SimLingo: Vision-Only Closed-Loop Autonomous Driving with Language-Action Alignment

Supplementary Material

A. Datasets

We provide a more detailed description of the collected datasets and how we generate the labels for each language-related dataset.

A.1 Driving dataset - Scenarios

In each Town, we collect data containing different scenarios, which we detail in the following (descriptions are taken from https://leaderboard.carla.org/scenarios/):

- **Control Loss without Previous Action:** The ego-vehicle loses control due to poor road conditions and must recover.
- **Unprotected Left Turn at Intersection with Oncoming Traffic:** The ego-vehicle performs an unprotected left turn at an intersection (can occur at both signalized and unsignalized intersections).
- **Right Turn at Intersection with Crossing Traffic:** The ego-vehicle makes a right turn at an intersection while yielding to crossing traffic (signalized and unsignalized intersections).
- **Crossing Negotiation at Unsignalized Intersection:** The ego-vehicle navigates an unsignalized intersection by negotiating with other vehicles. The assumption is that the vehicle entering the intersection first has priority.
- **Crossing Traffic Running a Red Light at an Intersection:** While the ego-vehicle is traveling straight through an intersection, a crossing vehicle runs a red light (signalized and unsignalized intersections).
- **Crossing with Oncoming Bicycles:** The ego-vehicle must turn at an intersection while yielding to bicycles crossing the intersection.
- **Highway Merge from On-Ramp:** The ego-vehicle merges into moving traffic on a highway.
- **Highway Cut-In from On-Ramp:** A vehicle merges into the ego-vehicle's lane from an on-ramp. The ego-vehicle must decelerate, brake, or change lanes to avoid a collision.
- **Static Cut-In:** Another vehicle cuts into the ego lane from a queue of stationary traffic. It must decelerate, brake, or change lanes to avoid a collision.
- **Highway Exit:** To exit the highway the ego-vehicle needs to cross a lane of moving traffic.
- **Yield to Emergency Vehicle:** An emergency vehicle is approaching from behind. The ego must create space for it to pass safely.
- **Obstacle in Lane - Same Direction:** An obstacle (e.g., a construction zone, an accident, or a parked vehicle) is blocking the ego lane. The ego vehicle must change lanes into traffic moving in the same direction to bypass the obstacle.
- **Obstacle in Lane - Opposite Direction:** An obstacle (e.g., construction zone, an accident, or a parked vehicle) is blocking the ego lane. The ego vehicle must change lanes into traffic moving in the opposite direction to bypass the obstacle.
- **Door obstacle:** The ego-vehicle needs to avoid a parked vehicle with its door opening into the lane.
- **Slow moving hazard at lane edge:** A slow-moving hazard (e.g. bicycle) partially obstructs the ego vehicle's lane. The ego-vehicle must either brake or carefully bypass the hazard (bypassing on lane with traffic in the same or opposite direction).
- **Vehicle invading lane on bend:** On a bend, an oncoming vehicle invades the ego vehicle's lane to avoid an obstacle. The ego-vehicle must brake or move to the side of the road to safely navigate past the oncoming vehicle.
- **Longitudinal control after leading vehicle's brake:** The leading vehicle in front of the ego-vehicle brakes suddenly to avoid an obstacle. The ego-vehicle must execute an emergency brake or avoidance maneuver to prevent a collision.
- **Obstacle avoidance without prior action:** The ego-vehicle encounters an unexpected obstacle or entity on the road. It must perform an emergency brake or avoidance maneuver.
- **Pedestrian emerging from behind parked vehicle:** A pedestrian suddenly emerges from behind a parked vehicle and enters the lane. The ego-vehicle must brake or take evasive action to avoid hitting the pedestrian.
- **Obstacle avoidance with prior action - pedestrian or bicycle:** While in the middle of a turn, the ego-vehicle encounters an obstacle such as a pedestrian or a bicycle crossing the road or a stopped vehicle in the road and must perform an emergency brake or avoidance maneuver.
- **Parking Cut-in:** A parked vehicle exits a parallel parking space into the ego-vehicle's path. The ego-vehicle must slow down to allow the parked vehicle to merge into traffic.
- **Parking Exit:** The ego-vehicle must exit a parallel

parking space and merge into moving traffic.

A.2 VQA - DriveLM

For the VQA data, we use the DriveLM-Carla [7] data generation method. Since we generate a new driving dataset, we extract questions and answers for our dataset instead of using the original dataset. For the training set, we generate in total 28M QA-pairs for 1M frames of Town 12. For the evaluation set, we use the keyframe extraction of DriveLM to evaluate on more interesting and less redundant frames. In addition, we balance the validation set to capture the same amount of samples for each answer type in the dataset.

Since the labels are auto-generated with a heuristic-based procedure, the QAs follow the same sentence structures and wordings. To avoid overfitting to specific phrases we include data augmentation. For this, we prompt GPT-4 to generate 20 alternative sentences for each question and answer, from which we sample during data loading.

A.3 Commentary

We generate language labels for the *Commentary* task based on a subset of the simulator state, which we save during the data collection. The structure of the *Commentary* labels is as follows: The first sentence describes the action according to the route with its justification (e.g., staying on the current lane, doing a lane change to go around an obstacle). It is followed by a description of the speed-related action (e.g., accelerate, keep the speed, stop), and the reason (e.g., because of the pedestrian, to follow the vehicle in front, to drive closer to the stop sign).

In the following, we detail the steps to obtain each part of the *Commentary* labels.

Route action. The default description is "*Follow the route.*". Only in special cases, we change the description. For this, we check if any scenario is active in the given frame and get the scenario type. We only change the default description for the scenarios requiring a deviation from the center lane of the original route (e.g., obstacle in lane, vehicle invading the lane, door obstacle). Since the ego action differs depending on the location relative to the scenario, we extract the relative positioning from the simulator information. Those locations are grouped into three phases: (1) before the lane deviation, (2) during the deviation, and (3) end of the deviation. *Before the deviation*, we differentiate between the scenario types and use a template sentence for each, for instance:

- Overtake the bikes on your lane.
- Go around the vehicle with the open door.
- Give way to the emergency vehicle.
- Go around the accident in your lane.
- Go around the construction site.
- Move slightly to the right to circumvent the oncoming cars entering your lane because of the construction cones.

During the deviation, describes the phase in which the ego already shifted lanes. We reuse the templates from (1) but add "Stay on your current lane to" before the templates (e.g., Stay on your current lane to overtake the bikes on your lane.)

End of the deviation is the phase where the ego vehicle needs to shift back to its original lane. Since our model is based on only front-view cameras we use a generic sentence for this (i.e., "Return to your original route after avoiding the obstacle.") as often the type of the obstacle or scenario is not visible anymore. This template can be easily changed for a model supporting multi-view inputs.

Speed action. We generate a high-level description of the ego action based on the current speed, the desired target speed based on the expert decision, and the current speed limit. We differentiate between the following types:

- Remain stopped
- Come to a stop now
- Maintain your current speed
- Maintain the reduced speed
- Increase your speed
- Slow down

For the scenarios that are used for the *route action* description (i.e., where the expert needs to deviate from the route), we use a different sentence template. This is only the case for the situation when the ego vehicle is *before the deviation* and is stopped and remains stopped for the next two seconds. In this case, we use the template "Wait for a gap in the traffic before changing lanes".

Speed Reason. Next, we leverage the Inteligent-Driver-Model (IDM) features, which the expert algorithm is based on. IDM identifies leading objects and calculates the optimal target speed for the ego vehicle based on the distance to the leading object. Leading objects include dynamic objects like other vehicles or pedestrians and static objects like traffic lights, stop signs, or construction sites. With this, we know for any sample in the dataset which object the main reason is for the given target speed and therefore the *speed action*. Based on the type of the leading object, we construct a language description of the object. For vehicles, this consists of the color of the vehicle, the type (e.g., SUV, police car), and a rough position relative to the ego (e.g., to the front right). For static objects, it is the name of the object (e.g. traffic light) and in case the object has a state this is also included (e.g., *red* traffic light). For pedestrians, we differentiate between children and adults. Based on the type of the leading object and the *speed action* we construct different sentences, for instance:

- since you've already stopped at the stop sign
- to avoid a collision with the *object description*

- due to the pedestrian crossing in front of you
- to remain behind the red SUV that is slowing down because of a red traffic light.
- to reach the target speed
- because the traffic light is green

If we are right before a junction we add another notice label regarding the positioning of the other vehicles in the junction. With this, the model needs to reason about whether it is safe to enter a junction or not. We collect the position and driving direction of each vehicle close to the junction and summarize the situation based on one of the following sentences:

- the other vehicles are stopped at the junction and the junction is clear
- the other vehicles are stopped at the junction and the vehicle in the junction is moving away
- pay attention to the vehicles coming towards the junction
- pay attention to the vehicle in the junction

A.4 Action Dreamer

We construct an offline, non-reactive simulation based on the collected dataset to generate alternative ego trajectories and evaluate their feasibility in terms of collision avoidance and adherence to traffic rules. For this purpose, we utilize the Kinematic Bicycle Model in combination with the PID controllers from the PDM-lite expert algorithm.

The core functionality of the Action Dreamer simulation is the *ego forecasting*, which predicts future ego vehicle poses based on the ego actions in each timestep. There are several approaches to generate these ego actions, allowing for modification to obtain the alternative trajectories. One approach involves perturbing the ground truth actions to produce slightly modified trajectories. Another approach uses the PID controllers to compute actions based on pre-defined path waypoints and target speeds. In this case, the lateral PID controller generates steering angles, while the longitudinal PID determines acceleration and braking values based on the desired target speed. Using these actions, the Kinematic Bicycle Model calculates the next vehicle pose. This process can be iteratively unrolled over multiple time steps to derive a complete trajectory.

We start with obtaining the current state of the simulator from the saved dataset. For each dynamic object, we also get the states for the following 10 timesteps. With this we can get the non-reactive trajectories for each object and perform collision checks with the ego vehicle. As default, we use the ground truth actions of the ego vehicle. In addition, we use the ground truth path provided by the experts' path planner, which includes waypoints spaced every 10 cm, as the default route to be followed. We then change those default values to obtain alternative trajectories for the modes: objects (collision), faster, slower, target speed, and lane changes.

The following steps describe how we obtain the necessary information (e.g., actions, updated paths with desired speeds, or a combination of both) for the ego forecasting method for each of the different modes we have.

- **Objects (Collision) Mode:** Filter all dynamic and static objects, retaining only those within a 15-meter radius of the ego route and at least 3 meters ahead of the ego vehicle. For each object, calculate its position, distance to the ego vehicle, and whether the ego vehicle (given its current speed) could reach the object within a given future timestep. Objects that are too far and unreachable are discarded. For reachable objects, we calculate the target speed required for the ego vehicle to precisely reach the object's location in the required time. For the path, we adjust the route waypoints to ensure they intersect with the target object's position. This modified route, along with the computed target speed, is then passed to the ego forecasting process.
- **Faster Mode:** We use the original path and actions for steering but set the acceleration to a random value above 50%.
- **Slower Mode:** Similar to the faster mode but we set the acceleration to zero and activate the brake.
- **Random Target Speed:** We assign a random target speed between 0 and 35 m/s and directly pass this together with the default path to the forecasting method.
- **Lane Changes:** We exclude frames where the vehicle is already performing a lane change or is in a junction. To obtain the number of possible options, we extract information on the number and types of lanes (e.g., driving lanes, parking lanes, or sidewalks) in both the same and opposite directions. For each of the options, we change the default path so that it reaches the specified lane. We randomize the starting distance of the lane change and the length of the transition phase. Those parameters are conditioned on the current ego speed.

B. Implementation details

B.1 Hyperparameter

Table 1 shows the hyperparameter we use to train SimLingo.

B.2 Training buckets

The majority of driving involves straight, uneventful segments. To maximize the collection of interesting scenarios during data collection, we focus on capturing a diverse range of challenging situations. However, some ratio of

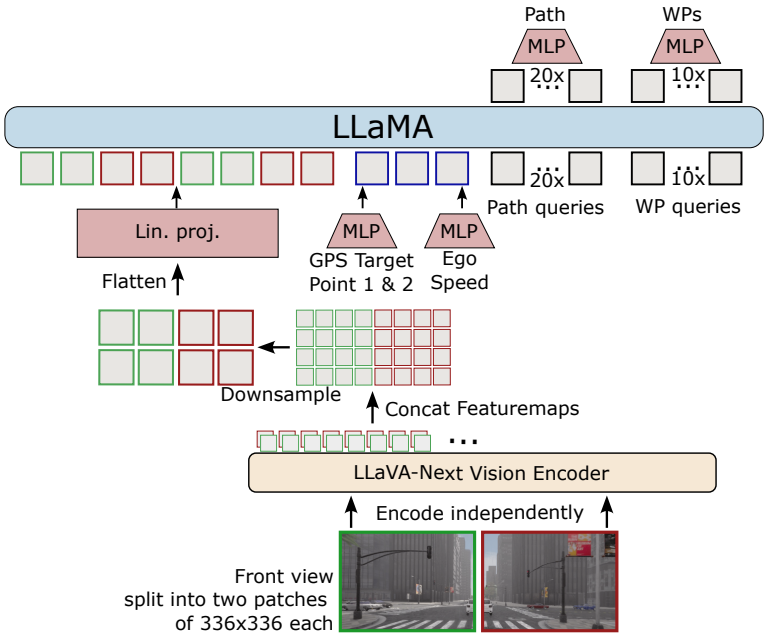

Figure 1. **SimLingo-BASE architecture.** The images are split in two, and each split is independently encoded and then concatenated, downsampled, and projected before feeding it into a transformer decoder which is based on the LLaMA architecture. The output utilizes the same output representation as SimLingo.

Epochs	14
Learning Rate	3e-5
Batch Size	96
Optimizer	AdamW
Weight decay	0.1
Betas	(0.9, 0.999)
LR schduler	One cycle cosine
Warmup steps	5% of total steps
LoRA alpha	64
LoRA r	32
LoRA dropout	0.1

Table 1. **Hyperparameter** choices to train SimLingo.

easy and uneventful data is inevitable. Training models on the entire dataset revealed that straight driving without hazards is effectively learned in the early epochs, resulting in wasted computation in later epochs as the models continue to train on these uninteresting samples. To address this issue, we create data buckets containing only the interesting samples and sample from these buckets during training instead of the entire dataset. We use (1) five buckets for different amounts of acceleration and deceleration with one specifically for starting from stop, excluding samples with an acceleration between -1 and 1, (2) two buckets for steering, excluding samples for going straight, (3) three buckets for vehicle hazard with vehicles coming from different directions, (4) one for a stop sign, red light, and walker hazards each, (5) one bucket for swerving around obstacles, (6) one bucket for "old" Towns commonly used in Leaderboard 1.0 models (Town 01-10) and (7) one bucket that samples from the whole dataset to keep a small portion of uneventful data such as driving straight.

B.3 SimLingo vs. SimLingo-BASE

Our base model SimLingo-BASE was designed as a lightweight model to research driving-specific design choices. With adding language capabilities, we also changed some of the settings to better match the added requirements. We note that because of the closure of the CARLA Leaderboard and because of computational overhead we did not repeat the experiments of the SimLingo-BASE with the new settings. We believe that the claims and results can be expected to still hold. Fig. 1 shows the architecture of SimLingo-BASE. We detail the exact differences between SimLingo-BASE and SimLingo:

1. Number of epochs: SimLingo-BASE is trained for 30 epochs. For SimLingo we reduce the number of epochs to 14.
2. Image encoder: SimLingo-BASE uses a Clip-ViT that is used as default in the LLaVA VLM. SimLingo uses the original image encoder of the InternVL2-1B. In both cases, we do full finetuning.
3. Language model: SimLingo-BASE uses a 50M parameter transformer decoder based on the LLaMA architecture which we train from scratch. SimLingo uses the default pretrained LLM from the InternVL2-1B model which we finetune with LoRA.

4. Loss function: SimLingo-BASE uses L2-Loss. After adding the additional Action Dreaming data we observed instabilities during training with the L2-Loss so we changed to the SmoothL1-Loss.

B.4 Metric descriptions

Leaderboard 2.0.
The Leaderboard uses the official CARLA metrics: Driving Score, Route Completion, and Infraction Score. Each metric is calculated for each route independently. After all routes are completed, the final metrics are derived by taking the arithmetic mean of the metrics across all routes. The overall driving score, calculated using the global values, is the primary metric for ranking methods.
Driving Score. The primary evaluation criterion is the *Driving Score*, denoted as:

$$DS_i = RC_i \cdot IS_i,$$

where RC_i represents the percentage of the i-th route completed, and IS_i is a penalty factor accounting for infractions incurred during the route.
Route Completion. This metric quantifies the proportion of the route successfully completed by the agent, expressed as a percentage.
Infraction Penalty. The penalty due to infractions, IS_i, is calculated as a product of all infractions:

$$IS_i = \prod_{j=1}^{N_I} (p_j)^{\#\text{infractions}_j},$$

where p_j denotes the penalty coefficient for the j-th type of infraction out of a total of N_I infraction types, which we specify int following. $\#\text{infractions}_j$ is the number of times this infraction occurred. The calculation begins with a base score of 1.0, which decreases with each infraction.
Infractions are categorized by severity, each associated with a penalty coefficient that reduces the driving score. Key infractions include:

- **Collisions with pedestrians:** $p_j = 0.50$.
- **Collisions with vehicles:** $p_j = 0.60$.
- **Collisions with static objects:** $p_j = 0.65$.
- **Running a red light:** $p_j = 0.70$.
- **Ignoring a stop sign:** $p_j = 0.80$.
- **Failure to yield to emergency vehicles:** $p_j = 0.7$.
- **Failure to maintain minimum speed:** Up to $p_j = 0.7$.
- **Off-road driving:** Reduces route completion score proportionally.

When one of the following events occurs, the route stops immediately:

- Route deviation (more than 30 meters off route).
- Blocked agent (more than 180 simulation seconds without action).
- Communication timeout (more than 60 seconds).
- Route timeout (exceeding allowed simulation time).

Leaderboard 2.0 metric discussion.
The Driving Score is calculated in a way, that it can be advantageous not completing the whole route. This is the case if the infractions incurred during a segment of the route reduce the driving score more than the potential gain from continuing the route. In this case stopping early to avoid further penalties leads to an overall higher driving score. This tradeoff only occurs for long routes. We refer to [11] for a mathematical description of this tradeoff, a detailed discussion and a proposal for a better metric calculation.
Bench2Drive.
Driving Score. The Driving Score is calculated similarly to the Leaderboard 2.0. The only difference to the original Driving Score is that the penalty "Failure to maintain minimum speed" is ignored. To take the driving speed into account the authors of the benchmark introduced the "Efficiency" metric. Since Bench2Drive uses short routes the discussed trade-off does not occur on this benchmark.
Success Rate. The Success Rate measures the percentage of completed routed without any infractions (ignoring the minimum speed penalty).
Efficiency. This uses the ratio of the ego vehicle speed to the speed of the surrounding actors. Since there was no penalty in Leaderboard 1.0 for low speeds most models used a very low speed, which makes driving and reacting to other dynamic actors much easier. The higher this efficiency metric the faster the model drives, making the driving task harder.
Comfortness. The comfortness metric takes the jerk magnitude, lateral and longitudinal acceleration, yaw acceleration, longitudinal jerk, and the yaw rate into account. If the mean of the ego vehicles measurement over the full route falls into the following thresholds it is treated as success.

- **Jerk Magnitude:** Maximum absolute magnitude of jerk is $8.37\,\text{m/s}^3$.
- **Lateral Acceleration:** Maximum absolute lateral acceleration is $4.89\,\text{m/s}^2$.
- **Longitudinal Acceleration:**
  - Maximum longitudinal acceleration is $2.40\,\text{m/s}^2$.
  - Minimum longitudinal acceleration is $-4.05\,\text{m/s}^2$.
- **Yaw Acceleration:** Maximum absolute yaw acceleration is $1.93\,\text{rad/s}^2$.
- **Longitudinal Jerk:** Maximum absolute longitudinal jerk is $4.13\,\text{m/s}^3$.
- **Yaw Rate:** Maximum absolute yaw rate is $0.95\,\text{rad/s}$.

The thresholds are taken from the Bench2Drive repository.
Vision-Language Understanding.

	Method	Ability (%) ↑					
		Merging	Overtaking	Emergency Brake	Give Way	Traffic Sign	**Mean**
w/ dist.	TCP [9]	16.18	20.00	20.00	10.00	6.99	14.63
	TCP-ctrl	10.29	4.44	10.00	10.00	6.45	8.23
	TCP-traj	8.89	24.29	51.67	40.00	46.28	28.51
	ThinkTwice [4]	27.38	18.42	35.82	50.00	54.23	37.17
	DriveAdapter [3]	28.82	26.38	48.76	50.00	56.43	42.08
w/o dist.	AD-MLP [10]	0.00	0.00	0.00	0.00	4.35	0.87
	UniAD-Tiny [2]	8.89	9.33	20.00	20.00	15.43	14.73
	UniAD-Base [2]	14.10	17.78	21.67	10.00	14.21	15.55
	VAD [5]	8.11	24.44	18.64	20.00	19.15	18.07
	TCP-traj w/o distillation	17.14	6.67	40.00	50.00	28.72	28.51
	SimLingo-BASE (LB2.0 model)	**60.00**	**60.00**	78.33	50.00	77.89	65.25
	SimLingo	54.01±2.63	57.04±3.40	**88.33**±3.34	**53.33**±5.77	**82.45**±4.73	**67.03**±2.12

Table 2. **Multi-Ability Results of Bench2Drive.** We outperform the existing methods in all abilities and can improve in average by 25 percentage points.

	DS ↑	Stat ↓
WPs	3.21	0.68
+Path	4.49	0.0

(a) **Output.**

	DS ↑
Clip ViT	6.87
w/o pretr.	0.45
Resnet-34	2.71

(b) **Vision encoder**.

	DS ↑
1300	3.93
1800	4.49
2100	6.87
2400	6.35

(c) **Early stopping**.

Table 3. Ablations of different parts of SimLingo-BASE, showcasing the superiority of the disentangled output representation and the large impact of the correct threshold for early stopping. The score of the default configuration is highlighted in gray. All numbers are official Leaderboard 2.0 scores.

For the tasks of VQA and Commentary, we use SPICE [1] and DriveLM's GPT Score [7] as metrics. SPICE is a metric used for image captioning with a higher correlation to human judgment than other automatic language metrics like Cider [8] or Meteor [6]. The GPT Score is based on the DriveLM implementation with two smaller changes. Since we could not directly compare to their numbers, because of a different evaluation set those changes should not have any impact on the conclusions drawn. The first change is using GPT-4 (gpt-4o-2024-08-06) instead of GPT-3.5. In addition, we add "Just rate the similarity of the content not the sentence structure. If the content is completely different rate with 0." to the prompt as we found this to be more accurate.

Action Dreaming.

For the Action Dreaming evaluation, we use Success Rate as the metric. Each category has its own definition of success, which we detail in the following:

- **Slow down**: We calculate the target speed for each waypoint of the predicted speed waypoints. Those target speeds represent the target speeds for future timesteps. We do linear regression to get the slope. Success is defined as the slope being smaller than $-0.05 * v$, with v being the current ego speed.
- **Speed up**: Same calculation as for *Slow down* but we use $slope > 0.05 * v$.
- **Target Speed**: Since we do not know if the target speed can be reached in the prediction horizon of the waypoints we compare the predictions with the ground truth actions instead of directly comparing to the target speed. We use two rules defining success: First, if the predicted target speed inferred from the last two waypoints is in a 20% range of the instructed target speed. Second, if the predicted target speed inferred from the last two waypoints is in the 20% range of the speed of the last two waypoints of the ground truth speed waypoints. This can be different from the instructed target speed due to limitations in the acceleration rates of the vehicle.
- **Lane Change**: We compare the final waypoint of the predicted path waypoint with the ground truth dreamer path and the ground truth expert path waypoints. We define the lane change as successful when the predicted final location is closer to the dreamer's final location than the expert final location.
- **Objects (Collisions)**: This describes the task of driving towards or crashing into specific objects. We first look at the path. If the path of the expert trajectory and the ground truth dreamer trajectory is different (Average Displacement Error $ADE > 1.0$) we count it as success if the predicted path is closer to the ground truth dreamer path than to the expert path ($ADE_{pred2expert} > ADE_{pred2dreamer}$). If the dreamer path is nearly identical to the expert path ($ADE < 1.0$) the instruction is about correct speed predictions (e.g., if the instruction is "drive towards a dynamic object" it is important to get the speed right and not just the path. The success is then de-

fined as $ADE_{pred2dreamer} < 1.0$ and the average predicted speed is within 30% of the ground truth dreamer speed.

C. Quantitative Results

C.1 Ablations on CARLA Leaderboard 2.0

We provide additional results on SimLingo-BASE. *Output representation.* Tab. 3a compares the DS on the Leaderboard for the different output representations. As the goal of the additional path prediction is improved lateral control, we also report the collisions with static layout as this is mainly caused due to bad steering. With the disentangled representation, we can reduce the layout collision from 0.68 to 0 showing the strength of additional path predictions.
Vision-Language and CLIP pretraining. We ablate the pretraining of the vision encoder and train the same model from scratch. Tab. 3b 'w/o pretr.' shows that the pretraining stage is essential for good driving performance (longer training can further improve the performance but is unlikely to reach the performance of the pretrained model). Additionally, we show a comparison to the widely used Resnet-34 pretrained on ImageNet. The decreased performance (2.71 vs. 6.87 DS) indicates the importance of the larger ViT and the internet-scale image-language pretraining.
Early stopping. As described in the metric section the DS on long routes is not optimal and favors models that do not complete the full route. We ablate the thresholds for the early stopping as it is not trivial to calculate the perfect trade-off as the routes and density of scenarios are secret (however a rough function of the expected DS can be calculated, which we used to get a rough range). Tab. 3c shows the Leaderboard DS for a given traveled distance in meters. This hyperparameter has a big impact on the final score.
Leaderboard variance. We submitted our base model SimLingo-BASE with an early stopping threshold of 2100 and 2400 three times to the leaderboard to get an estimate of the evaluation variance. For the 2100 model, we obtain the following scores: 6.9, 5.5, and 5.3 resulting in a mean DS of 5.9 with a standard deviation of 0.87. The base model with a threshold of 2400 obtained 6.4, 6.3, and 4.8 resulting in a mean of 5.83 with a standard deviation of 0.90.

C.2 Bench2Drive Multi-Ability Results

Tab. 2 shows the Bench2Drive Multi-Ability metrics. Consistent with the findings in the main paper SimLingo outperforms existing works. Especially in the abilities *Merging, Overtaking, Emergency Brake* and *Traffic Sign* we get a large boost in performance. *Give way* is still challenging.

	Bench2Drive	
	DS ↑	SR(%) ↑
w/o CoT	84.41±1.76	64.84±2.42
with CoT	**85.07**±0.95	**67.27**±2.11

Table 4. **Inference Mode**. BEnch2Drive results for using SimLingo with chain-of-thought (CoT) during inference and without. We see a small improvement when using CoT.

C.3 Chain-of-Thought Inference Mode

We ablate the Chain-of-Thought (CoT) inference mode of SimLingo in Tab. 4. When using the Commentary task as CoT we see a small improvement so we decided to use this as the default mode. However, since without CoT the performance does not drop much, using the model without CoT is a feasible option especially when inference speed is important.

D. Qualitative Results

We provide more qualitative results of different navigational commands in Fig. 2. Red dots are the predicted path waypoints and green are the predicted speed waypoints. Each row is captured from a closed-loop run while changing the navigation command. The second row shows how the model can successfully differentiate between different situations and adapt its behavior given a certain command. The vehicle starts in the right lane. When giving the instruction "Turn left..." the model initiates a lane change to the left lane. After finishing this lane change it stays on this lane even though the command is still "Turn left...". So the model learns to reason about the meaning of the different lanes, that it is forbidden to go on an oncoming lane, and that "Turn left" not always mean to do a 90-degree turn. In the third row, we prompt the model with a misleading instruction: "Turn right" on an intersection without a lane going to the right. The third and fourth image shows that the model is slightly confused but when unrolling closed-loop the car still goes straight and stays on the road. In the fourth row, we show out-of-distribution commands. The model is still able to choose a valid path when using a command that does not make sense like "I really like my dog". When using the command "Why is there a tree on the right side?" the model still picks the left turn, indicating that it does not just overfit on single words like *left* or *right* but also takes the context into account. The last image with the command "Do a U-Turn" shows that the model is not capable of following concepts it has never seen during training.
Fig. 3, Fig. 4, Fig. 5, Fig. 6 and Fig. 7 shows qualitative results for the different Dreamer modes. In most cases, the model correctly follows the instructions even if it clearly goes against the expert driving behavior (e.g., accelerating at a red traffic light). We also include some of the failure cases of the model in the red boxes where the model ignores the instructions.

References

[1] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *ECCV*, 2016. 6

[2] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, et al. Planning-oriented autonomous driving. In *CVPR*, 2023. 6

[3] Xiaosong Jia, Yulu Gao, Li Chen, Junchi Yan, Patrick Langechuan Liu, and Hongyang Li. DriveAdapter: breaking the coupling barrier of perception and planning in end-to-end autonomous driving. In *ICCV*, 2023. 6

[4] Xiaosong Jia, Penghao Wu, Li Chen, Jiangwei Xie, Conghui He, Junchi Yan, and Hongyang Li. Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *CVPR*, 2023. 6

[5] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *ICCV*, 2023. 6

[6] Alon Lavie and Abhaya Agarwal. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *ACL Workshop*, 2007. 6

[7] Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Zhang Hanxue, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. DriveLM: driving with graph visual question answering. In *ECCV*, 2024. 2, 6

[8] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, 2015. 6

[9] Peng Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. In *NeurIPS*, 2022. 6

[10] Jiang-Tian Zhai, Ze Feng, Jihao Du, Yongqiang Mao, Jiang-Jiang Liu, Zichang Tan, Yifu Zhang, Xiaoqing Ye, and Jingdong Wang. Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenes. *arXiv preprint arXiv:2305.10430*, 2023. 6

[11] Julian Zimmerlin. Tackling carla leaderboard 2.0 with end-to-end imitation learning. Technical report, University of Tübingen, 2024. 5

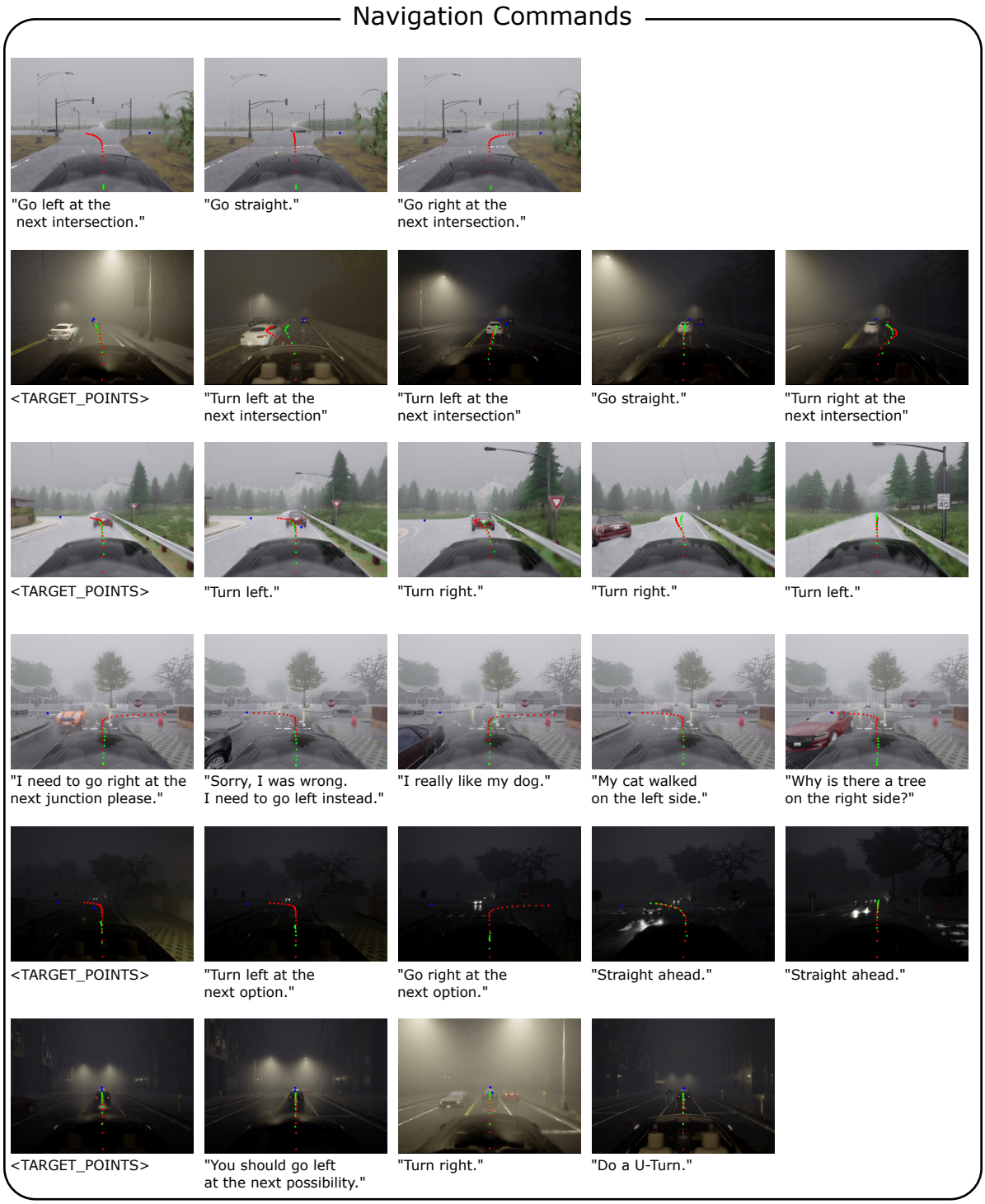

Figure 2. **Navigational Commands.** We show closed-loop results for different in-distribution and out-of-distribution commands. Red: path waypoints, green: speed waypoints.

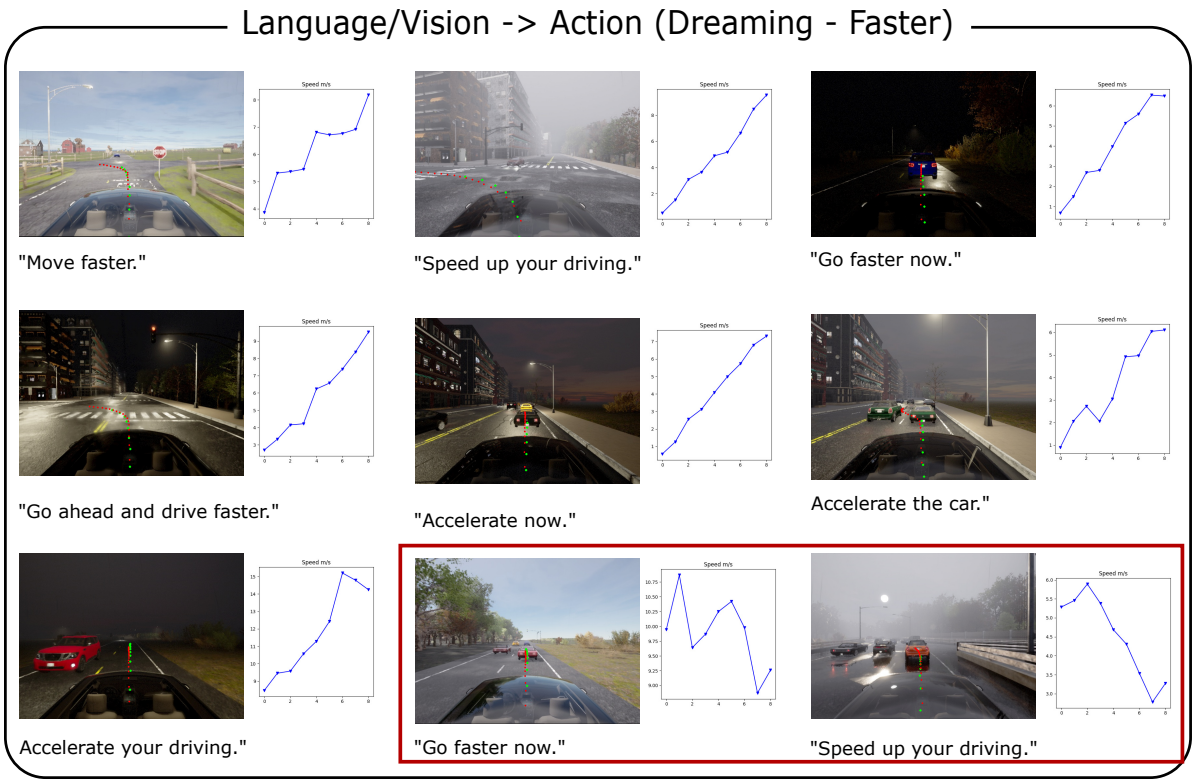

Figure 3. **Dreamer mode - Faster.** We show in blue the predicted speed curve (inferred from the speed waypoints). Inside the red border are examples where the model does not follow the command correctly. Red: path waypoints, green: speed waypoints.

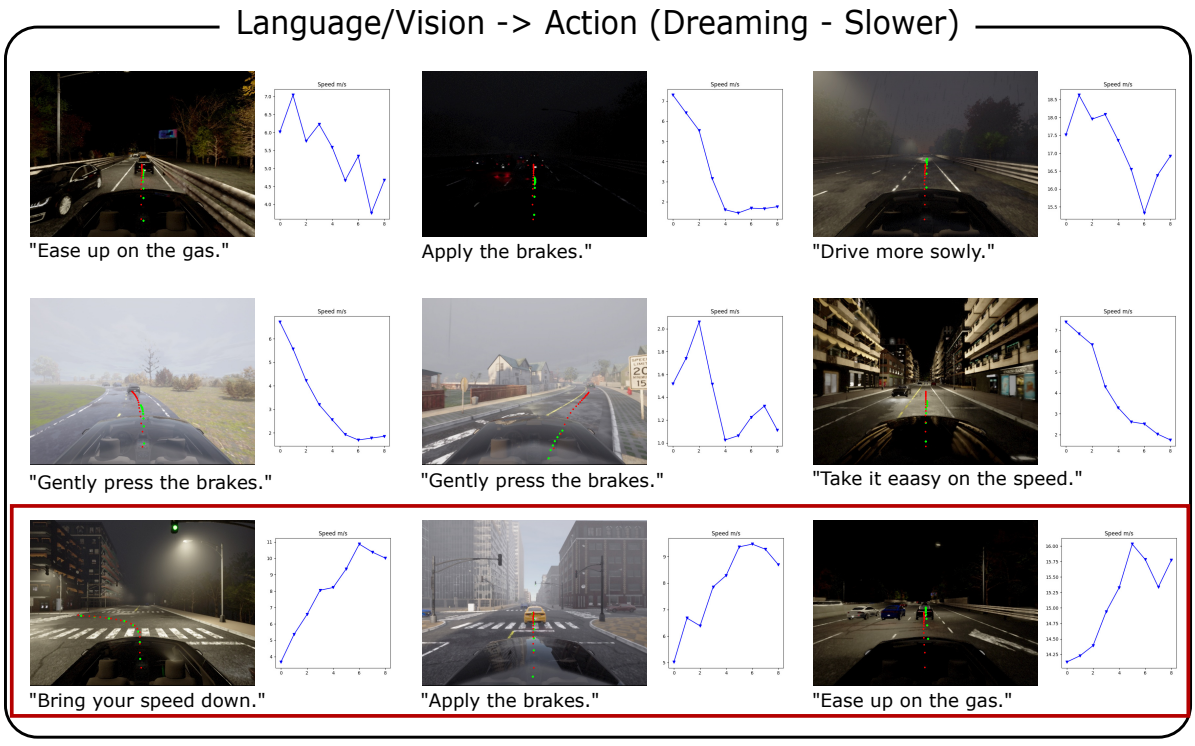

Figure 4. **Dreamer mode - Slower.** We show in blue the predicted speed curve (inferred from the speed waypoints). Inside the red border are examples where the model does not follow commands correctly. Red: path waypoints, green: speed waypoints.

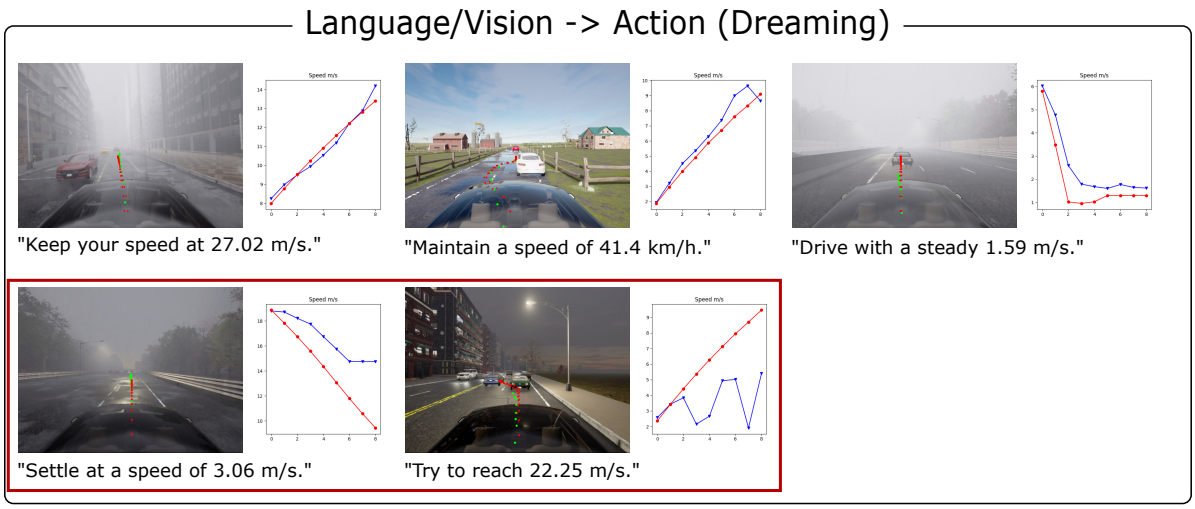

Figure 5. **Dreamer mode - Target speed.** We show in red line plot the ground truth speed curve and in blue the predicted one (inferred from the speed waypoints). Inside the red border are examples where the model does not follow commands correctly. Red: path waypoints, green: speed waypoints.

Figure 6. **Dreamer mode - Object (Collision).** Inside the red border are examples where the model does not follow dreamer mode command correctly. Red: path waypoints, green: speed waypoints.

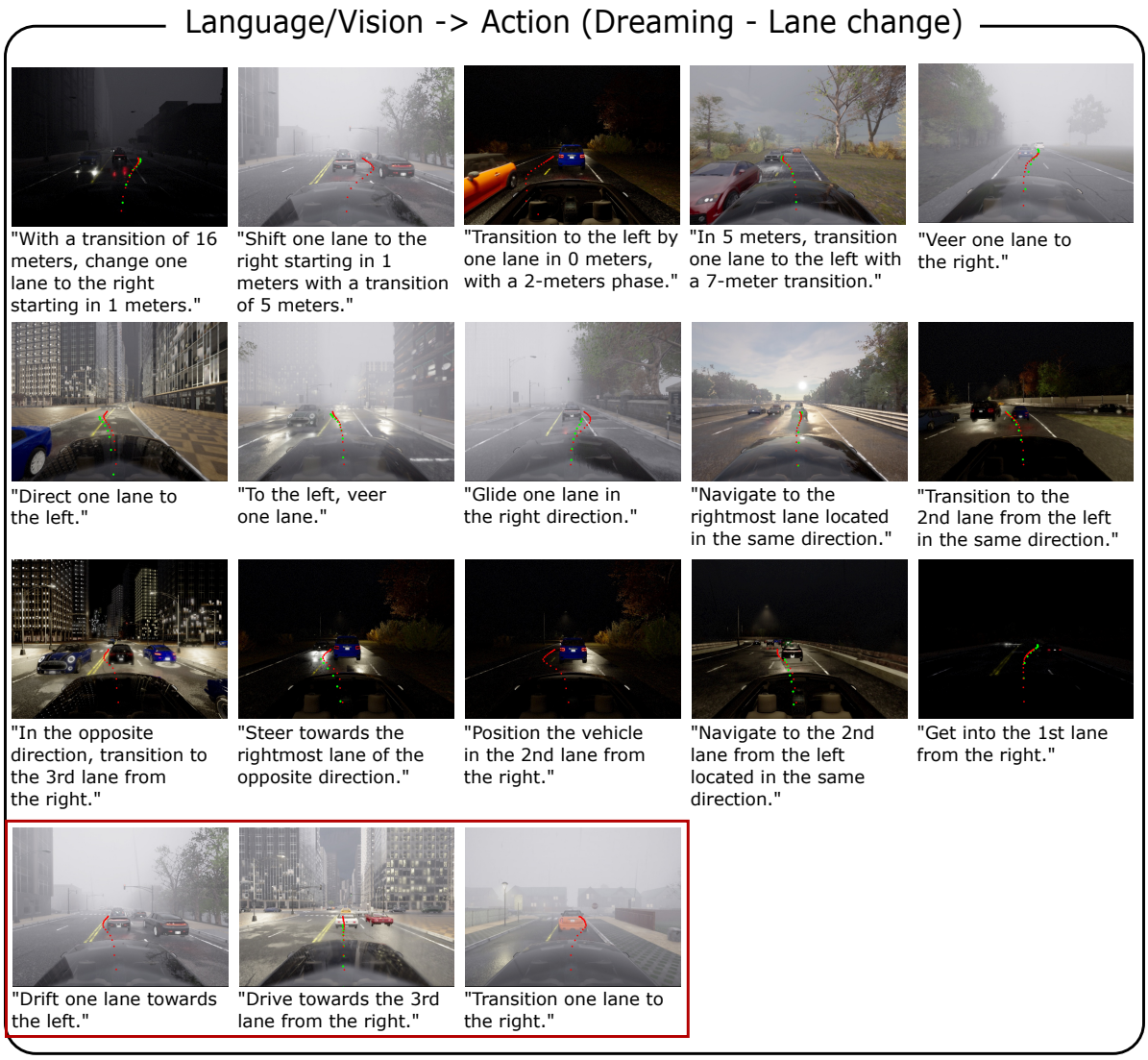

Figure 7. **Dreamer mode - lane changes.** Inside the red border are examples where the model does not follow dreamer mode commands correctly. Red: path waypoints, green: speed waypoints.