# *ResAD*: Normalized Residual Trajectory Modeling for End-to-End Autonomous Driving

Supplementary Material

## 6. Further Implementation Details

Following the Transfuser baseline, we incorporate two auxiliary tasks, 3D object detection and 2D Bird's-Eye-View (BEV) semantic segmentation. The agent queries $E_{\text{agent}}$ derived from the detection task and the BEV features $E_{\text{BEV}}$ from the segmentation task are subsequently fed into our proposed diffusion decoder. We utilize weights pre-trained on the ImageNet dataset to initialize the model. The LiDAR sensor is configured with a perception range of 32 meters in the forward, backward, left, and right directions. To incorporate the vehicle's own state, the ego status vector includes the current velocity, acceleration, and the driving command. This vector is processed through a Multi-Layer Perceptron (MLP) to generate a state embedding, denoted as $E$.

**Diffusion Decoder.** Given the set of sampled noisy normalized residuals $\{z_k^{(i)}\}_{k=1}^{K}$ anchored to the inertial reference cluster, we first apply PRNorm to stabilize the input feature distribution. The decoder then employs a sequence of cross-attention mechanisms to interact with the multi-modal context: it first queries the Bird's Eye View (BEV) features, followed by cross-attention with Agent features and Ego status features to capture dynamic scene interactions and vehicle states. These aggregated features are processed by a FFN. To incorporate conditional guidance, we utilize a Modulation layer that injects information from both the diffusion timestep embeddings and the unique positional encodings derived from the perturbed inertial references. An MLP head subsequently predicts the refined residual features, which are mapped back to the original physical scale via the De-PRNorm operation. This structure forms a cascaded diffusion decoder. During inference, ResAD iteratively denoises the residuals through these layers, adds them to the corresponding inertial references, and utilizes the Trajectory Ranker to select the optimal final trajectory.

**NAVSIM v1.** In this benchmark, each predicted trajectory is sent to a simulator, which validates the driving metrics in the corresponding environment. The planning capabilities of models are assessed using the PDM score (PDMS), which is calculated as follows:

$$\text{PDMS} = \text{NC} \times \text{DAC} \times \frac{(5 \times \text{TTC} + 2 \times \text{C} + 5 \times \text{EP})}{12}, \tag{14}$$

where the sub-metrics NC, DAC, TTC, C, EP represent the No At-Fault Collisions, Drivable Area Compliance, Time to Collision, Comfort, and Ego Progress.

**NAVSIM v2.** A new Extended PDM Score (EPDMS) is introduced in NAVSIM v2, which can be formulated as:

$$\begin{aligned}\text{EPDMS} = \text{NC} \times \text{DAC} \times \text{DDC} \times \text{TL} \times \\ \frac{(5 \times \text{TTC} + 2 \times \text{C} + 5 \times \text{EP} + 5 \times \text{LK} + 5 \times \text{EC})}{22}.\end{aligned} \tag{15}$$

The extended sub-metrics DDC, TL, LK, and EC correspond to the Driving Direction Compliance, Traffic Lights Compliance, Lane Keeping Ability, and Extended Comfort.

**Multimodal Trajectory Ranker.** We employ a two-stage training on ***ResAD***. Initially, the trajectory planner is trained. This pre-trained planner is then utilized to generate training data for a subsequent Multimodal Trajectory Ranker. The objective of the Ranker is to identify the optimal trajectory from the multiple candidates proposed by the planner. For each candidate trajectory, the Ranker receives the trajectory itself and an associated environmental feature $E_{\text{env}}$ as input. $E_{\text{env}}$ is formed by concatenating the agent query $E_{\text{agent}}$ with the corresponding BEV features $E_{\text{BEV}}$ as $E_{\text{env}} = \text{Concat}(E_{\text{agent}}, E_{\text{BEV}})$. We set the learning rate for the Ranker to $1 \times 10^{-4}$ and trained the model for 50 epochs.

## 7. Further Qualitative Comparison

In this section, we provide additional visualization results for challenging scenarios from the NAVTEST split of the NAVSIM dataset. The red circle encloses the failure cases of the trajectory predicted by DiffusionDrive. The prediction of our proposed method is depicted by the orange circle and arrow, which is then projected onto the front-view image for visualization.

**Going straight.** Fig. 7 highlights the top-1 and top-5 scoring trajectories of DiffusionDrive and the proposed ***ResAD*** in going straight scenarios. In straight-driving scenarios, the proposed method demonstrably avoids generating trajectories that would lead to a collision with the lead vehicle. This validates the effectiveness of our **Normalized Residual Trajectory Modeling**. To avert a potential collision implied by the inertial reference, ***ResAD*** opts to decelerate or change lanes. Furthermore, even in these straight-driving situations, ***ResAD*** actively explores plausible and context-aware maneuvers for lane-changing and overtaking.

**Turning left.** Fig. 8 highlights the top-1 and top-5 scoring trajectories of DiffusionDrive and the proposed ***ResAD*** in

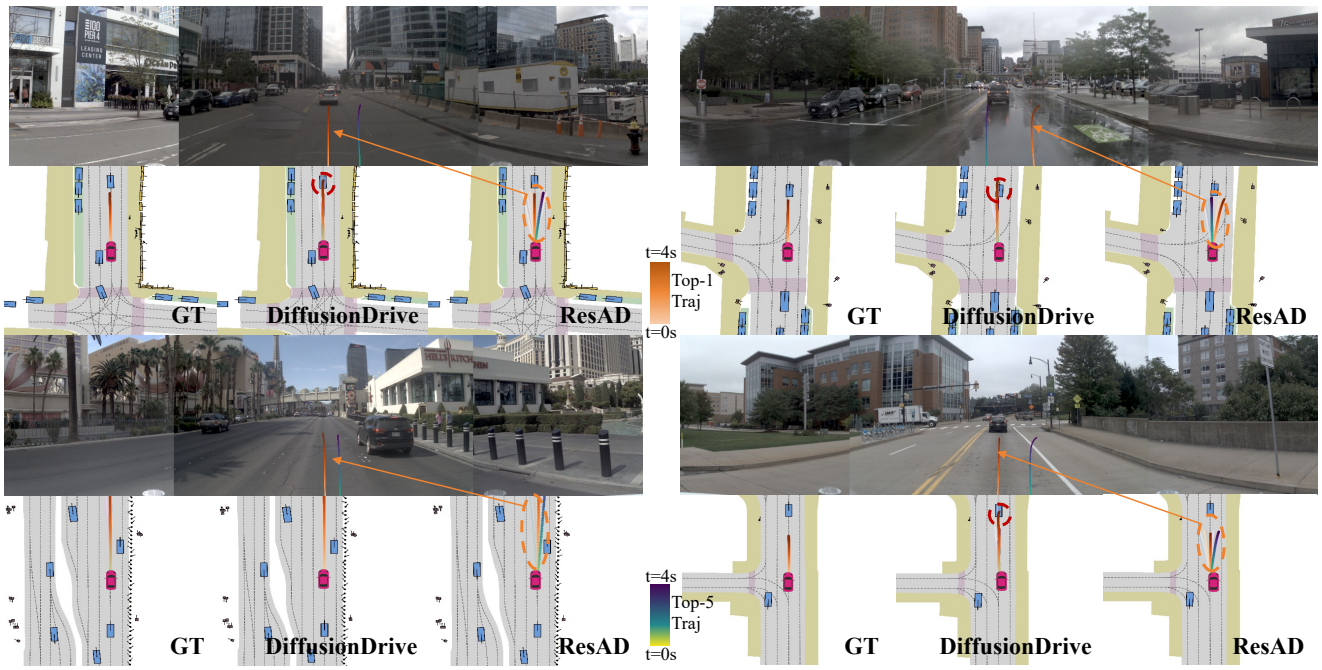


Figure 7. Qualitative comparison of DiffusionDrive, and ***ResAD*** on going straight scenarios of NAVSIM NAVTEST split.

turning left scenarios. As observed, the proposed method can effectively generate multi-modal trajectories to accomplish the left-turn task. Compared to DiffusionDrive, our approach is more attentive to the current scene. It avoids the issue of generating scene-irrelevant trajectories, which can occur when using fixed clustering anchors for the denoising process, as DiffusionDrive does.

**Turning right.** Fig. 9 highlights the top-1 and top-5 scoring trajectories of DiffusionDrive and the proposed ***ResAD*** in turning right scenarios. The proposed method demonstrates its capability to generate diverse multi-modal trajectories for executing a right turn. Unlike DiffusionDrive, which relies on fixed clustering anchors for denoising and thus risks producing scene-irrelevant paths, our method exhibits superior context awareness by avoiding this mechanism.

Furthermore, we have deployed our method, ***ResAD***, on a real-world vehicle. The **real-world demonstration** is available in the project webpage. To mitigate potential privacy concerns, the resolution of these videos has been reduced.

## 8. Supplementary Clarifications

**Rationale behind the CV inertial reference.** We intentionally adopt a simple constant-velocity (CV) inertial reference derived solely from the current ego state. The purpose of this reference is not to approximate the final trajectory as accurately as possible, but to provide a stable and broadly available physical baseline for residual learning. Under this formulation, the reference captures default inertial motion, while the residual absorbs scene-driven control corrections. As a result, steady cruising usually yields small residuals, whereas braking, yielding, turning, or obstacle avoidance induces stronger and more directional deviations. In contrast, a more complex inertial reference may absorb part of the maneuvering behavior into the prior, weakening the residual signal and making it harder for the model to associate scene semantics, such as road curvature or surrounding obstacles, with driving intent. In addition, in our real-world closed-loop experiments (6s@10Hz), we found that directly integrating noisy high-order states, such as acceleration or yaw rate, can accumulate drift and inter-frame jitter over long horizons. The CV model therefore provides a more stable anchor, allowing the diffusion decoder to focus on learning context-aware driving policies. We further compare different inertial reference designs on NAVSIM in Tab. 6, where the CV model achieves the best performance.

| IR Model | Input State | NC↑ | DAC↑ | TTC↑ | Comf.↑ | EP↑ | PDMS↑ |
|---|---|---|---|---|---|---|---|
| CA | $v_0, a_0$ | 97.2 | 95.3 | 92.5 | 100 | 79.9 | 85.4 |
| CTRV | $v_0, \omega_0$ | 96.6 | 94.9 | 90.3 | 100 | 78.3 | 84.3 |
| **CV(Baseline)** | $v_0$ | **98.0** | **97.5** | **94.1** | **100** | **83.3** | **88.8** |

Table 6. Comparison of IR models: CV (const. velocity), CA (const. acceleration), and CTRV (const. speed & yaw rate).

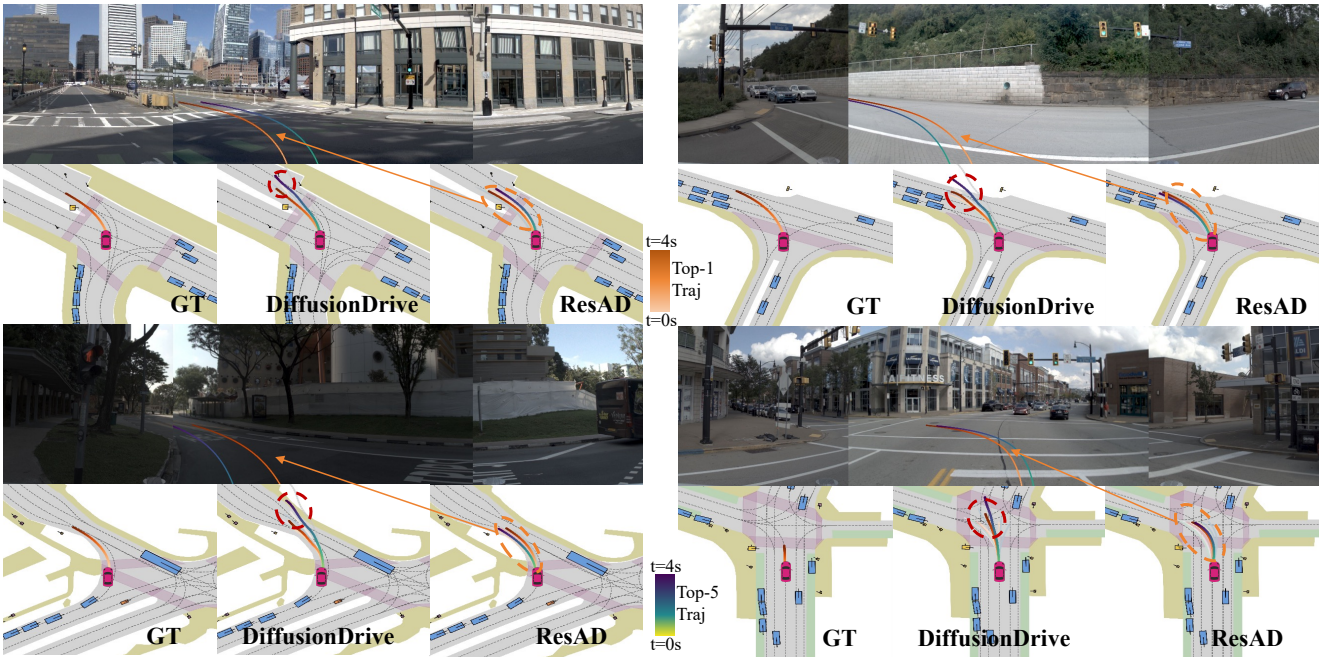


Figure 8. Qualitative comparison of DiffusionDrive, and ***ResAD*** on turning left scenarios of NAVSIM NAVTEST split.

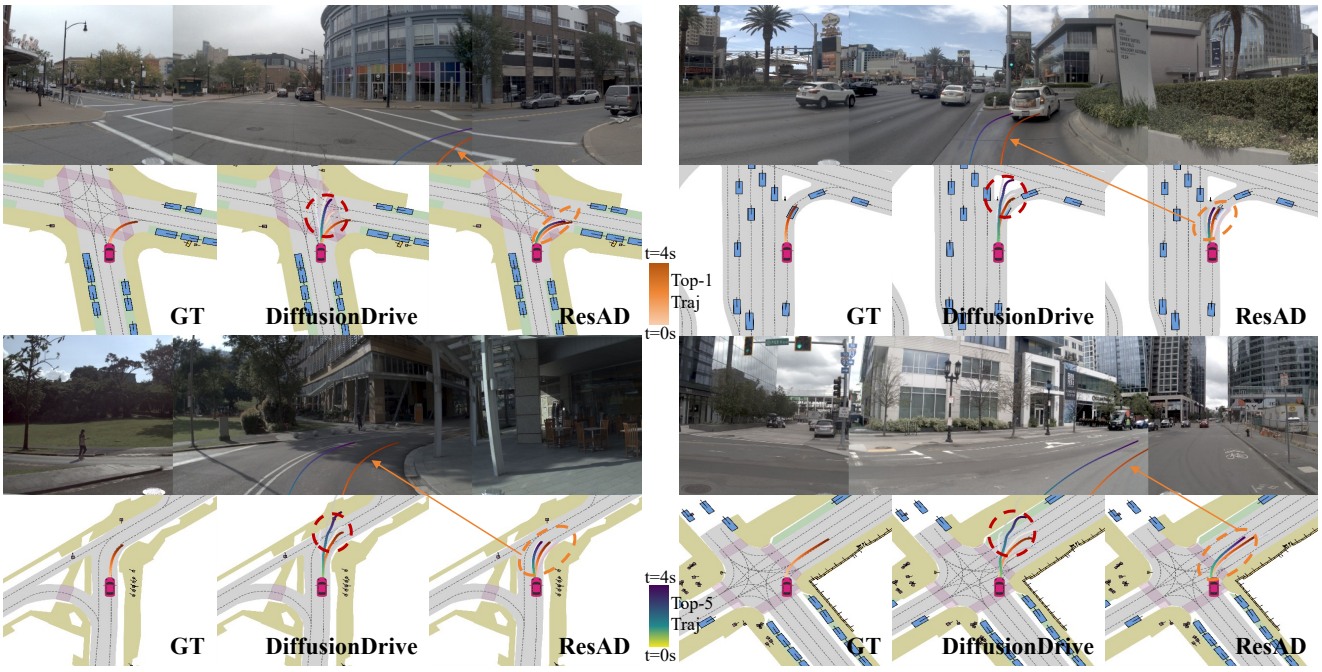


Figure 9. Qualitative comparison of DiffusionDrive, and ***ResAD*** on turning right scenarios of NAVSIM NAVTEST split.