# Appendix

## A. Training Details

Table 1. Hyperparameters for language world model training.

| | Hyperparameter | |
|---|---|---|
| SFT | Global Batch size | 128 |
| | Batch size per GPU | 4 |
| | LoRA rank [2] | 64 |
| | LoRA $\alpha$ | 16 |
| | Epoch | 1 |
| | Learning rate | $1 \times 10^{-5}$ |
| | Weight decay | 0.1 |
| RLVR | Max response length | 1024 |
| | Batch size | 128 |
| | PPO (GRPO) mini batch size | 4 |
| | KL loss coefficient | 0.04 |
| | Group size | 4 |
| | Learning rate | $5 \times 10^{-6}$ |
| Sampling | Top-$p$ | 0.95 |
| | Temperature | 1.2 |
| | Repetition Penalty | 1.2 |

Through an empirical study, we observed that for effective policy model optimization using the GRPO algorithm [1], the responses of the reference model must exhibit sufficient diversity within a limited group size. Accordingly, we increased the repetition penalty and temperature slightly compared to the default settings.

**Algorithm 1** Splitting Dataset into SFT and RFT data for Visual Path Planning

Load entire trajectory metadata list $D$
Compute desired split sizes for:
- SFT vs. RFT (e.g., 4:1 ratio)
- Straight vs. Turn trajectories (e.g., 6:4 or 4:6 depending on SFT/RFT)

For each sample $d \in D$, compute variance of $x$-coordinates over its trajectory
Sort all samples in descending order of $x$-variance
Split into two groups:
- Top-$N$ samples with high variance $\rightarrow$ turning samples
- Remaining samples $\rightarrow$ straight samples

From turning samples:
- Assign the first $N_1$ to RFT-turn
- Assign the remaining $N_2$ to SFT-turn

From straight samples:
- Assign first $M_1$ to SFT-straight
- Assign next $M_2$ to RFT-straight

For each sample in the SFT and RFT sets:
- Convert sample format into instruction-following format with:
  - An image reference
  - A natural language prompt
  - A reasoning + answer pair
- Append to respective output list (SFT or RFT)

Save both lists as JSON files

## B. Split Dataset

### B.1. Dataset Construction and Splitting Strategy

To enable both supervised and reinforcement fine-tuning for visual path planning, we split the dataset into Supervised Fine-Tuning (SFT) and Reinforcement Fine-Tuning (RFT) subsets using a structured and variance-aware strategy. We first load the complete set of trajectory metadata $D$ and determine the target ratios for SFT versus RFT (e.g., 4:1), as well as the distribution of turning and straight trajectories within each subset (e.g., 6:4 or 4:6 depending on the setting). For each trajectory $d \in D$, we calculate the variance of its $x$-coordinates, which serves as a proxy for trajectory curvature. We then sort the trajectories in descending order of $x$-variance. Based on this ordering, we classify the top-$N$ trajectories with high variance as turning trajectories and the remaining as straight trajectories. From these, we allocate subsets to SFT and RFT splits: a portion of turning trajectories to RFT-turn and the remainder to SFT-turn, and similarly, straight trajectories to SFT-straight and RFT-straight. Each selected sample is then converted into an instruction-following format consisting of an image reference, a natural language prompt, and a corresponding reasoning-answer pair. These formatted samples are saved as separate JSON files for the SFT and RFT datasets.

## B.2. Validation Set Construction

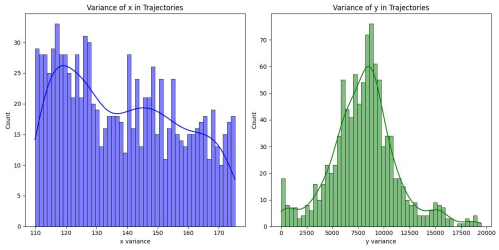


Figure 1. Distribution of trajectories in $D_{easy}$. trajectories exhibit lower x-variance than $D_{hard}$ as can be seen in Fig. 2.

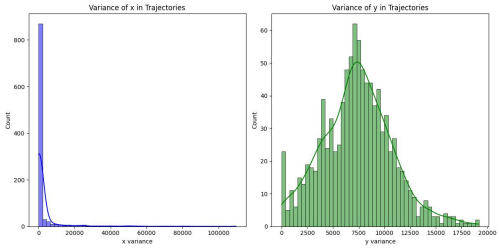


Figure 2. Distribution of trajectories in $D_{hard}$. trajectories exhibit lower x-variance than $D_{easy}$ as can be seen in Fig. 1.

For evaluation, we construct validation sets focused on moderate and hard scenarios. We begin with a dense set of validation trajectories $D_{\text{dense}}$ about 12K samples and remove any overlap with standard annotations $D_{\text{standard}}$ to obtain candidate validation samples $D_{\text{val}}$. We compute the $x$-variance for each sample and sort them accordingly. To form the easy set $D_{\text{easy}}$, we select a middle slice (e.g., 1K samples centered around the median $x$-variance). For the hard set $D_{\text{hard}}$, we randomly sample 0.7K trajectories from the top 70% (high-variance) and 0.3K from the bottom 10% (low-variance) of the sorted list. Both validation subsets are saved as JSON files for downstream evaluation.

Table 2. Summary of Validation Subsets Construction

| Subset | Sampling Strategy | #Trajectories |
|---|---|---|
| Easy ($D_{\text{easy}}$) | Median-centered 1K slice of $x$-variance | 1K |
| Hard ($D_{\text{hard}}$) | 0.7K from top 70% + 0.3K from bottom 10% | 1K |

**Algorithm 2** Construct Validation Sets (Easy, Hard)

Load dense validation trajectories $D_{dense}$
Load standard validation annotations $D_{standard}$
$D_{val} \leftarrow D_{dense} \setminus D_{standard}$
For each $d \in D_{val}$, compute variance of $x$ over trajectory
$D_{sorted} \leftarrow$ sort $D_{val}$ by $x$-variance (descending)
$N \leftarrow |D_{sorted}|$
**Easy Set:** $D_{easy} \leftarrow D_{sorted}[N/2 - 500 : N/2 + 500]$
**Hard Set:**
Randomly sample 700 from top 70% of $D_{sorted}$
Randomly sample 300 from bottom 10% of $D_{sorted}$
$D_{hard} \leftarrow$ combined samples
Save $D_{easy}$ and $D_{hard}$ as JSON

## C. Pseudo code of OOD Evaluation

**Algorithm 3** Procedure for Out-of-Distribution evaluation with as pseudo code.

**Require:** Dataset $\mathcal{D} = \{(I_i, B_i)\}_{i=1}^{N}$ where $I_i$: image, $B_i$: 2D bounding boxes
**Require:** Model $\mathcal{M}$ predicting trajectory $\mathcal{T} = \{(x_j, y_j)\}_{j=1}^{20}$
**Ensure:** Fail Rate, Avg 2D BBox Collision, Avg Penetration Length
**Initialize:** $C_{fail} \leftarrow 0$, $C_{bbox} \leftarrow 0$, $L_{pen} \leftarrow 0$, $N \leftarrow 0$, $T_{bbox} \leftarrow 0$
**for** each $(I, B)$ in $\mathcal{D}$ **do**
  $\mathcal{T} \leftarrow \mathcal{M}(I)$ ▷ 20-point trajectory
  $traj \leftarrow$ LineString($\mathcal{T}$), $c \leftarrow 0$
  **for** each $b \in B$ **do**
    $p \leftarrow$ Polygon($b$)
    **if** $traj$.intersects($p$) **then**
      $c \leftarrow c + 1$
      $L_{pen} \leftarrow L_{pen} +$ length$(traj \cap p)$
    **end if**
    $T_{bbox} \leftarrow T_{bbox} + 1$
  **end for**
  **if** $c > 0$ **then** $C_{fail} \leftarrow C_{fail} + 1$
  **end if**
  $C_{bbox} \leftarrow C_{bbox} + c$, $N \leftarrow N + 1$
**end for**
**return** $\frac{C_{fail}}{N}$, $\frac{C_{bbox}}{N}$, $\frac{L_{pen}}{N}$

**Trajectory Fail Rate:** The proportion of predicted trajectories that intersect with at least one 2D bounding box in the scene:

$$\text{Fail Rate} = \frac{\text{Number of collision trajectories}}{\text{Total number of samples}} \quad (1)$$

**Average 2D BBox Collision:** The average number of

bounding boxes that each predicted trajectory collides with:

$$\text{Collision Count} = \frac{\text{Total bbox collisions}}{\text{Total number of samples}} \tag{2}$$

**Penetration Length:** The average geometric length of trajectory segments that penetrate into bounding boxes:

$$\text{Avg Penetration Length} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{|B_i|} \text{length}(\mathcal{T}_i \cap bbox_j)}{\text{Total number of samples}} \tag{3}$$

where $\mathcal{T}_i \cap bbox_j$ represents the intersection between trajectory $\mathcal{T}_i$ and bounding box $bbox_j$.

## D. Qualitative Results on OOD Benchmark

We qualitatively evaluate the proposed method on CODA-LM, a zero-shot dataset composed of diverse corner cases involving out-of-distribution (OOD) road hazards. Since CODA-LM does not provide ground-truth trajectories, we rely on visual inspection of the predicted paths to assess plausibility, safety, and contextual awareness. As shown in Fig. 3, the model fine-tuned with RLVR produces more realistic and contextually appropriate trajectories compared to the one trained via supervised fine-tuning. In scenarios involving roadwork, blocked lanes, and ambiguous path constraints, the supervised model often generates overly linear or risk-prone trajectories that lack appropriate deviation or caution. In contrast, the proposed method consistently generates smoother, safer, and more context-aware trajectories, often adjusting path curvature to avoid obstacles such as cones, barriers, and vehicles. These results demonstrate the model's capacity to generalize to previously unseen situations by aligning its reasoning and path generation with latent planning cues in the scene, even without explicit supervision. While the absence of quantitative metrics in CODA-LM limits numerical evaluation, these qualitative outcomes suggest that reinforcement fine-tuning enhances the model's ability to adapt to driving contexts.

## References


[1] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 1

[2] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 1

Figure 3. Qualitative results comparing supervised fine-tuning (top) and RLVR-based reinforcement fine-tuning (bottom) across diverse out-of-distribution (OOD) hazard scenarios. Examples include construction zones, adverse weather (e.g., rain and poor lighting), unexpected pedestrian behavior, and road obstacles. The proposed method generates more accurate and context-aware trajectories under complex conditions, indicating better robustness in real-world hazard cases.