ST-P3: End-to-end Vision-based Autonomous Driving via Spatial-Temporal Feature Learning Appendix

Shengchao Hu[1†], Li Chen[2*], Penghao Wu[1,3†], Hongyang Li[1,2], Junchi Yan[1,2], and Dacheng Tao[4]

[1] MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
[2] Shanghai AI Laboratory, Shanghai, China
[3] The University of California, San Diego, CA, USA
[4] JD Explore Academy, JD.com Inc., Beijing, China
charles-hu@sjtu.edu.cn lichen@pjlab.org.cn

A Implementation Details of the Network

A.1 Architecture for Perception and Prediction

Spatial-Temporal Perception. For nuScenes dataset, we first crop and resize the original image $\mathbb{R}^{3\times900\times1600}$ to $\mathbb{R}^{3\times224\times480}$ and take the past 3 frames to the model, denoted by $I_t^n \in \mathbb{R}^{3\times224\times480}$, where $t \in \{1,2,3\}, n \in \{1,\ldots,6\}$. We use EfficientNet-b4 [8] as the backbone, obtaining features $f_i^k \in \mathbb{R}^{C\times H_e\times W_e}$ and depth estimation $d_i^k \in \mathbb{R}^{D\times H_e\times W_e}$ where $C=64, D=48, H_e=28, W_e=60$. Note that the depth ranges from $2m$ to $50m$ with spacing $1m$. After spreading the feature across the entire ray of space according to the predicted depth distribution, camera images from all angles are denoted by $u_i^k \in \mathbb{R}^{C\times D\times H_e\times W_e}$. Then with the ego-motion matrix, features for all surround cameras and timestamps could be transformed to the coordinate system centered at the SDV at time t, resulting in ego-centric features $\{u_i^{'}\}$. Finally the BEV features maps $b_i \in \mathbb{R}^{C\times H_b\times W_b}$ could be sum pooled from $\{u_i^{'}\}$ with $H_b=200, W_b=200$.

In order to boost the features, we can integrate historical information to every frame through an accumulation method in the temporal fusion step. Then these features are fed into a temporal network realized by 3D convolutions to better align temporal features. In particular, in order to take different receptive fields into account, we apply different 3D kernel sizes on the time channel with $[(2,3,3),(1,3,3)]$ and a pyramid pooling with kernel size $(2,200,200)$. We concatenate all the outputs to feed into the final compression convolution layer, getting the final output $x_{1\sim t} \in \mathbb{R}^{C\times H_b\times W_b}$. Note that the output features keep the same shape with the input after passing through the temporal network, and features from different timestamps are integrated.

* Correspondence author. † Work done during internship at Shanghai AI Laboratory.

Future Prediction. In our experiments, we model the future uncertainty by two different distributions: Gaussian and Bernoulli. For Gaussian distribution, the present feature x_t is passed through several residual block layers and an average pooling layer to get the hidden state $\mathbb{R}^{32\times1\times1}$. Then a 2D convolution with kernel size $(1, 1)$ fits the mean and log variance of the Gaussian with $R^{32} \times R^{32}$. For Bernoulli distribution, since each grid of the spatial is $0 - 1$, we pass x_t to several residual block layers and through a LogSigmoid to regress the probability in each grid. For the prediction network, we utilize the convolutional Gated Recurrent Unit as our basic module, and each gate is realized by a 2D convolution with stride 3. We combine the predicted features through a trusting gate implemented by a 2D convolution which has 2 output channels, and use it as the weight to sum the two predicted features. The mixed predicted features are then used as the hidden state for the "uncertainty" pathway and the input state for the "historical" pathway. Through this method, we could recursively predicts future states $(\hat{x}_{t+1}, \ldots, \hat{x}_{t+H})$.

Decoders for BEV Representations. All task heads share the same backbone considering the robustness of the decoder, which is implemented by the first three layers of ResNet-18 [3] and three upsampling layers of factor 2 with skip connections. Features now has 64 channels, and then are passed to different heads according to the task requirement. Each head is composed of two 2D convolutions with different output channels. In particular, we set that the number of output channel for all semantic segmentation heads is 2. However, we need to predict offset, center, future flow for instance segmentation task, thus they are set as 2, 1 and 2 respectively.

A.2 Planning

In this section we will give more detailed the scoring functions and the implementation of the refinement GRU network.

Safety Cost. The SDV should not collide with other objects on the road and need to consider their future motion when planning its trajectory. For this purpose, we use the predicted occupancy map to penalize the trajectories that intersect with the occupied regions. Formally, for trajectories τ at each timestamp t, we will penalize τ if the SDV polygon g intersects with the grids which are occupied by other objects (with a safety margin indicated by parameter λ), denoted by the $o_g(\tau, t, \lambda)$. The safety cost related to objects collision is given by:

$$f_o(\tau, o) = \sum_t \sum_g o_g(\tau, t, 0) + o_g(\tau, t, \lambda)v(\tau, t), \tag{1}$$

where the first term penalizes the intersection grids whereas the second term penalizes the high-velocity motion with uncertainty occupancy.

Moreover, SDV usually follows a leading vehicle and should keep a certain safe distance from it which mainly depends on the speed of the leading vehicle. Since the HD map is unavailable and thus the leading vehicle along the center

lane is unknown, instead we compute the occupancy grid in front of the SDV with a distance L determined by the SDV velocity. Hence if we are too close to the leading vehicle, the cost will be large that the trajectory will be evaluated as bad. However it cannot take effect when it is in lane change manoeuvres since the leading vehicle is not directly in front of SDV.

The above objectives focus on moving objects, meanwhile the vehicle should stay in the middle of the lane line as well. We can ensure this traffic safety with the perceived lanes in the first stage, with a cost function that is set as the distance to lane lines.

Refinement. Getting the selected trajectory τ^* according to the scoring functions, we then refine it through a GRU network. Specially, the hidden state of the GRU is the feature of the front camera from the encoder, and the input state is the concatenation of the current position, the position from the selected trajectory τ^* and the target point. Note that the initial current position is $(0, 0)$. Doing the same process, we could recursively obtain the final refinement trajectory taking into consideration the front camera information and the target point information.

B Depth Supervision

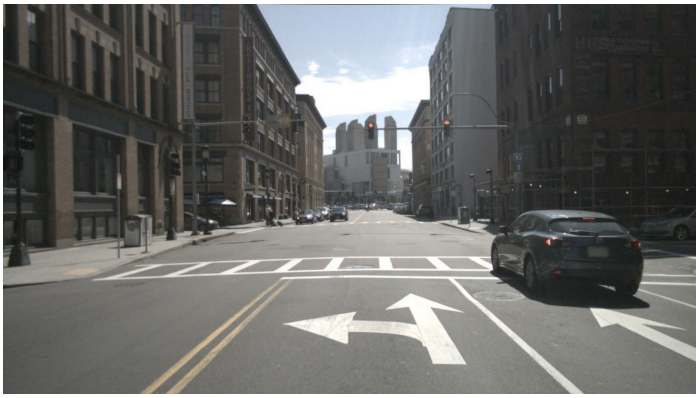
In this section, we introduce how the depth maps of nuScenes dataset are generated for explicit supervision.

We adopt a self-supervised, semantic segmentation required method named FSRE-Depth [5], which has great performance on depth generation. Since FSRE-Depth use semantic segmentation as input, we first train a segmentation model on Mapillary Vistas Dataset [6]. Then we apply it on nuScenes dataset to generate semantic results, as shown in Fig. 1(b). In the next step, We train FSRE-Depth with front view images in nuScenes, using ResNet-50 as the backbone. The resolution of input images is 1216×672. After 45 epochs of training, the model can predict sufficient results on front view images. To leverage the performance of depth model, we further train it on each camera view for 15 epochs. Thus, our depth maps of all camera views could be produced by this model individually. To evaluate our depth maps, we calculate depth estimation metrics with LiDAR points every 20 images. The result is shown in Tab. 1, which indicates that our depth maps have decent quality. This result in Fig. 1(c) would be utilized to explicitly supervise the depth estimation in the perception module of ST-P3.

C Experiments

C.1 Protocols

Dataset. We evaluate ST-P3 in both open-loop and closed-loop environments. We adopt nuScenes dataset [1] for the open-loop setting, and CARLA simulator [2] for the closed-loop demonstration.



(a)

(b)

(c)

Fig. 1. (a) Original image; (b) Segmentation result; (c) Predicted depth map

Table 1. Depth map evaluation. ARE: absolute relative error; SRE: square relative error; RSME: root mean square error; RSME log: root mean square logarithmic error; δ: accuracy (threshold 1.25)

View	ARE	SQE	RSME	RSME log	δ	δ^2	δ^3
FRONT	0.1522	3.5165	6.8756	0.2252	0.8720	0.9474	0.9701
FRONT LEFT	0.2516	2.6591	5.7600	0.3035	0.7290	0.8755	0.9317
FRONT RIGHT	0.3030	5.9052	6.6903	0.3278	0.7067	0.8651	0.9231
BACK	0.2297	5.5757	7.8486	0.3020	0.8011	0.9159	0.9535
BACK LEFT	0.2752	3.6375	5.7691	0.3110	0.6982	0.8673	0.9279
BACK RIGHT	0.3021	3.9396	6.2027	0.3413	0.6605	0.8503	0.9279

For nuScenes, by default we take the $1.0s$ of past context and predict the future $2.0s$ contexts, which corresponds to 3 frames in the past and 4 frames in the future. Since each batch input to the model contains 7 frames of contexts, we just follow the offical split method to split the dataset into training, validation that consist of 26124 and 5719 samples, respectively.

For CARLA, we conduct closed-loop evaluation. Note that we still need to train our model in a open-loop manner first. We follow the dataset collection in [7] and use the Town05 scenario for evaluation and the rest for training. It is worth mentioning that the camera setup in CARLA only consists of 4 cameras; thus it could not cover the 360° field of view around the SDV. The context from the past $1.0s$ and current 4 cameras images are passed to our model, then a trajectory is produced and executed by the simulator until SDV getting to the destination or surpassing the time limitation.

Implementation Details. We adopt EfficientNet-B4 [8] as the backbone, and detailed model description would be illustrated in the Supplementary. We use the Adam optimizer with a constant learning rate 2×10^{-4}. We train our model on 4 Tesla V100 GPUs for 20 epochs at mixed precision. The BEV spatial is $(100m, 100m)$ around the SDV with resolution $(0.50m, 0.50m)$ on nuScenes dataset, following the setting in [4] for a fair comparison. While on CARLA, it is $(40m, 40m)$ around SDV with resolution $(0.20m, 0.20m)$. The time interval between two consecutive frames is $0.5s$ for both datasets.

C.2 Open-loop Experimental Results on nuScenes

In this section we show more qualitative results on nuScenes dataset [1]. We present the visualization of the learned cost volume and the composite graph of multiple semantic elements in the meantime. Note that darker color means smaller cost value here, and vice versa. As shown in Fig. 2-3, basically all the places occupied by cars have higher cost values, while open drivable areas have lower cost values.

C.3 Closed-loop Planning Results on CARLA Simulator

In this section we show the qualitative results on the CARLA simulator [2], similar to the open-loop results. As shown in Fig. 6, when the car deviates from

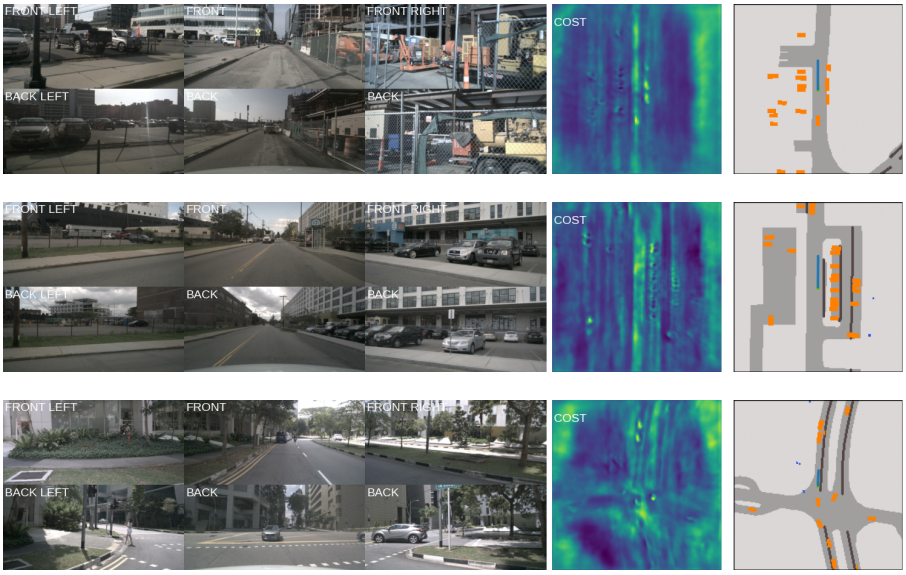


Fig. 2. Qualitative results of ST-P3 on the straight road. We show our BEV intermediate representations and planned trajectory (blue) in the right. We also present the learned subcost map from prediction module. Note that a darker color indicates a smaller cost value

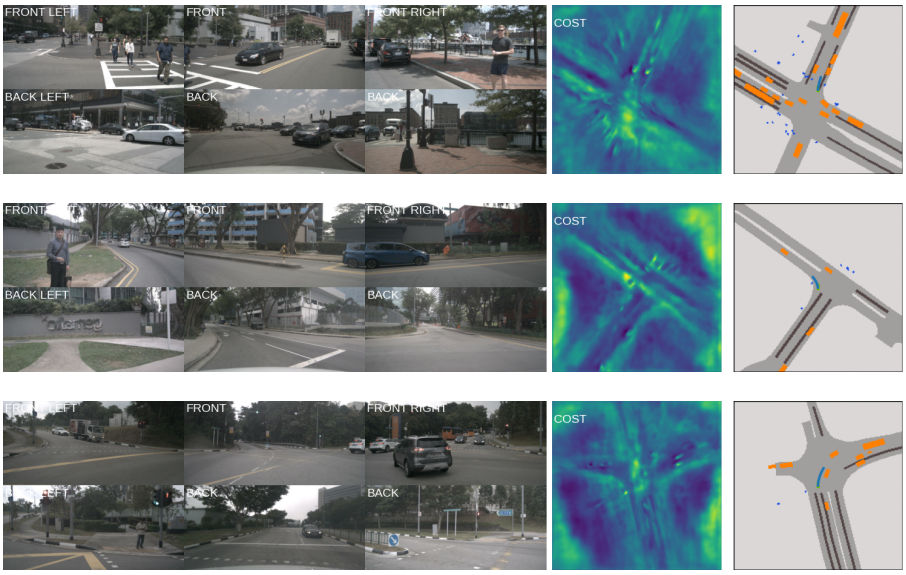


Fig. 3. Qualitative results of ST-P3 when turning at intersections

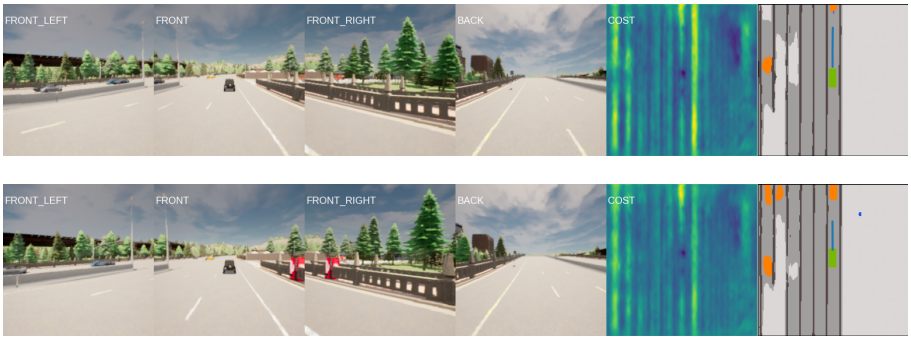


Fig. 4. Qualitative results of ST-P3 in closed-loop planning on the straight road

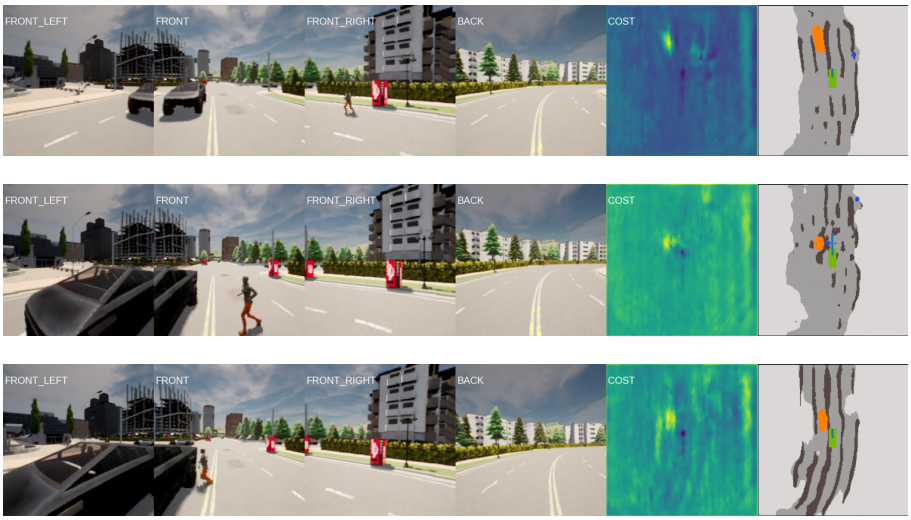


Fig. 5. ST-P3 predicts a slowing down trajectory when detecting a pedestrian

the center of lanes, the detection accuracy of the map drop significantly. This is probably because the data collection strategy that most training data is on normal circumstance. When the map accuracy is low, the traditional method which utilizes the sampler and cost map generally deviates from the expected track behavior, but due to our refinement operation, the car can still travel to the expected track correctly. We provide more visualization in different scenarios in Fig. 4-7.

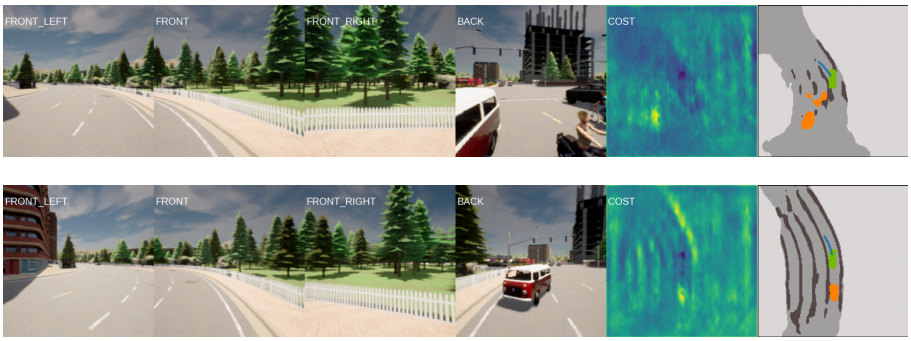


Fig. 6. BEV representations of ST-P3 become normal when driving on a predetermined trajectory (centerlines)

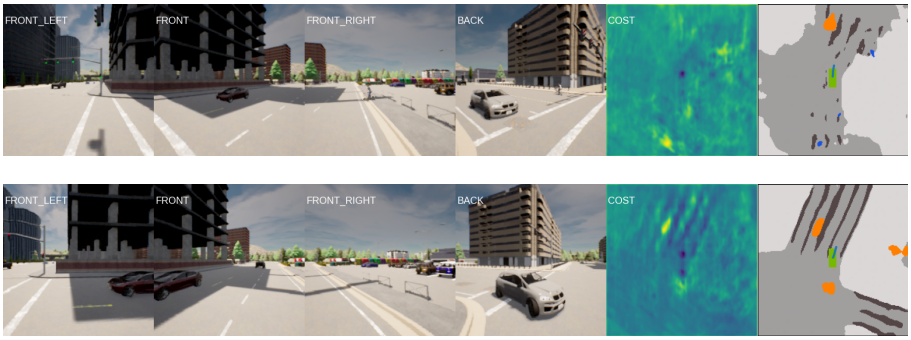


Fig. 7. Qualitative results of ST-P3 in closed-loop when turning at intersections

References

1. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: CVPR (2020) 3, 5
2. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: CoRL (2017) 3, 5
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 2
4. Hu, A., Murez, Z., Mohan, N., Dudas, S., Hawke, J., Badrinarayanan, V., Cipolla, R., Kendall, A.: Fiery: Future instance prediction in bird's-eye view from surround monocular cameras. In: ICCV (2021) 5
5. Jung, H., Park, E., Yoo, S.: Fine-grained semantics-aware representation enhancement for self-supervised monocular depth estimation. In: ICCV (2021) 3
6. Neuhold, G., Ollmann, T., Bulò, S.R., Kontschieder, P.: The mapillary vistas dataset for semantic understanding of street scenes. In: ICCV (2017) 3
7. Prakash, A., Chitta, K., Geiger, A.: Multi-modal fusion transformer for end-to-end autonomous driving. In: CVPR (2021) 5
8. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: ICML (2019) 1, 5