal trajectories. Initial approaches focused on goal-oriented strategies [13], [14] and probability heatmaps [15], [16], while more recent models such as MTR [4], QCNet [5], and others [17], [3] leverage Transformer architectures [18] to better capture scene-agent interactions. These advances are further enhanced by emerging paradigms, including graph-based modeling [19], [20], pre-training [21], [22], history-aware designs [23], [24], GPT-style decoding [25], [26], and post-refinement techniques [27], [28]. In parallel, there has been increasing attention to multi-agent prediction, where models predict trajectories for all agents jointly, as explored in [8], [3], aiming to enhance consistency and interaction reasoning in complex driving scenarios.

b) Trajectory planning: As the final stage of autonomous driving, trajectory planning plays a key role in ensuring safe and precise navigation. A classic rule-based method is the Intelligent Driver Model (IDM) [29], which follows a leading vehicle while maintaining a safe distance. The release of the nuPlan [30] dataset and its standardized simulation benchmark has paved the way for advancing learning-based motion planners. Recent works explore purely learning-based approaches [7], [31], [32], [33], [34], while others combine learning and rule-based methods for hybrid planning [35]. Some methods explore accurate planning by leveraging large language models [36], [37], [38], [39], [40], [41], world models [42], [43], tree policy [44], generative models [45], [46], diffusion models [33], [47], mixture of experts [48], POMDP planning [49], and reinforcement learning [50], [51], [52], [53].

c) Polar representation in autonomous driving: While Cartesian representation is commonly used, Polar representation is favored in some methods for its inherent advantages in autonomous driving. In LiDAR-based object detection [54], [55], [56], Polar coordinates naturally align with the radial distribution of LiDAR data and the distance-dependent density variation in point clouds, allowing for more effective modeling of these characteristics. PolarFormer [57] further leverages a cross-attention mechanism to capture the geometric structure of BEV features in Polar space, achieving strong performance in camera-based object detection. However, the potential of Polar representation in trajectory prediction and planning remains underexplored. To the best of our knowledge, we are the first to introduce it in this context, and our method achieves notable improvements.

III. Methodology

In this section, we introduce **Polaris**, a novel framework for trajectory prediction and planning using Polar representation, as illustrated in Figure 2. The architecture comprises key components, including Polar Scene Context Encoding and Polar Relationship Refinement, both of which leverage a Relative Embedding Transformer, as shown in Figure 3. It also incorporates loss calculation in both Cartesian and Polar coordinate systems.

A. Problem formulation

Trajectory prediction and planning involve predicting the future trajectories of surrounding agents and planning the ego agent's path, based on the HD map and historical agent trajectories. We employ a vectorized representation, similar to that used in other methods [2], [4]. For the input, the HD map consists of several lane instances, each lane is composed of multiple points. Specifically, this can be represented as $M \in \mathbb{R}^{N_\mathrm{m} \times L \times C_\mathrm{m}}$, where N_m, L, and C_m denote the number of lane instances, points per lane, and feature channels (e.g., position), respectively. Agents, which include traffic participants such as vehicles and pedestrians, are represented as $A \in \mathbb{R}^{N_\mathrm{a} \times T_\mathrm{h} \times C_\mathrm{a}}$. Here, N_a, T_h, and C_a stand for the number of agents, historical timestamps, and motion states (e.g., position, velocity, acceleration). For the output, the model predicts the future trajectories $A_\mathrm{f} \in \mathbb{R}^{K \times N_\mathrm{aoi} \times T_\mathrm{f} \times 2}$ for the agents of interest, where K, N_aoi, T_f indicate the modalities, the number of agents of interest, and future timestamps, respectively. The associated probabilities $P_\mathrm{f} \in \mathbb{R}^{K \times N_\mathrm{aoi}}$ are predicted as well. We utilize Polar coordinates to represent

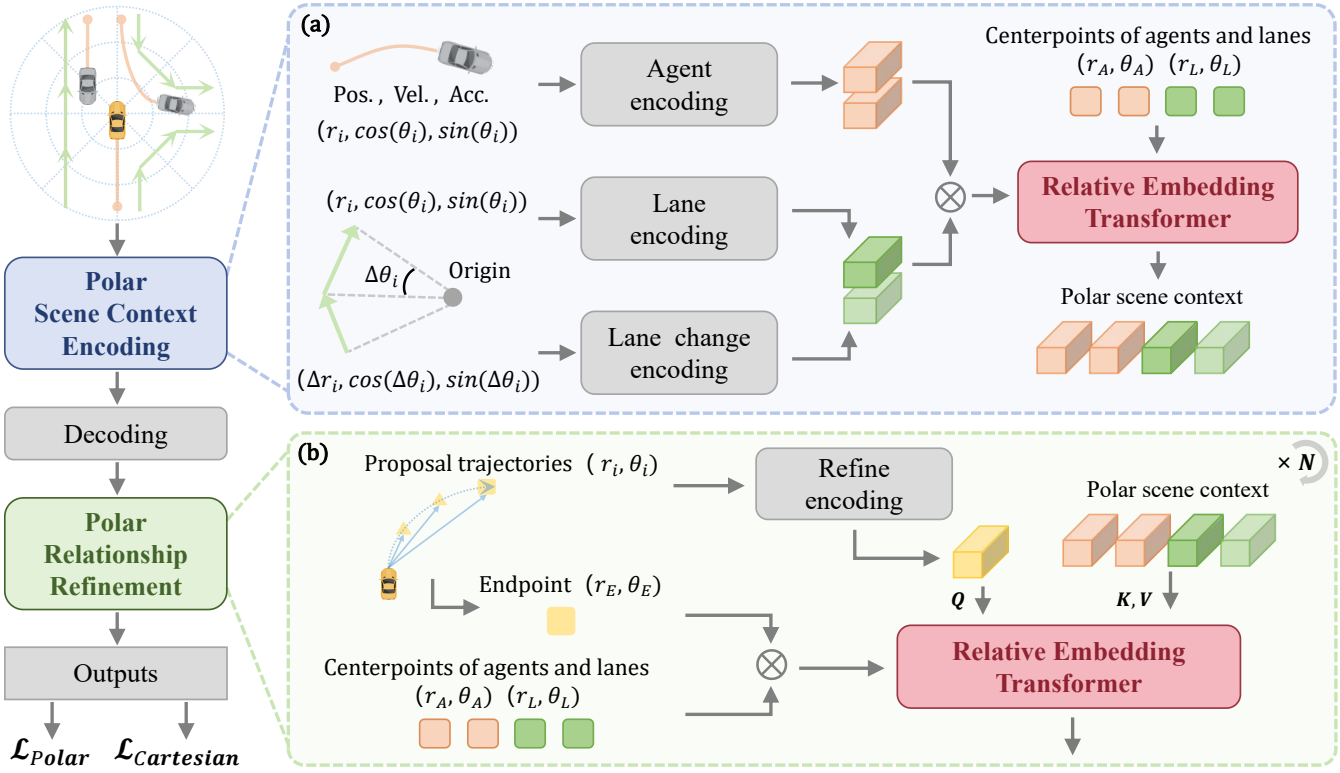

Fig. 2: Overview of our **Polaris** framework. (a) The Polar Scene Context Encoding separately encodes the motion states of agents, lanes, and lane changes, all within the Polar coordinate. It then employs the Relative Embedding Transformer to model interactions and derive the scene context. Subsequently, the decoding module predicts the proposal future trajectories in Polar representation. (b) The Polar Relationship Refinement then re-encodes the proposal trajectories and interacts with the scene context using the Relative Embedding Transformer to produce the final output. Losses are computed in both Cartesian and Polar coordinate.

position, velocity, acceleration, and other attributes as (r, θ) rather than (x, y).

B. Polar scene context encoding

As shown in Figure 2 (a), when we obtain the Polar representation of the HD map as M and agents as A, we need to encode them to effectively capture the scene context. To enhance the model's ability to learn, we input the Polar coordinates in the form of $(r, cos\theta, sin\theta)$. For the HD map, we utilize a PointNet-based lane encoder, as described in previous works [4], [21]. To leverage the advantages of Polar representation for modeling relative relationships, we compute the difference between the coordinates of adjacent lane points to obtain the lane change variable, denoted as $dM \in \mathbb{R}^{N_m \times (L-1) \times C_m}$. We use separate encoders to extract the map features $F_m \in \mathbb{R}^{N_m \times C}$ and $F_{dm} \in \mathbb{R}^{N_m \times C}$; we then combine them to derive the final map feature F_m. The process can be formulated as:

$$\begin{aligned} F_m &= \text{PointNet}_M(M), \\ F_{dm} &= \text{PointNet}_{dM}(dM), \\ F_m &= \text{MLP}(\text{Concat}(F_m, F_{dm})). \end{aligned} \tag{1}$$

For the agents, we aggregate the historical trajectory features up to the current time step, and utilize Mamba [58] blocks to extract features $F_a \in \mathbb{R}^{N_a \times C}$, given their exceptional capability for efficient and effective sequence modeling, as described in the previous work [6]. Subsequently, the scene context features $F_s \in \mathbb{R}^{(N_a+N_m) \times C}$ are formed by combining the agent and map features, which are then propagated through the Relative Embedding Transformer for intra-interaction learning. The centerpoints of lane instances and agents (i.e., the middle point of each lane instance and the current position of each agent) are also input into the Relative Embedding Transformer. Further details are discussed below in the Relative Embedding Transformer section. The overall process can be formulated as:

$$\begin{aligned} F_a &= \text{Mamba}(A), \\ F_s &= \text{Concat}(F_m, F_a), \\ F_s &= \text{Transformer}(F_s). \end{aligned} \tag{2}$$

C. Trajectory decoding

After obtaining the scene context features, we aim to decode multi-modal future trajectories for the agents of interest. We utilize vanilla Transformer blocks to facilitate interactions between the multi-modal trajectory queries and the scene context features. Subsequently, an MLP is employed

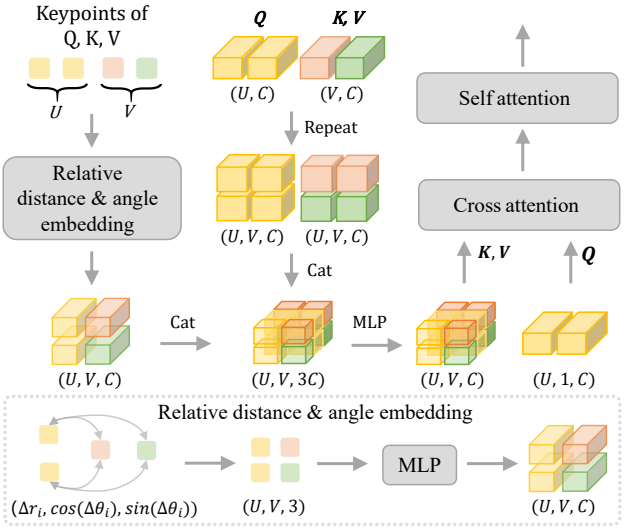

Fig. 3: Overview of our **Relative Embedding Transformer**. We use U queries and V keys and values as an example to illustrate the feature dimensions. For simplicity, the batch size dimension is omitted.

to decode the proposal trajectories $A_{\mathrm{f_{prop}}} \in \mathbb{R}^{K \times N_{\mathrm{aoi}} \times T_{\mathrm{f}} \times 2}$ in the Polar coordinate system, represented by (r, θ), along with their associated probabilities $P_{\mathrm{f_{prop}}} \in \mathbb{R}^{K \times N_{\mathrm{aoi}}}$.

D. Polar relationship refinement

As shown in Figure 2 (b), to further enhance the interaction between future trajectories and scene context features, we design Polar Relationship Refinement modules to refine the proposal trajectories. The proposal trajectories are re-encoded with an MLP to obtain the multi-modal trajectory queries $Q_{\mathrm{traj}} \in \mathbb{R}^{K \times N_{\mathrm{aoi}} \times C}$. Additionally, the endpoints $P_{\mathrm{E}}(r_{\mathrm{E}}, \theta_{\mathrm{E}})$ of the multi-modal trajectories, which are crucial for the accuracy of the overall trajectory, are extracted and combined with the centerpoints of lane instances and agents. The centerpoints are the same as those discussed in the encoder section. These keypoints of future trajectories and scene context features are used to model relative relationships within the Relative Embedding Transformer. Then, the trajectory queries Q_{traj}, the scene context features F_{s}, and the keypoints of query, key and value are input into the Relative Embedding Transformer. Finally, an MLP is used to obtain the refined outputs, including trajectories $A_{\mathrm{f_{ref}}}$ and probabilities $P_{\mathrm{f_{ref}}}$. The refinement module is iterated multiple times to achieve the final results. The refinement process can be described as:

$$\begin{aligned} Q_{\mathrm{traj}} &= \mathrm{MLP}(A_{\mathrm{f_{prop}}}), \\ Q_{\mathrm{traj}} &= \mathrm{Transformer}(\mathrm{Q} = Q_{\mathrm{traj}}, \mathrm{K}, \mathrm{V} = F_{\mathrm{s}}), \\ A_{\mathrm{f_{ref}}}, P_{\mathrm{f_{ref}}} &= \mathrm{MLP}(Q_{\mathrm{traj}}). \end{aligned} \tag{3}$$

E. Relative embedding transformer

As shown in Figure 3, taking full advantage of the strengths of Polar representation, the Relative Embedding Transformer is designed to embed the relative positions of the query, key, and value during interactions. First, the relative distances and angles of the keypoints (e.g., centerpoints for lane instances and agents, endpoints for future trajectories) in the query, key, and value are calculated in the form of $(\Delta r, cos(\Delta\theta), sin(\Delta\theta))$. An MLP is then used to encode these relative positions into relative position features. The relative relationships among the query, key, and value are determined by relative distances and angles of their keypoints in this manner. The query, key, and value features are repeated to match the size of the relative position features. The query, key, and value features, along with the relative position features, are then concatenated and passed through an MLP to adjust their dimensions, forming the new key and value. Finally, cross-attention and self-attention mechanisms are applied to obtain the output features. In the encoder part, the query, key, and value all correspond to the scene context features F_{s}. In the refinement part, the query consists of the multi-modal trajectory queries Q_{traj}, while the key and value remain the scene context features F_{s}.

F. Training losses

During the training process, predicted trajectories are supervised using the smooth-l1 loss $\mathcal{L}_{\mathrm{reg}}$ for regression, while the associated probabilities are supervised using the cross-entropy loss $\mathcal{L}_{\mathrm{cls}}$ for classification. Losses are calculated for both the proposal outputs and the refinement outputs. The total loss comprises two parts: the Polar loss, derived from the direct outputs in Polar coordinates (r, θ), and the Cartesian loss, derived from transforming the outputs into Cartesian coordinates (x, y). All losses are weighted equally, and the winner-take-all strategy is employed, optimizing only the best prediction with the minimal average prediction error compared to the ground truth. The overall process can be formulated as follows:

$$\begin{aligned} \mathcal{L}_{\mathrm{proposal/refine}} &= \mathcal{L}_{\mathrm{reg}} + \mathcal{L}_{\mathrm{cls}}, \\ \mathcal{L}_{\mathrm{Polar/Cartesian}} &= \mathcal{L}_{\mathrm{proposal}} + \mathcal{L}_{\mathrm{refine}}, \\ \mathcal{L}_{\mathrm{total}} &= \mathcal{L}_{\mathrm{Polar}} + \mathcal{L}_{\mathrm{Cartesian}}. \end{aligned} \tag{4}$$

IV. Experiments

A. Experimental settings

a) Datasets and evaluation metrics: We evaluate our method on two widely used and challenging datasets: the Argoverse 2 [1] dataset for trajectory prediction and the nuPlan [30] dataset for trajectory planning. The Argoverse 2 dataset is sampled at 10 Hz and provides 5 seconds of historical trajectories along with 6 seconds of future predictions. The nuPlan dataset uses a simulator that runs each scenario for 15 seconds at 10 Hz. For trajectory prediction, we use standard metrics, including $minADE$, $minFDE$, MR, and $b\text{-}minFDE$. For trajectory planning, nuPlan evaluates performance using three key metrics: the open-loop score (OLS), the non-reactive closed-loop score (NR-CLS), and the reactive closed-loop score (R-CLS). The evaluations cover 6 modes for both Argoverse 2 and nuPlan.

Method	$minFDE_1 \downarrow$	$minADE_1 \downarrow$	$minFDE_6 \downarrow$	$minADE_6 \downarrow$	$MR_6 \downarrow$	$b\text{-}minFDE_6 \downarrow$
MTR[4]	4.39	1.74	1.44	0.73	0.15	1.98
GANet [59]	4.48	1.77	1.34	0.72	0.17	1.96
ProphNet [60]	4.74	1.80	1.33	0.68	0.18	1.88
QCNet [5]	4.30	1.69	1.29	0.65	0.16	1.91
CaDeT [61]	4.33	1.74	1.24	0.67	0.15	1.86
SmartRefine [28]	4.17	1.65	1.23	0.63	0.15	1.86
RealMotion [62]	3.93	1.59	1.24	0.66	0.15	1.89
Polaris (Ours)	**3.78**	**1.53**	**1.15**	**0.62**	**0.13**	**1.80**
QCNet [5]	3.96	1.56	1.19	0.62	0.14	1.78
DyMap [63]	3.99	1.59	1.21	0.66	0.15	1.78
DeMo [6]	**3.70**	**1.49**	**1.11**	**0.60**	**0.12**	1.73
Polaris (Ours)	3.78	1.53	**1.11**	0.61	**0.12**	**1.71**

TABLE I: Performance of trajectory prediction *on the Argoverse 2 single-agent test set from the official leaderboard*. For each metric, the best result is highlighted in **bold**, and the second-best is underlined. The upper section reports results from single-model, while the lower section includes results with model ensembling.

Method	$avgMinFDE_1 \downarrow$	$avgMinADE_1 \downarrow$	$avgMinFDE_6 \downarrow$	$avgMinADE_6 \downarrow$	$actorMR_6 \downarrow$
FJMP [64]	4.00	1.52	1.89	0.81	0.23
Forecast-MAE [21]	3.33	1.30	1.55	0.69	0.19
RealMotion [62]	2.87	1.14	1.32	0.62	0.18
DeMo [6]	2.78	1.12	1.24	0.58	0.16
Polaris (Ours)	**2.67**	**1.07**	**1.18**	**0.56**	**0.15**

TABLE II: Performance of trajectory prediction *on the Argoverse 2 multi-agent test set from the official leaderboard.*

Type	Method	OLS ↑	NR-CLS ↑	R-CLS ↑
Rule	IDM [29]	0.20	0.56	0.62
	PDM-Closed [35]	0.26	0.65	0.75
Hybrid	GameFormer [65]	0.75	0.67	0.69
	PDM-Hybrid [35]	0.74	0.66	0.76
Learning	UrbanDriver [66]	0.77	0.52	0.49
	PDM-Open [35]	0.79	0.34	0.36
	PlanCNN [67]	0.52	0.49	0.52
	GC-PGP [68]	0.74	0.43	0.40
	PlanTF [7]	0.83	0.73	0.62
	BeTopNet [32]	0.84	**0.77**	0.69
	Polaris (Ours)	**0.86**	0.74	**0.70**

TABLE III: Performance of open-loop and closed-loop planning *on the nuPlan dataset under the Test 14 Hard benchmark.*

b) Implementation details: We train our models for 80 epochs using the AdamW optimizer, with a batch size of 4 per GPU. The training process is end-to-end with a learning rate of 0.001 and a weight decay of 0.01. Additionally, we use a cosine learning rate schedule with a 10-epoch warm-up phase. Experiments are conducted on 8 NVIDIA GeForce RTX 3090 GPUs. The training process requires about 20 hours. As for the number of layers in each component, the encoding module has 3 Relative Embedding Transformer layers, the decoding module has 2 vanilla Transformer layers, and there are 2 refining modules, each with 2 Relative Embedding Transformer layers.

B. Comparison with state of the art

We compare Polaris with several recent models on the Argoverse 2 dataset, as shown in Table I. To ensure a fair comparison, the performance is evaluated both with and without model ensembling. In the single-model setting, the results show that Polaris outperforms all previous methods across all metrics, including state-of-the-art models such as QCNet [5], SmartRefine [28], and RealMotion [62]. When employing ensembling techniques similar to those used by other methods, Polaris further improves its performance, significantly surpassing most existing approaches across all metrics and achieving results comparable to DeMo [6]. We also evaluate Polaris under the multi-agent setting, as reported in Table II, where it continues to demonstrate strong performance.

For the trajectory planning task, we compare Polaris with other top-performing methods on the nuPlan dataset under the Test 14 Hard benchmark, as shown in Table III. As a purely learning-based method with no hand-crafted rules, Polaris outperforms state-of-the-art models such as PlanTF [7] and BeTopNet [32], and achieves competitive performance compared to the rule-based method PDM-Closed [35].

C. Ablation study

a) Effects of components: Table IV demonstrates the effectiveness of each component in our method. The baseline is shown in ID-1. While the pipeline is similar to previous methods in the Cartesian coordinate system, we input and output in the Polar coordinate system, and only the agent position and lane position are encoded. In ID-2, the Agent and Lane Change Embedding is incorporated. By integrating changes in adjacent lane points and changes in agent position (i.e., velocity and acceleration), we observe a notable performance improvement. In ID-3, we add Polar Relationship Refinement modules using the vanilla Transformer. These modules aim to capture the relationship between the predicted trajectories and the scene context features. To

ID	A.L.C. Embed	Polar Refine	Rel. Transf.	$minFDE_1$	$minADE_1$	$minFDE_6$	$minADE_6$	MR_6
1				4.51	1.79	1.48	0.76	0.21
2	✓			4.46	1.78	1.40	0.72	0.19
3	✓	✓		4.26	1.69	1.35	0.70	0.18
4	✓	✓	✓	3.88	1.55	1.21	0.63	0.14

TABLE IV: Ablation study on the core components of Polaris *on the Argoverse 2 validation set*. "A.L.C. Embed" indicates Agent and Lane Change Embedding. "Polar Refine" indicates Polar Relationship Refinement module. "Rel. Transf." indicates Relative Embedding Transformer.

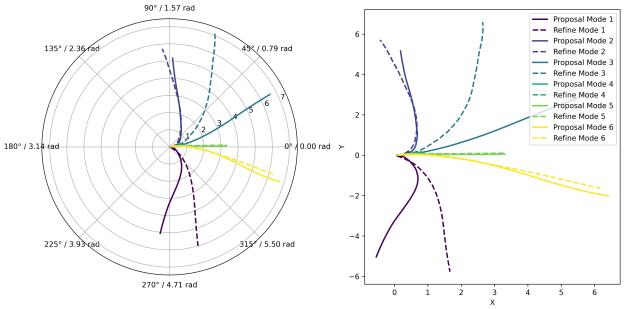

Fig. 4: Predicted multi-modal trajectories in the Polar representation (left), and their conversion to Cartesian representation (right). The solid line represents the future trajectory of the proposal, while the dashed line represents the future trajectory after refinement.

effectively improve prediction accuracy, they need to be combined with the Relative Embedding Transformer. As a result, the performance is only slightly better than that of ID-2, which uses the vanilla Transformer. Finally, in ID-4, we use the Relative Embedding Transformer to replace the vanilla Transformer. After integrating all these techniques, our model achieves outstanding performance.

b) Effects of losses: Table V demonstrates the effectiveness of different loss calculation strategies. We observe that computing the loss solely in either the Polar coordinate system or the Cartesian coordinate system leads to moderate performance. Notably, as shown in the third row, calculating the loss in both coordinate systems simultaneously results in a significant performance improvement. This indicates that leveraging both coordinate systems during training enables the model to make more accurate trajectory predictions.

Cartesian Loss	Polar Loss	$minFDE_6$	$minADE_6$	MR_6
✓		1.29	0.68	0.16
	✓	1.34	0.71	0.17
✓	✓	1.21	0.63	0.14

TABLE V: Ablation study on the losses.

c) Effects of the depth of refinement modules: Table VI shows the number of Polar Relationship Refinement modules. The first row presents the results without the refinement modules. We observe that a depth of two achieves an optimal balance between efficiency and performance.

Number	$minFDE_6$	$minADE_6$	MR_6
0	1.33	0.69	0.17
1	1.25	0.65	0.15
2	1.21	0.63	0.14
3	1.20	0.63	0.14

TABLE VI: Ablation study on the number of the Polar Relationship Refinement modules.

d) Polar $v.s.$ Cartesian: To highlight the advantages of Polar representation, we conduct an ablation study comparing the performance of Cartesian and Polar representations. As shown in Table VII, the first row represents the original version of the Cartesian representation, where all model components are kept consistent, varying only the input and output in the Cartesian coordinate system using (x, y). To ensure a fair comparison, we still calculate the relative relationships between traffic elements using $(\Delta r, \Delta\theta)$. The results indicate that the Cartesian representation is not naturally suited for extracting relative relationships, such as distances and angles between elements. This requires complex mathematical computations, leading to inference speeds nearly twice as slow as those achieved with Polar coordinates. In the second row, we modify the architecture for Cartesian-based input, using $(\Delta x, \Delta y)$ to calculate the relative relationships between traffic elements, better aligning with the Cartesian coordinate system, while keeping the other components consistent. The results show that, although faster than the first row, this approach still falls short because the Cartesian representation does not naturally capture the influence of traffic elements based on distance and direction, making it slightly less effective than the first row.

The comparison of results between the first, second, and third rows shows that trajectory prediction and planning in Polar representation better aligns with these relative relationships, leading to superior performance in the third row compared to the Cartesian representation in the first and second rows. This highlights the advantage of our approach.

Coordinate	$minFDE_6$	$minADE_6$	MR_6	Inf.
Cartesian_{ori}	1.30	0.68	0.16	110ms
Cartesian_{mod}	1.33	0.69	0.16	67ms
Polar	1.21	0.63	0.14	48ms

TABLE VII: Ablation study on the coordinate systems. "Inf." indicates inference speed.

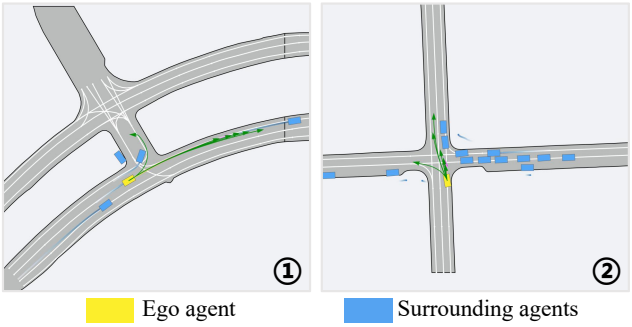

Fig. 5: Qualitative results *on the Argoverse 2 validation set*. The predicted trajectories are shown in green, and the ground truth trajectory is shown in pink.

D. Efficiency analysis and qualitative results

a) Efficiency analysis: Optimizing the trade-offs between performance, inference speed, and model size is crucial for deployment. We compare our model with recent state-of-the-art models on the Argoverse 2 benchmark. For model size, our model is 4.4M, whereas QCNet [5] is 7.7M, and SmartRefine [28] is 8.0M. Despite its smaller size, our model achieves superior performance compared to these recent high-performance methods. For inference speed, measured on an NVIDIA GeForce RTX 3090 GPU with a batch size of 1, the average inference time of Polaris is 48 ms, significantly faster than QCNet's 88 ms.

b) Qualitative results: We present qualitative results to highlight the effectiveness of our model. As shown in Figure 4, we provide an example of predicted multi-modal trajectories in the Polar coordinate system and their conversion to the Cartesian coordinate system. The solid line and the dashed line mean the proposal trajectories and the refined trajectories, respectively. As shown, predicting multi-modal trajectories involves forecasting trajectory point coordinates, which can result in large variations. However, in the Polar representation, both r and θ exhibit relatively small changes, making the predictions more stable and accurate compared to the larger variations in (x, y) coordinates in the Cartesian coordinate system. We can also observe the significant optimization of the trajectories after refinement. As shown in Figure 5, we also present the trajectory prediction results of our model. The design of our framework enables accurate forecasting of driving behavior. It not only predicts future trajectories precisely but also provides multi-modal outputs, capturing diverse behaviors such as turning, following, and lane-changing for overtaking.

V. Conclusion

We propose a framework, Polaris, that advances trajectory prediction and planning by leveraging the Polar coordinate system. Compared to traditional Cartesian-based approaches, our method more effectively captures both the movement of traffic elements and their relative spatial relationships. By integrating Polar Scene Context Encoding and Polar Relationship Refinement, both utilizing the Relative Embedding Transformer, Polaris enables precise modeling of interactions between traffic elements. Experiments on Argoverse 2 and nuPlan validate that Polaris achieves state-of-the-art performance in trajectory prediction and planning.

a) Limitations and future work: The conversion between Cartesian and Polar coordinates introduces computational overhead, which may affect efficiency. In future work, we aim to explore Polar representations in end-to-end autonomous driving.

Acknowledgements

This work was supported in part by New Generation Artificial Intelligence-National Science and Technology Major Project (2025ZD0123004), Ningbo grant (2025Z038) and National Natural Science Foundation of China (Grant No. 62376060).

References

[1] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes *et al.*, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," in *NeurIPS*, 2021.

[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 *CVPR*, 2020.

[3] J. Ngiam, V. Vasudevan, B. Caine, Z. Zhang, H.-T. 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 *ICLR*, 2022.

[4] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Motion transformer with global intention localization and local movement refinement," *NeurIPS*, 2022.

[5] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang, "Query-centric trajectory prediction," in *CVPR*, 2023.

[6] B. Zhang, N. Song, and L. Zhang, "Decoupling motion forecasting into directional intentions and dynamic states," in *NeurIPS*, 2024.

[7] J. Cheng, Y. Chen, X. Mei, B. Yang, B. Li, and M. Liu, "Rethinking imitation-based planners for autonomous driving," in *ICRA*, 2024.

[8] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying," *IEEE TPAMI*, 2024.

[9] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," in *CoRL*, 2020.

[10] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," in *CVPR*, 2020.

[11] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov *et al.*, "Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction," in *ICRA*, 2022.

[12] Z. Zhou, L. Ye, J. Wang, K. Wu, and K. H. Lu, "Hierarchical vector transformer for multi-agent motion prediction," in *CVPR*, 2022.

[13] J. Gu, C. Sun, and H. Zhao, "Densetnt: End-to-end trajectory prediction from dense goal sets," in *ICCV*, 2021.

[14] L. Zhang, P. Li, J. Chen, and S. Shen, "Trajectory prediction with graph-based dual-scale context fusion," in *IROS*, 2022.

[15] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Home: Heatmap output for future motion estimation," in *ITSC*, 2021.

[16] ——, "Gohome: Graph-oriented heatmap output for future motion estimation," in *ICRA*, 2022.

[17] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp, "Wayformer: Motion forecasting via simple & efficient attention networks," in *ICRA*, 2023.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, 2017.

[19] X. Jia, P. Wu, L. Chen, Y. Liu, H. Li, and J. Yan, "Hdgt: Heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding," *IEEE TPAMI*, 2023.

[20] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *ECCV*, 2020.

[21] J. Cheng, X. Mei, and M. Liu, "Forecast-mae: Self-supervised pre-training for motion forecasting with masked autoencoders," in *ICCV*, 2023.

[22] Z. Lan, Y. Jiang, Y. Mu, C. Chen, and S. E. Li, "Sept: Towards efficient scene representation learning for motion prediction," in *ICLR*, 2024.

[23] D. Park, J. Jeong, S.-H. Yoon, J. Jeong, and K.-J. Yoon, "T4p: Test-time training of trajectory prediction via masked autoencoder and actor-specific token memory," in *CVPR*, 2024.

[24] X. Tang, M. Kan, S. Shan, Z. Ji, J. Bai, and X. Chen, "Hpnet: Dynamic trajectory forecasting with historical prediction attention," in *CVPR*, 2024.

[25] J. Philion, X. B. Peng, and S. Fidler, "Trajeglish: Learning the language of driving scenarios," in *ICLR*, 2024.

[26] A. Seff, B. Cera, D. Chen, M. Ng, A. Zhou, N. Nayakanti, K. S. Refaat, R. Al-Rfou, and B. Sapp, "Motionlm: Multi-agent motion forecasting as language modeling," in *ICCV*, 2023.

[27] S. Choi, J. Kim, J. Yun, and J. W. Choi, "R-pred: Two-stage motion prediction via tube-query attention-based trajectory refinement," in *ICCV*, 2023.

[28] Y. Zhou, H. Shao, L. Wang, S. L. Waslander, H. Li, and Y. Liu, "Smartrefine: A scenario-adaptive refinement framework for efficient motion prediction," in *CVPR*, 2024.

[29] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, 2000.

[30] N. Karnchanachari, D. Geromichalos, K. S. Tan, N. Li, C. Eriksen, S. Yaghoubi, N. Mehdipour, G. Bernasconi, W. K. Fong, Y. Guo *et al.*, "Towards learning-based planning: The nuplan benchmark for real-world autonomous driving," *arXiv preprint*, 2024.

[31] J. Cheng, Y. Chen, and Q. Chen, "Pluto: Pushing the limit of imitation learning-based planning for autonomous driving," *arXiv preprint*, 2024.

[32] H. Liu, L. Chen, Y. Qiao, C. Lv, and H. Li, "Reasoning multi-agent behavioral topology for interactive autonomous driving," in *NeurIPS*, 2024.

[33] Y. Zheng, R. Liang, K. ZHENG, J. Zheng, L. Mao, J. Li, W. Gu, R. Ai, S. E. Li, X. Zhan, and J. Liu, "Diffusion-based planning for autonomous driving with flexible guidance," in *ICLR*, 2025.

[34] X. Chen, J. Yan, W. Liao, T. He, and P. Peng, "Int2planner: An intention-based multi-modal motion planner for integrated prediction and planning," in *AAAI*, 2025.

[35] D. Dauner, M. Hallgarten, A. Geiger, and K. Chitta, "Parting with misconceptions about learning-based vehicle motion planning," in *CoRL*, 2023.

[36] Y. Zheng, Z. Xing, Q. Zhang, B. Jin, P. Li, Y. Zheng, Z. Xia, K. Zhan, X. Lang, Y. Chen *et al.*, "Planagent: A multi-modal large language agent for closed-loop vehicle motion planning," *arXiv preprint*, 2024.

[37] M. Hallgarten, J. Zapata, M. Stoll, K. Renz, and A. Zell, "Can vehicle motion planning generalize to realistic long-tail scenarios?" in *IROS*, 2024.

[38] R. Zhang, X. Guo, W. Zheng, C. Zhang, K. Keutzer, and L. Chen, "Instruct large language models to drive like humans," *arXiv preprint*, 2024.

[39] G. Kou, F. Jia, W. Mao, Y. Liu, Y. Zhao, Z. Zhang, O. Yoshie, T. Wang, Y. Li, and X. Zhang, "Padriver: Towards personalized autonomous driving," *arXiv preprint*, 2025.

[40] R. Yao, Y. Wang, H. Liu, R. Yang, Z. Peng, L. Zhu, and J. Ma, "Calmm-drive: Confidence-aware autonomous driving with large multimodal model," *arXiv preprint*, 2024.

[41] Y. Chen, Z.-h. Ding, Z. Wang, Y. Wang, L. Zhang, and S. Liu, "Asynchronous large language model enhanced planner for autonomous driving," in *ECCV*, 2024.

[42] A. B. Vasudevan, N. Peri, J. Schneider, and D. Ramanan, "Planning with adaptive world models for autonomous driving," *arXiv preprint*, 2024.

[43] L. Xiao, J.-J. Liu, S. Yang, X. Li, X. Ye, W. Yang, and J. Wang, "Learning multiple probabilistic decisions from latent world model in autonomous driving," *arXiv preprint*, 2024.

[44] Z. Huang, P. Karkus, B. Ivanovic, Y. Chen, M. Pavone, and C. Lv, "Dtpp: Differentiable joint conditional prediction and cost evaluation for tree policy planning in autonomous driving," in *ICRA*, 2024.

[45] Y. Hu, S. Chai, Z. Yang, J. Qian, K. Li, W. Shao, H. Zhang, W. Xu, and Q. Liu, "Solving motion planning tasks with a scalable generative model," in *ECCV*, 2024.

[46] Z. Xie, S. Zuo, W. Zheng, Y. Zhang, D. Du, J. Zhou, J. Lu, and S. Zhang, "Gpd-1: Generative pre-training for driving," *arXiv preprint*, 2024.

[47] B. Yang, H. Su, N. Gkanatsios, T.-W. Ke, A. Jain, J. Schneider, and K. Fragkiadaki, "Diffusion-es: Gradient-free planning with diffusion for autonomous and instruction-guided driving," in *CVPR*, 2024.

[48] Q. Sun, H. Wang, J. Zhan, F. Nie, X. Wen, L. Xu, K. Zhan, P. Jia, X. Lang, and H. Zhao, "Generalizing motion planners with mixture of experts for autonomous driving," *arXiv preprint*, 2024.

[49] Z. Huang, C. Tang, C. Lv, M. Tomizuka, and W. Zhan, "Learning online belief prediction for efficient pomdp planning in autonomous driving," *arXiv preprint*, 2024.

[50] D. Zhang, J. Liang, K. Guo, S. Lu, Q. Wang, R. Xiong, Z. Miao, and Y. Wang, "Carplanner: Consistent auto-regressive trajectory planning for large-scale reinforcement learning in autonomous driving," in *CVPR*, 2025.

[51] X. Tang, M. Kan, S. Shan, and X. Chen, "Plan-r1: Safe and feasible trajectory planning as language modeling," *arXiv preprint*, 2025.

[52] Z. Huang, X. Weng, M. Igl, Y. Chen, Y. Cao, B. Ivanovic, M. Pavone, and C. Lv, "Gen-drive: Enhancing diffusion generative driving policies with reward modeling and reinforcement learning fine-tuning," *arXiv preprint*, 2024.

[53] B. Jaeger, D. Dauner, J. Beißwenger, S. Gerstenecker, K. Chitta, and A. Geiger, "Carl: Learning scalable planning policies with simple rewards," *arXiv preprint*, 2025.

[54] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh, "Polarnet: An improved grid representation for online lidar point clouds semantic segmentation," in *CVPR*, 2020.

[55] Q. Chen, S. Vora, and O. Beijbom, "Polarstream: Streaming object detection and segmentation with polar pillars," in *NeurIPS*, 2021.

[56] M. Nie, Y. Xue, C. Wang, C. Ye, H. Xu, X. Zhu, Q. Huang, M. B. Mi, X. Wang, and L. Zhang, "Partner: Level up the polar representation for lidar 3d object detection," in *ICCV*, 2023.

[57] Y. Jiang, L. Zhang, Z. Miao, X. Zhu, J. Gao, W. Hu, and Y.-G. Jiang, "Polarformer: Multi-camera 3d object detection with polar transformer," in *AAAI*, 2023.

[58] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint*, 2023.

[59] M. Wang, X. Zhu, C. Yu, W. Li, Y. Ma, R. Jin, X. Ren, D. Ren, M. Wang, and W. Yang, "Ganet: Goal area network for motion forecasting," in *ICRA*, 2023.

[60] X. Wang, T. Su, F. Da, and X. Yang, "Prophnet: Efficient agent-centric motion forecasting with anchor-informed proposals," in *CVPR*, 2023.

[61] M. Pourkeshavarz, J. Zhang, and A. Rasouli, "Cadet: a causal disentanglement approach for robust trajectory prediction in autonomous driving," in *CVPR*, 2024.

[62] N. Song, B. Zhang, X. Zhu, and L. Zhang, "Motion forecasting in continuous driving," in *NeurIPS*, 2024.

[63] B. Fan, H. Yuan, Y. Dong, Z. Zhu, and H. Liu, "Bidirectional agent-map interaction feature learning leveraged by map-related tasks for trajectory prediction in autonomous driving," *IEEE Transactions on Automation Science and Engineering*, 2025.

[64] L. Rowe, M. Ethier, E.-H. Dykhne, and K. Czarnecki, "Fjmp: Factorized joint multi-agent motion prediction over learned directed acyclic interaction graphs," in *CVPR*, 2023.

[65] Z. Huang, H. Liu, and C. Lv, "Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving," in *ICCV*, 2023.

[66] O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska, "Urban driver: Learning to drive from real-world demonstrations using policy gradients," in *CoRL*, 2022.

[67] K. Renz, K. Chitta, O.-B. Mercea, A. S. Koepke, Z. Akata, and A. Geiger, "Plant: Explainable planning transformers via object-level representations," in *CoRL*, 2022.

[68] M. Hallgarten, M. Stoll, and A. Zell, "From prediction to planning with goal conditioned lane graph traversals," in *ITSC*, 2023.