Swin-Trajectory: Technical Report for 2025 Waymo Vision-based End-to-End Driving Challenge

Sungjin Park[1†] Gwangik Shin[1†] Jaeha Song[1†] Sumin Lee[1] Hyukju Shon[1]
Byounggun Park[1] Jinhee Na[2] Hawook Jeong[2] Soonmin Hwang[1]
[1]Hanyang University [2]RideFlux Inc.
{shihtzu333, david5432, archiiive99, maroona, sohnhyck, okharry1, soonminh}@hanyang.ac.kr
{jhna, hawook}@rideflux.com

Abstract

End-to-end autonomous driving aims to predict motion plans or control commands directly from raw sensor inputs. Although recent methods often incorporate vision language models (VLMs) or auxiliary tasks, we take a minimalist approach called Swin-Trajectory, a transformer-based waypoint predictor that uses only a single front-facing camera and structured ego-vehicle history. We employ a lightweight Swin Transformer as backbone to extract dense image features, and use cross-attention between these features and waypoint queries—derived from historical trajectories and ego states—to capture spatial-temporal context for trajectory prediction. Our model runs at 14ms on an RTX 4090 and achieves competitive performance in the challenge.

1. Introduction

End-to-end (E2E) autonomous driving is an emerging paradigm in which driving behaviors are inferred directly from sensory inputs using a single unified model. Compared to traditional modular approaches, E2E methodology has gained considerable attention due to its potential to holistically optimize the entire driving system, resulting in significant improvements in safety and performance.

Recent end-to-end (E2E) methods have shown strong performance by incorporating auxiliary perception tasks [1, 2] and further leveraging vision language models (VLMs), such as EMMA [3], which leverages their reasoning capabilities to handle even more challenging scenes. However, such approaches inevitably result in larger and more complex model architectures. In contrast, we take a minimalist approach: a lightweight, deployment-friendly model trained solely with trajectory supervision.

†equal contribution

Our model, **Swin-Trajectory**, is a streamlined E2E planner that utilizes only a single front-facing camera and ego-vehicle status without any additional supervision. Through careful engineering to mitigate shortcut learning, our model achieves a compelling balance between accuracy and computational efficiency.

2. Method

To predict a sequence of future waypoints in the vehicle-centric coordinate frame, we utilize a single front-facing camera and structured waypoint queries. As shown in Figure 1, our architecture consists of 3 components: feature extractor, ego-info encoder and trajectory decoder module.

2.1. Feature Extraction

We begin by extracting visual features from the input image using a Swin Transformer [4] as backbone. To provide geometric grounding for the dense image features, we utilize 3D positional encoding. Specifically, we sample dense pixel locations from the feature map grid and project each pixel into 3D space using a set of predefined depth bins. The resulting 3D coordinates are normalized and passed through a linear positional encoder to obtain 3D positional encoding.

2.2. Ego-info Encoder

To encode the historical waypoints and ego-vehicle states into waypoint queries, the historical waypoints $\mathbf{P} \in \mathbb{R}^{B\times N_{\text{his}}\times 2}$ are flattened into $\mathbb{R}^{B\times 2N_{\text{his}}}$ and passed through a linear layer to produce initial query embeddings $\mathbf{Q}_{his} \in \mathbb{R}^{B\times N_{\text{pred}}\times C}$.

The ego-vehicle state vector $\mathbf{E} \in \mathbb{R}^{B\times 12}$ includes velocity, acceleration, yaw rate, initial speed, curvature estimated from past trajectory, fixed vehicle size, and a one-hot encoded driving intention. The ego-vehicle state vector $\mathbf{E} \in \mathbb{R}^{B\times 12}$ is then encoded via another linear layer to generate both keys and values $\mathbf{K}_e, \mathbf{V}_e \in \mathbb{R}^{B\times N_{\text{pred}}\times C}$.

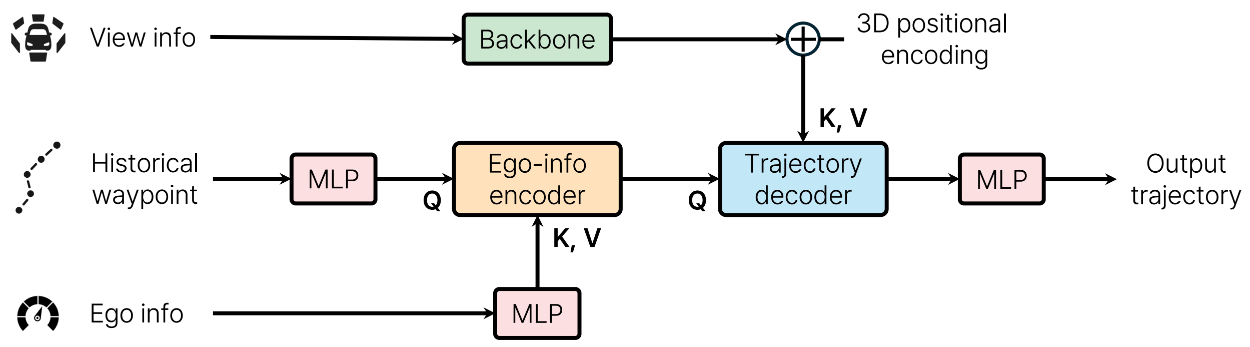

Figure 1. Overall Architecture. The model comprises three main components: (1) a Swin Transformer backbone for extracting spatial image features with 3D positional encoding, (2) a waypoint query encoder that fuses historical waypoints and ego-vehicle states via *Ego-info Encoder* to produce refined waypoint queries, (3) a Transformer decoder that applies *Trajectory Decoder* between the waypoint queries and flattened image features to predict future trajectories.

These are fused via a single transformer layer, where $\mathbf{Q}_{his}$ serves as the query and $\mathbf{K}_e$, $\mathbf{V}_e$ serve as the memory, resulting in refined waypoint queries $\mathbf{Q}_{way} \in \mathbb{R}^{B\times N_{\text{pred}}\times C}$ that integrate both temporal history and structured ego information. These queries are then used in subsequent decoder blocks with image features for final prediction.

2.3. Trajectory Decoder

The trajectory decoder block takes the image features $\mathbf{F} \in \mathbb{R}^{B\times N_{\text{cam}}\times H\times W\times C}$ as input from Section 2.1. These features are first flattened along the spatial and camera dimensions to form a sequence: $\mathbf{F}_{\text{flat}} \in \mathbb{R}^{B\times(N_{\text{cam}}\cdot H\cdot W)\times C}$. The flattened feature sequence serves as the key and value input to a transformer layer, while the refined waypoint queries $\mathbf{Q}_{\text{way}} \in \mathbb{R}^{B\times N_{\text{pred}}\times C}$ from Section 2.2 are used as the queries. The transformer layers fuses spatial image features with waypoint query, and the final trajectory is predicted through a linear projection layer applied to the decoder output.

2.4. Loss

We utilize the L2 loss for trajectory prediction. Given the predicted trajectory $\hat{\mathbf{Y}} \in \mathbb{R}^{N\times 2}$ and the ground truth trajectory $\mathbf{Y} \in \mathbb{R}^{N\times 2}$, the loss is defined as:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{\mathbf{Y}}_i - \mathbf{Y}_i\right\|_2^2 \tag{1}$$

3. Experiments

3.1. Dataset and metrics

We use the Waymo Open Dataset for End-to-End Driving (WOD-E2E) for our experiments. Each data sample includes multi-view camera images, past trajectories and ego status. The prediction target is a 5-second future trajectory in an open-loop setting.

The Rater Feedback Score (RFS) evaluates trajectory prediction by measuring deviation from human rater trajectories within predefined trust regions. At $t = 3$ and $t = 5$ seconds, predictions are scored based on lateral and longitudinal thresholds, scaled by the ego's initial speed. Full score is awarded within the region, while deviations incur exponential penalties. The final score is clipped between 4 and 10. and the penalty is computed as:

$$\text{RFS} = \bar{s} \times 0.1^{\max\left(\max\left[\frac{\Delta_{\text{lng}}}{\tau_{\text{lng}}}, \frac{\Delta_{\text{lat}}}{\tau_{\text{lat}}}\right]-1,0\right)} \tag{2}$$

On the validation set, a single ground-truth trajectory is treated as a single rater with $\bar{s} = 10$.

3.2. Sampling Strategy

During training, image sequences are provided at approximately 10 Hz. We observed that important driving scenarios typically occur in the middle of a sequence. Consequently, using all available frames would introduce significant redundancy for common scenarios, diminishing the model's ability to effectively handle rare but critical situations. To address this, we sample frames at fixed intervals of 15 frames, which reduces redundancy and training cost.

For the same reason, using every frame on evaluation would lead to exaggerated scores, as many frames correspond to less-complex driving situations. This may obscure the model's performance in more critical or challenging scenarios. We adopt a two sampling strategy to address this: we select a single representative frame per sequence—the frame 80 frames before the end of each scene, and sample every 20 frames from whole validation set.

			Validation		Test			
#Cameras	**#Blocks**	**Semantic**	**RFS(Single) ↑**	**RFS(Interval) ↑**	**RFS ↑**	**ADE ↓**	**GFLOPs**	**Latency (ms)**
1	1	–	6.52	7.18	7.49	2.90	93.34	13.24
1	1	✓	6.70	7.40	7.41	3.02	93.36	13.28
1	3	–	6.63	7.28	**7.54**	**2.81**	93.67	13.83
3	1	–	6.37	7.12	7.41	2.93	279.83	30.19
3	3	–	6.41	7.15	7.46	2.98	280.69	31.34

Table 1. **Performance on WOD-E2E validation set.** **#Cameras** indicates the number of input camera views, and **#Blocks** refers to the number of trajectory decoder block used. **Semantic** indicates whether semantic masks are used, which serve as the fourth channel when concatenated with the RGB image to form a 4-channel input. **RFS(Single)** is computed from each scene's 80 frames (8 seconds at 10 Hz) before the last frame, while **RFS(Interval)** is averaged over frames sampled every 20 frames across each scene. **RFS** and **ADE** represent scores obtained from the official submission on the test set.

Scenario Cluster	**1-Camera**	**3-Camera**
Foreign Object Debris	**6.58**	6.07
Intersections	**6.73**	6.41
Special Vehicles	6.11	**6.45**
Cyclist	**6.57**	6.29
Multi-Lane Maneuvers	**6.59**	6.14
Pedestrian	6.73	**6.96**
Cut-ins	**6.09**	6.07
Single-Lane Maneuvers	**6.37**	6.33
Construction	5.84	**6.73**
Others	**8.53**	7.32
Overall	**6.63**	6.41

Table 2. **Rater Feedback Score per scenario cluster on the single sampled validation set.** Each value represents the average Rater Feedback Score for each scenario cluster as defined in WOD-E2E. All results are based on 3-block models.

3.3. Implementation Details

Our experiments are conducted on NVIDIA RTX 4090 GPU. We employ a Swin Transformer-Tiny's image encoder pre-trained in ImageNet, with patch size (2, 4, 4) and window size (8, 7, 7). The encoder processes image with a spatial resolution of 800×972. We train our model on the training split for 50 epochs using the AdamW optimizer with a batch size of 8. The learning rate is set to 1×10^{-6} for the image backbone and 1×10^{-5} for the other modules, with weight decay of 0.01. We adopt a cosine annealing learning rate scheduler with linear warmup for the first 500 iterations. The warmup ratio is set to 0.33, and the minimum learning rate is set to 1×10^{-3} of the initial value. Gradient clipping with a maximum norm of 35 is applied to stabilize training. Dropout with a ratio of 0.1 is applied throughout the backbone and decoder block. No model ensembling is used during training. For experiments using additional semantic masks, we employ UPerNet [5] to generate semantic segmentation features.

3.4. Results

As shown in Table 1, increasing the number of trajectory decoder block improves performance. The 3-block variant without semantic input achieved the highest test score, validating the effectiveness of our minimal design.

As an additional experiment, we investigated whether incorporating semantic information could further improve performance. We used the output of a pre-trained semantic segmentation model and appended a single-channel semantic prior to the RGB input, forming a 4-channel image. While this variant yielded better results on the validation set, it underperformed on the test set—likely due to noise and domain gap in the segmentation outputs.

When comparing the 1-camera and 3-camera setups, we observe that the 3-camera configuration resulted in lower overall performance, despite its higher computational cost. We assume that this degradation is due to the model's limited capacity (36M parameters), which struggles to disentangle redundant or misaligned features introduced by flattening and jointly encoding multiple views. This observation suggests that increasing the number of input views alone does not guarantee better performance. Instead, it emphasizes the importance of adaptive and content-aware multi-view fusion strategies that selectively extract complementary information across views.

As shown in Table 2 and Figure 2, there are scene types—such as *Construction*—where the 3-camera setup clearly outperforms the 1-camera variant. This suggests that additional views can be beneficial, supporting the idea that multi-view input holds strong potential, but its success hinges on how effectively the information is integrated.

4. Conclusion

In this report, we introduced **Swin-Trajectory**, a lightweight and minimalist end-to-end driving model designed for the Waymo Vision-based E2E Driving Challenge. Our approach leverages only a single front-facing

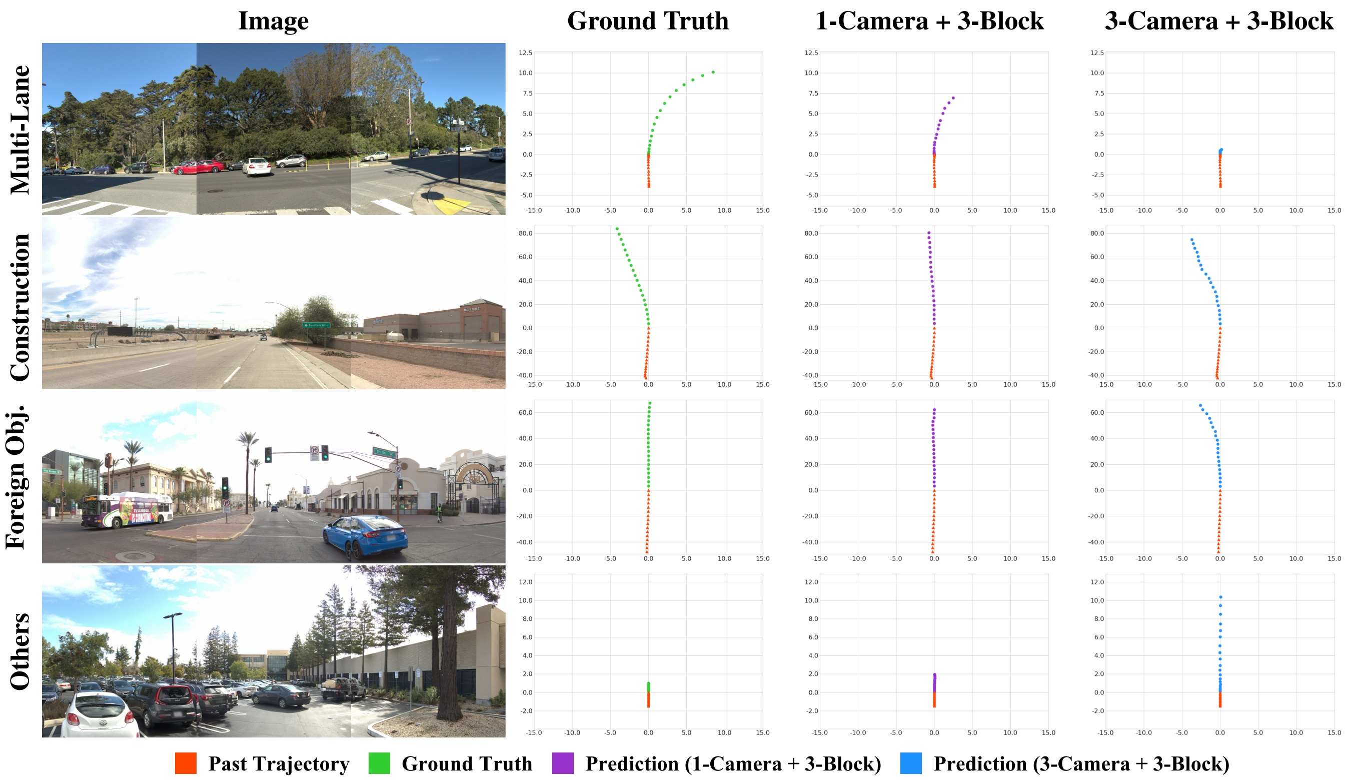

Figure 2. Qualitative results across different scenario clusters. We compare predictions from the 1-Camera + 3-Block and 3-Camera + 3-Block settings. Except for the *Construction* scenario, the 1-Camera + 3-Block setting generally outperforms the 3-Camera counterpart. Each row presents a qualitative visualization from the validation set, highlighting representative differences across clusters. For clarity, some cluster names are abbreviated: ”Multi-Lane” stands for *Multi-Lane Maneuvers*, and ”Foreign Obj.” refers to *Foreign Object Debris*.

camera and structured ego inputs, avoiding any reliance on semantic labels or HD maps. Despite its simplicity, Swin-Trajectory effectively models both spatial and temporal contexts through an encoder–decoder attention framework: *ego-info encoder* integrates historical trajectory and vehicle state, while *trajectory decoder* enables cross-attention between waypoint queries and dense image features.

Extensive experiments on WOD-E2E validation and test sets demonstrate that our architecture achieves competitive results with low latency and computation cost. Notably, we show that increasing the number of decoder block significantly improves performance in the single-camera setup. While the 3-camera variant exhibits advantages in certain scenarios like construction zones, our results suggest that multi-view fusion requires more sophisticated handling to consistently outperform the single-view baseline. Future works may explore adaptive fusion strategies and integration of external priors for further improvement.

References

[1] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17853–17862, 2023. 1

[2] Xinshuo Weng, Boris Ivanovic, Yan Wang, Yue Wang, and Marco Pavone. Para-drive: Parallelized architecture for real-time autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15449–15458, June 2024. 1

[3] Jyh-Jing Hwang, Runsheng Xu, Hubert Lin, Wei-Chih Hung, Jingwei Ji, Kristy Choi, Di Huang, Tong He, Paul Covington, Benjamin Sapp, et al. Emma: End-to-end multimodal model for autonomous driving. *arXiv preprint arXiv:2410.23262*, 2024. 1

[4] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 1

[5] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, pages 418–434, 2018. 3