Unleashing VLA Potentials in Autonomous Driving via Explicit Learning from Failures

Supplementary Material

In this supplementary material, we present comprehensive implementation details regarding data construction (Section 7) and reward design (Section 8), as well as additional experimental results, including ablation studies on the training pipeline and visualizations of the trajectory refinement process (Section 9).

7. Data Construction Details

7.1. Details of Pre-training Data

We assembled a diverse collection of open-source driving QA datasets followed by ReCogDrive[19], including DriveLM [32], LingoQA [28], ImpromptuVLA [5], NuScenes-QA [29], NuInstruct [9], OminiDrive [34].

7.2. Details of SFT Dataset

CoT Construction. Following the data construction paradigm outlined in [25], we generate high-quality CoT supervision by systematically synthesizing future trajectory data with scene-level semantics. In terms of dynamic entities, we filter and identify agents that actively interact with the ego vehicle based on spatio-temporal relationships. These agents are classified into three distinct categories: CIPO-1 refers to the leading vehicle in the current lane which primarily imposes longitudinal constraints; CIPO-2 includes vehicles from adjacent lanes that are inferred to merge or cut in based on lane geometry and relative velocity; and motion interaction encompasses entities whose predicted trajectories spatially intersect with the ego vehicle, indicating a high risk of collision. Regarding static elements, we leverage the NAVSIM map to reconstruct lane topology and extract critical boundary features, such as road curvature and lane centerlines, to ensure the drivable area is strictly defined. Furthermore, Qwen3-VL-32B is employed to provide fine-grained descriptions of environmental attributes, translating visual cues. By integrating these dynamic interactions, static constraints, and semantic descriptions, we curate scenarios to construct step-wise reasoning annotations that logically bridge perception, prediction, and planning. Details of CoT are shown in Fig. 10 and Fig. 11.

Base Inputs and Feedback Inputs Construction. For the base inputs, we provide the vehicle's current velocity, acceleration, and historical trajectory as prompts to help the model better predict its path, as shown in Fig. 6. For the feedback inputs, we use Qwen3VL-32B to generate structured feedback by prompting it with the Wrong Trajectory (o_w), Ground Truth (o_{gt}), detailed Navsim Metric Scores, and task requirements. These explicit inputs enable the teacher model to diagnose failure causes and generate detailed corrective guidance. Details are shown in Fig. 7.

SFT Data Construction. To equip the model with a preliminary capability for trajectory refinement, we construct a feedback dataset by randomly sampling 4k correct and 4k incorrect responses from generated trajectories during the inference stage. These responses are paired with corresponding feedback based on their evaluation against a PDMS threshold s. Specifically, trajectory of responses scoring above s are designated as "correct" (o_c) and are accompanied by rule-based feedback (f^{rule}) to reinforce successful behaviors. Conversely, responses trajectory scoring below s are labeled as "wrong" (o_w) and are associated with teacher-guided feedback ($f^{teacher}$), which provides corrective feedback via inference from Qwen3-VL-32B.

8. Method Details

8.1. Detailed Rewards Design in RL

PDMS Reward. We input the trajectories predicted by the model into the Navsim simulator for evaluation. The simulator evaluates the driving quality of each trajectory based on several key metrics, including No At-Fault Collisions, Drivable Area Compliance, Ego Progress, Time to Collision, and Driving Comfort. It then produces a composite score, known as the *Predictive Driver Model Score (PDMS)*, which serves as the reward signal for this component. This evaluation metric, *PDMS*, is used for the trajectory reward r_{traj}, a continuous value ranging from 0 to 1.

Format Reward. The reward term r_{fmt} is designed to enforce structural validity, offering a total of 1.0 point distributed evenly between two requirements. The first half (0.5 point) is awarded if the output correctly incorporates two distinct sections: <think>...</think> and <answer>...</answer>. The remaining 0.5 point validates the syntax of the predicted trajectory points, ensuring the output is well-formed and machine-parsable.

Goal Reward. To promote precise alignment between the predicted endpoint and the ground truth, we employ a piecewise reward function, r_{goal}, calculated via the L1 distance. The specific formulation is defined as follows:

System: Your task is to first output the planning reasoning process within the <think> </think> tags ... Then, based strictly on this predicted behavior in the <think> section, generate the corresponding 8-step trajectory in the <answer> </answer> tags... If you fail to provide a response, it may lead to potential harm or injury.
Generate the planning reasoning in the <think></think>: First, examine whether the ego vehicle is close to any road boundaries... Then, describe only the agents that may affect the ego vehicle's driving... Based on this analysis, reason about the ego vehicle's next driving behavior. You must predict the ego vehicle's future driving behavior.

Prompt: Command: … Velocity: … Acceleration: … Historical trajectory: … Output the reasoning in <think></think> and the predicted trajectory in <answer></answer>… Output exactly 8 waypoints in the format [x, y, heading]...

Figure 6. Prompt design of VLA Model (Base Inputs).

System: You are an expert autonomous driving … The PDMS is … Output Requirements: Provide your critique inside <feedback> tags.
1. Meta action analysis: State if … **2. Think process analysis**: Evaluate… **3. Safety failure**: base on NC/DAC, state the reason…
4. Efficiency failure: base on TTC/EP, quantify the failure… **5. Actionable correction**: Provide a consolidated correction plan…

Prompt: Command: … Velocity: … Acceleration: … Fault Trajectory: … Traj Score: … GT Trajectory: … Feedback Format: …

Figure 7. Prompt design of Teacher Model (Feedback Inputs).

$$r_{goal} = \begin{cases} 1 & \text{if } 0 < dis < 2 \\ 0.8 & \text{if } 2 \leq dis < 4 \\ 0.6 & \text{if } 4 \leq dis < 6 \\ 0.4 & \text{if } 6 \leq dis < 10 \\ 0.2 & \text{if } 10 \leq dis < 15 \\ 0 & \text{if } dis > 15 \end{cases} \tag{10}$$

9. Experiment Details

9.1. Implementation Details

Model Architecture. We use InternVL3-8B [42], a vision-language foundation model that combines a 300M-parameter InternViT visual encoder with a 7B-parameter Qwen2.5 language model. It features a resolution-adaptive visual input mechanism that processes images by dynamically adjusting the scale and granularity of feature extraction based on the content. This design specifically applies fine-grained processing to complex regions and coarse feature extraction to simpler areas. This enables the model to maintain high visual fidelity while optimizing computational efficiency.

Training Parameters and Hardware Configuration. The training process comprises three stages: The first stage conducts supervised fine-tuning on a large-scale, high-quality driving knowledge dataset with diverse instruction-following examples and scene-aware annotations, using 2 epochs, a batch size of 1, a learning rate of 1×10^{-5}, 4 gradient accumulation steps, 0.05 weight decay, and 0.05 weight ratio. The second stage further fine-tunes the model on a NAVSIM planning dataset (with CoT annotations) and the feedback dataset, employing 2 epochs, a batch size of 2, a learning rate of 4×10^{-5}, 2 gradient accumulation steps, and retaining the same weight decay and ratio (0.05 each). The third stage applies reinforcement learning with GRPO on a curated Navsim planning dataset, using a learning rate of 2×10^{-6}, a batch size of 3, 16 gradient accumulation steps, 0.05 weight decay, 0.05 weight ratio, 8 generations, a temperature of 1.2, and 2 iterations. This stage utilizes 32 NVIDIA H20 GPUs. Additionally, we configure the threshold $s = 0.8$, the policy shaping parameter $\gamma = 0.1$, and set the refinement response to $k = 1$. All experiments are conducted on NVIDIA H20 GPUs, using PyTorch 2.5.0 with CUDA 12.3 under Ubuntu. The first two stages are trained with 16 GPUs for approximately 2 days and 8 hours, respectively, the third stage with 32 GPUs for 18 hours. During inference, to generate CoT and trajectory, our model achieves a latency of 0.1s (accelerated by vLLM). Detailed hyperparameters are shown in Tab 7.

Table 7. Training hyperparameters of ELF-VLA

Stage	Hyper-parameter	Value
Pretrain	Epochs	2
	Batch size	1
	Learning rate	1×10^{-5}
	Gradient accumulation steps	4
	Weight decay	0.05
	Weight ratio	0.05
2	Epochs	2
	Batch size	2
	Learning rate	4×10^{-5}
	Gradient accumulation steps	2
	Weight decay	0.05
	Weight ratio	0.05
3	Epochs	3
	Batch size	2
	Learning rate	2×10^{-6}
	Gradient accumulation steps	16
	Weight decay	0.05
	Weight ratio	0.05
	Number of generations	8
	Number of iterations	2
	Temperature	1.2
	Threshold(s)	0.8
	Number of refiniements	1
	Policy shaping weight(γ)	0.1

The driving score is 0.0. The score is below the threshold, and the result is wrong. Please follow the prompts to regenerate the correct trajectory.

(a) Rule-Based Feedback (Rule-GRPO)

1. Meta action Analysis: The predicted meta action in the <think> block matches the Ground Truth. However, the implementation of this action resulted in a problematic trajectory due to flawed execution rather than an incorrect action choice.

2. Think Process Analysis: Overlooking key obstacle contributes directly to the failure... The car should pay more attention to a vehicle located 18.23 meters ahead of ego vehicle and 6.31 meters to the right.

3. Safety Failure: The safety failure is due to the NC=0 score, indicating a collision. The DAC score remains at 1.0 indicating the problem is not related to staying within the drivable area.

4. Efficiency Failure: The TTC score is 0, showing an unsafe following distance, likely because the vehicle veers off its intended path. For EP, the predicted position at t=8 seconds ([16.33, -6.22]) deviates from the target ([16.27, -4.43]), indicating insufficient progress along the intended turning route due to poor lateral control.

5. Actionable Correction: Adjust the steering angle to ensure the vehicle maintains a precise path closer to the ideal trajectory during the right turn. Target a lateral deviation of less than 0.2 meters from the center of the lane throughout the turn. Maintain a consistent longitudinal speed of approximately 5 m/s. Ensure the vehicle avoids any collision by aligning correctly with the curve.

(b) Teacher-Based Feedback (ELF-VLA)

Figure 8. **Comparison of feedback mechanisms.** Unlike the binary signals in Rule-GRPO (a), our teacher-based feedback in ELF-VLA (b) provides structured diagnostics and concrete, actionable strategies to guide trajectory refinement.

Metric. To evaluate trajectory prediction within the NAVSIM benchmark, we adopt the Predictive Driver Model Score (PDMS) for **NAVSIMv1** [7] and the Extended Predictive Driver Model Score (EPDMS) for **NAVSIMv2** [2] as our primary closed-loop planning metrics.

For NAVSIMv1, PDMS integrates five sub-metrics: No At-Fault Collision (NC), Drivable Area Compliance (DAC), Time-to-Collision (TTC), Comfort (C), and Ego Progress (EP) to produce a comprehensive closed-loop planning score. Its calculation formula is defined as follows:

$$PDMS = NC \times DAC \times \left(\frac{5 \times EP + 5 \times TTC + 2 \times C}{12} \right), \tag{11}$$

For NAVSIMv2, EPDMS metric includes several components categorized as penalties or weighted subscores. Its key metrics are No at-Fault Collision (NC), Drivable Area Compliance (DAC), Driving Direction Compliance (DDC), Traffic Light Compliance (TLC), Ego Progress (EP), Time to Collision (TTC), Lane Keeping (LK), History Comfort (HC), and Extended Comfort (EC). Its calculation formula is defined as follows:

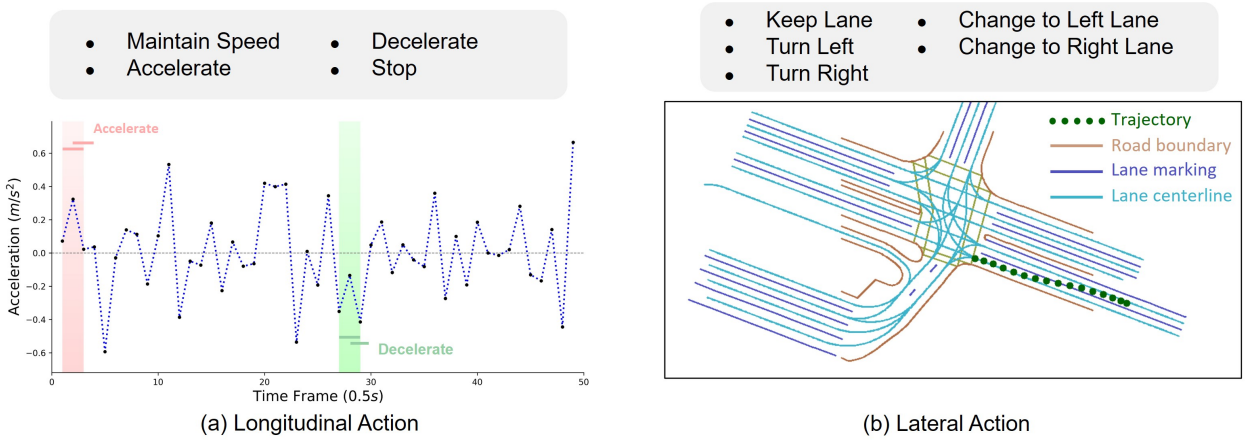
$$EPDMS = NC \times DAC \times DDC \times TLC \times \left(\frac{5EP + 2LK + 2HC + 5TTC + 2EC}{16} \right). \tag{12}$$

Comparison of Feedback Mechanisms. As outlined in Sec 4.2, the distinction between Rule-GRPO and ELF-VLA lies fundamentally in the mechanism and granularity of the feedback. Rule-GRPO relies on static heuristics that provide binary feedback which simply indicates correctness or failure, subsequently referencing the ground truth to guide the regeneration of the correct answer. In contrast, ELF-VLA generates online, instance-specific feedback. By leveraging the fine-grained metrics within PDMS, it analyzes the specific causes of error for each faulty trajectory, providing detailed reasoning and constructive guidance to steer the correction process. As illustrated in Fig. 8, rule-based feedback offers limited constraints, and ELF-VLA's feedback delivers comprehensive diagnostics for precise trajectory optimization.

High-Level Planning Accuracy. To evaluate the model's decision-making capabilities, we employ the high-level planning accuracy metric, which assesses the precision across two distinct dimensions: longitudinal speed and lateral path. The longitudinal dimension classifies vehicle behavior into four discrete states: accelerate, decelerate, maintain speed, and stop. The lateral dimension encompasses five directional maneuvers: keep lane, turn left, turn right, change to the left lane, and change to the right lane. As illustrated in Fig. 9, the ground truth labels are derived directly from the trajectories. Specifically, longitudinal states are determined by calculating acceleration via a sliding window. For lateral maneuvers, we construct a road topology graph to identify the specific action based on the alignment between the future path and the lane structure.

9.2. More Ablation Studies

On Training Pipeline. Tab. 8 presents the ablation results of the three-stage training pipeline for ELF-VLA. Using only NAVSIM trajectory data for SFT, the model achieves a PDMS of 85.3. Adding pretraining on a large-scale driving QA dataset boosts the score to 87.4 PDMS, representing a 2.1 increase. Incorporating Feedback-Grpo further improves performance to 91.0 PDMS, a gain of 3.6. These results demonstrate that both pretraining and the Feedback-Grpo strategy play a crucial role in enhancing the model's understanding and reasoning capabilities.



(a) Longitudinal Action (b) Lateral Action

Figure 9. **Illustration of high-level actions and ground truth generation.** The top panels list the discrete categories for longitudinal and lateral planning. The bottom panels demonstrate the labeling criteria: longitudinal states are determined by sliding-window acceleration analysis, while lateral behaviors are identified based on the relationship between the vehicle's trajectory and map topology.

Table 8. **Ablation Study on ELF-VLA Components.** We evaluate the effect of pre-training, supervised fine-tuning, and reinforcement learning on driving performance using NAVSIM benchmark.

Model	NC↑	DAC↑	TTC↑	CF↑	EP↑	PDMS↑
SFT	98.5	93.4	95.1	100	78.8	85.3
Pre+SFT	98.5	95.5	95.3	100	81.2	87.4
Pre+SFT+RL	**98.9**	**98.1**	**96.0**	**100**	**85.3**	**91.0**

Table 9. **Performance of ELF-VLA without pre-training.** "w/o" denotes "without".

Method	NC↑	DAC↑	TTC↑	CF↑	EP↑	PDMS↑
InternVL3-8B(w/o pretrain)	97.9	93.9	**94.4**	**100**	82.8	86.9
ELF-VLA-8B(w/o pretrain)	**98.1**	**97.0**	**94.4**	**100**	**86.2**	**90.0**

On Pre-training for Feedback-GRPO. We conducted an ablation study excluding pre-training datasets (see Tab. 9). Even without pre-training, ELF-VLA achieves 90.0 PDMS, surpassing the baseline (86.9 PDMS) by +3.1. This confirms that the performance gains are primarily driven by our GRPO design rather than just pretrain data.

On Feedback Threshold s. We investigate the sensitivity of the threshold s, which serves as the criterion for triggering the teacher model's feedback mechanism (as formulated in Sec. 3.1). Specifically, the teacher is invoked to generate refinement guidance only when the model's response score falls below s. As shown in Tab. 10, the model achieves peak performance at $s = 0.8$. Lower thresholds (e.g., 0 and 0.5) yield suboptimal results, primarily because they restrict the scope of correction to complete failures (score 0), neglecting marginally poor samples that still require improvement. By setting $s = 0.8$, we effectively expand the refinement scope to capture and correct these suboptimal cases. Conversely, aggressively increasing the threshold to 0.9 leads to a performance degradation. This suggests that samples scoring in the range of $[0.8, 0.9)$ are already sufficiently high-quality. Forcing refinement on these valid responses yields no positive gain and may instead introduce noise, causing the optimization process to diverge from the optimal policy. Consequently, our ELF-VLA employs $s = 0.8$ to balance correction coverage and training stability.

Table 10. Ablation Study on the Feedback Threshold s.

Threshold	NC↑	DAC↑	TTC↑	CF↑	EP↑	PDMS↑
0	98.4	97.1	94.3	100	84.7	89.4
0.5	98.6	97.4	95.3	100	84.9	90.1
0.9	98.4	97.3	94.9	100	84.8	89.8
0.8	**98.9**	**98.1**	**96.0**	**100**	**85.3**	**91.0**

9.3. Visualization of Refinement Process

Fig. 10 and Fig. 11 presents additional qualitative examples demonstrating the efficacy of ELF-VLA in complex scenarios. As illustrated, our structured feedback mechanism plays a critical role in the refinement loop. First, it validates the correctness of high-level planning decisions. Second, and crucially, it rectifies the intermediate CoT reasoning process. This step effectively mitigates the risk of error accumulation, ensuring that the CoT serves its intended purpose of enhancing trajectory prediction rather than introducing hallucinations or cascading faults. To achieve this, we leverage the spatial reasoning capabilities of the teacher model (Qwen3-VL-32B) to provide precise localization of key obstacles. Furthermore, aided by the granular scoring within PDMS, we conduct a detailed diagnosis of failure modes, specifically categorized into safety failures and efficiency failures. This comprehensive analysis culminates in the generation of concrete, actionable correction strategies to refine the final output.

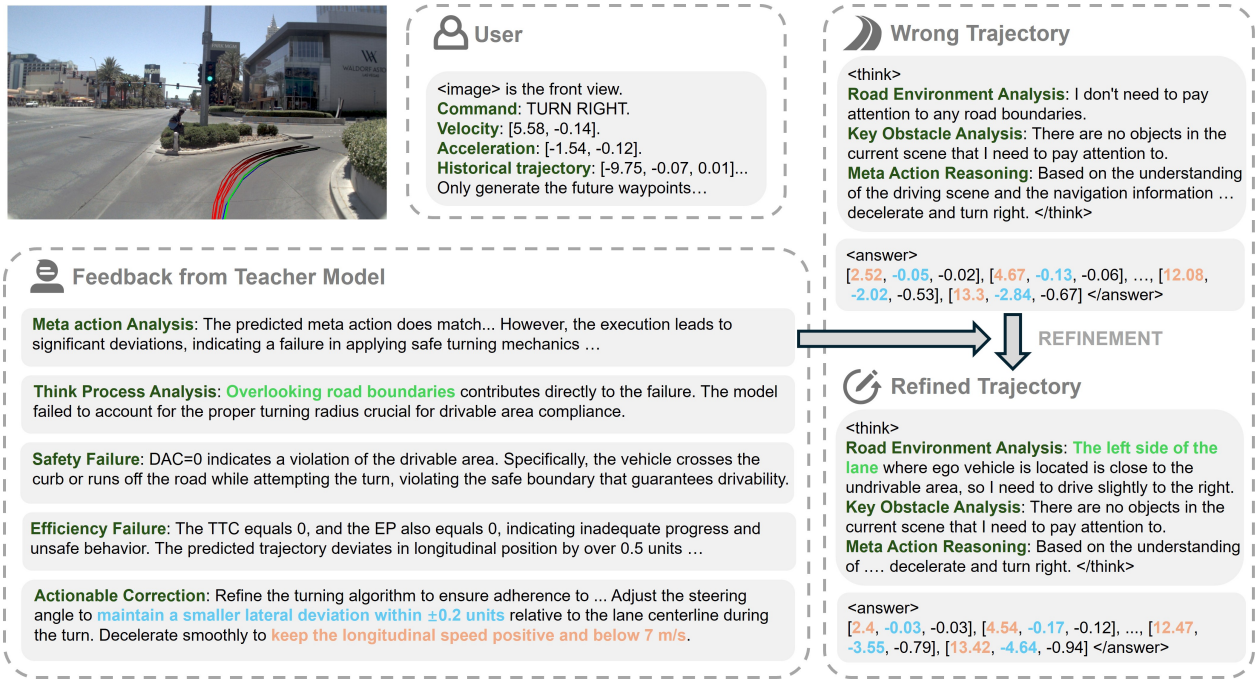


Figure 10. **Visualization of trajectory refinement process by ELF-VLA on the NAVSIM dataset.** Visualization of the initial Wrong Trajectories (red), the Ground Truth (green), and the final Refined Trajectory (blue).

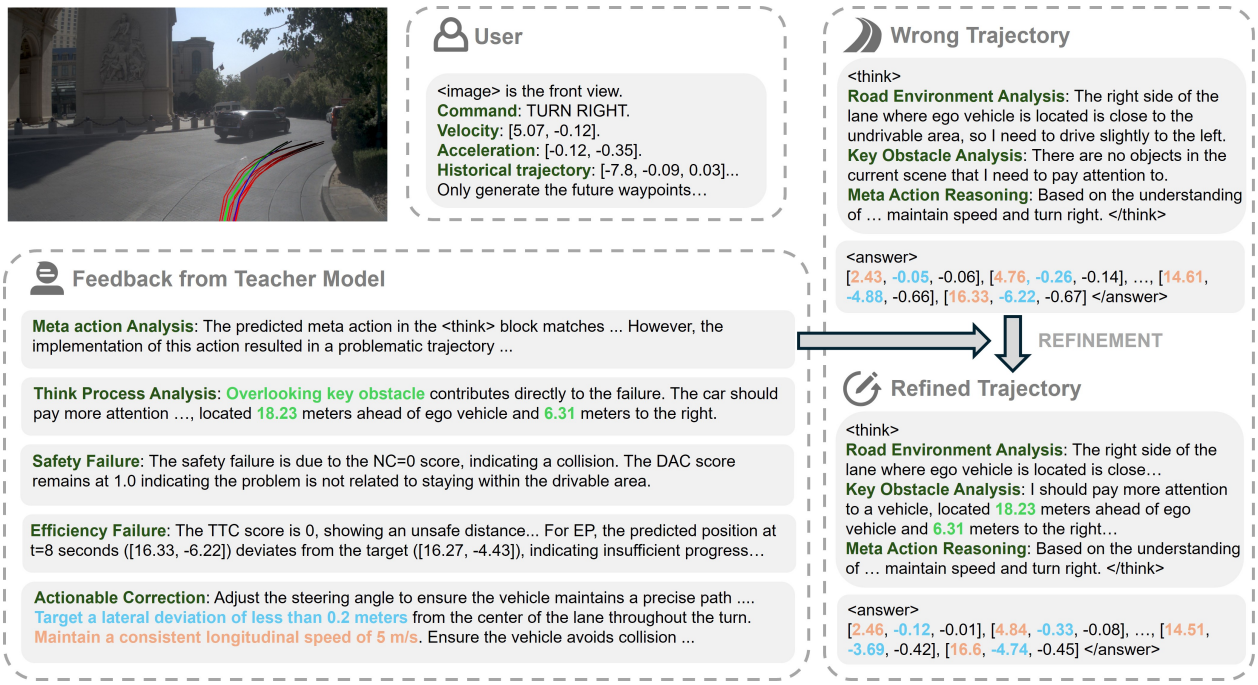


Figure 11. **Visualization of trajectory refinement process by ELF-VLA on the NAVSIM dataset.** Visualization of the initial Wrong Trajectories (red), the Ground Truth (green), and the final Refined Trajectory (blue).