POLICY PRE-TRAINING FOR AUTONOMOUS DRIVING VIA SELF-SUPERVISED GEOMETRIC MODELING
Supplementary Materials

In this Supplementary document, we first provide detailed network structures in Sec. A. More description and visual illustrations of the downstream tasks are discussed in Sec. B. Last, we discuss limitations and common failure cases in Sec. C.

A NETWORK DETAILS

For all experiments, the backbone of the visual encoder is ResNet-34 (He et al., 2016), and the detailed structure of it is provided in Table 7. For DepthNet and PoseNet, we follow the same model structure as Godard et al. (2019) with a two-layer MLP focal length head and a two-layer MLP optical center head added to the bottleneck of the PoseNet to predict the intrinsic matrix. Please refer to Godard et al. (2019) for model details.

For the Navigation, Navigation Dynamic, and Reinforcement Learning tasks, we use CILRS (Codevilla et al., 2019) and the model details are provided in Table 8. For the Leaderboard Town05-long task, TCP (Wu et al., 2022) is chosen as our agent, and we refer readers to Wu et al. (2022) for model details. For the nuScenes Planning, the trajectory planning model structure is shown in Table 9.

Table 7: Detailed structure of the visual encoder.

Layer Type	Channels	Stride	Kernel Size	Activation Function
Image Encoder				
ResNet-34				
Measurement Encoder				
Conv	256	1	1	ReLU
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
Conv	6	1	1	ReLU
Average Pooling				

Table 8: Detailed structure of the CILRS model.

Layer Type	Dims in	Dims out	Activation Function
Image Encoder			
ResNet-34		512	
Speed Encoder			
FC	1	256	ReLU
FC	256	512	-
Speed Pred Head			
FC	512	256	ReLU
FC	256	256	ReLU
FC	256	256	ReLU
Control Pred Head			
FC	512	256	ReLU
FC	256	256	ReLU
FC	256	3	Sigmoid

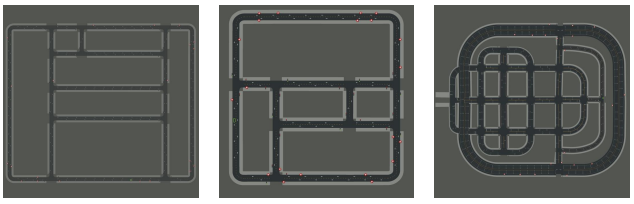
Table 9: Detailed structure of the trajectory planning model.

Image Encoder			
ResNet-34			
Bottleneck			
Layer Type	Dims in	Dims out	Activation Function
FC	512	256	ReLU
FC	256	256	-
Decoder			
Layer Type	Hidden dim	Input Dim	Output Dim
GRU	256	2	2

B Downstream Tasks Details

For **Navigation** and **Navigation Dynamic**, training data is collected in Town01, and the closed-loop testing is conducted in Town02. The maps of Town01 and Town02 are shown in Fig. 5. The agent needs to follow a series of sparse waypoints to navigate from the start point to the end point and avoid collisions. The difference between Navigation and Navigation Dynamic is that there are other dynamic vehicles and pedestrians in the town. Examples are provided in Fig. 6.

The **Leaderboard-Town05-long** task is more close to real-world urban driving, with different challenging scenarios added to the route. The map of Town05 is shown in Fig. 5.



Town 01 Town 02 Town 05

Figure 5: Maps of Town01, Town02, and Town05.

Navigation Navigation Dynamic

Figure 6: Examples of the front view image for Navigation and Navigation Dynamic tasks.

C LIMITATIONS

In this part, we analyze some failure cases and limitations of our method. Since the visual encoder need to predict the future motion based on a single front-view image, there might be some factors that directly influence the driving decision not shown in the image (*e.g.*, vehicles behind the ego vehicle, factors related to the driver, navigation information). Some of such cases are provided in Fig. 7. In these cases, the visual encoder does not get enough information to make the correct prediction. These samples during training may hamper the learning process. After training, one may use the difference between the prediction from PoseNet and that from visual encoder to filter out these samples, and re-train the visual encoder.

Figure 7: Failure cases where the driving decision/future motion can not be inferred from I_t. For the cases in Row 1 and Row 2, by comparing I_t and I_{t+1}, we know that the ego vehicle stops. However, there is no clear clue in I_t indicating it should stop. For the case in Row 3, the ego vehicle is turning left, while we could hardly tell the turning direction from I_t alone.