

Safe Local Motion Planning with Self-Supervised Freespace Forecasting

Supplementary Materials

Peiyun Hu¹, Aaron Huang¹, John Dolan¹, David Held¹, Deva Ramanan^{1,2}

¹ Robotics Institute, Carnegie Mellon University, ² Argo AI

{peiyunh@cs, aaronhua@andrew, jdolan@andrew, dheld@andrew, deva@cs}.cmu.edu

1 Additional visuals

Cost maps: We visualize the learned cost maps for a couple of scenarios on nuScenes. Please refer to the caption of Fig. 1 for details. We also include a video (.mp4) to show the whole log sequence of each scenario.

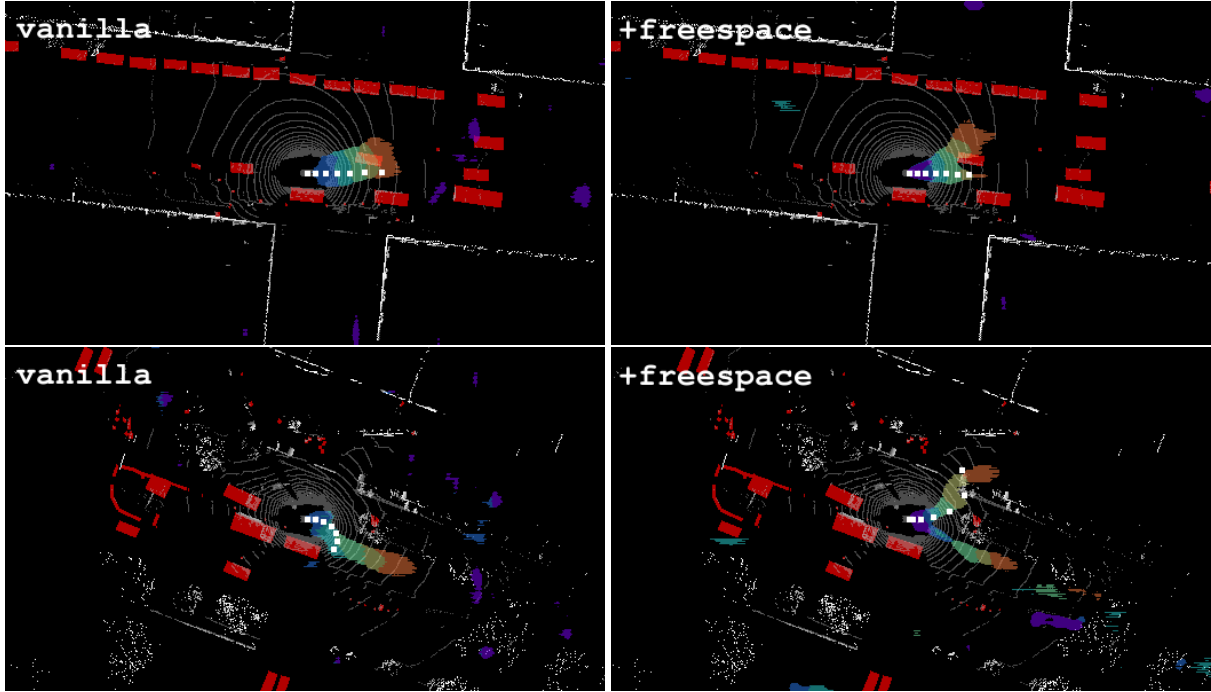


Figure 1: Examples of learned cost maps from the nuScenes test set. We visualize space-time cost maps by highlighting the low-cost regions of each future timestep. We use different colors to indicate different future timesteps. For reference, we also plot the top-scoring sampled trajectory and other objects in the scene (red). On the left, we visualize cost maps learned with ego-vehicle trajectories as the only source of supervision (vanilla). On the right, we visualize cost maps learned with both ego-vehicle trajectories and future freespace as supervision (+freespace). At top, we visualize a lane changing scenario. The *vanilla* cost map fails to capture the presence of a vehicle that is occupying the left lane, producing dangerously low costs. The cost map learned with future freespace manages to identify the other vehicle as high cost. At bottom, we visualize a scenario where the ego-vehicle attempts to take an unprotected left turn. The cost map learned with only ego-trajectories fails to identify a viable option for turning left. The cost map learned with future freespace manages to identify both viable options. See [nuscenescenes_1.mp4](#) and [nuscenescenes_2.mp4](#).

Forecasted freespace: We include a video to show forecasted freespace alongside ground-truth future freespace. We include a screenshot from the video in Fig. 2. Please refer to the caption.

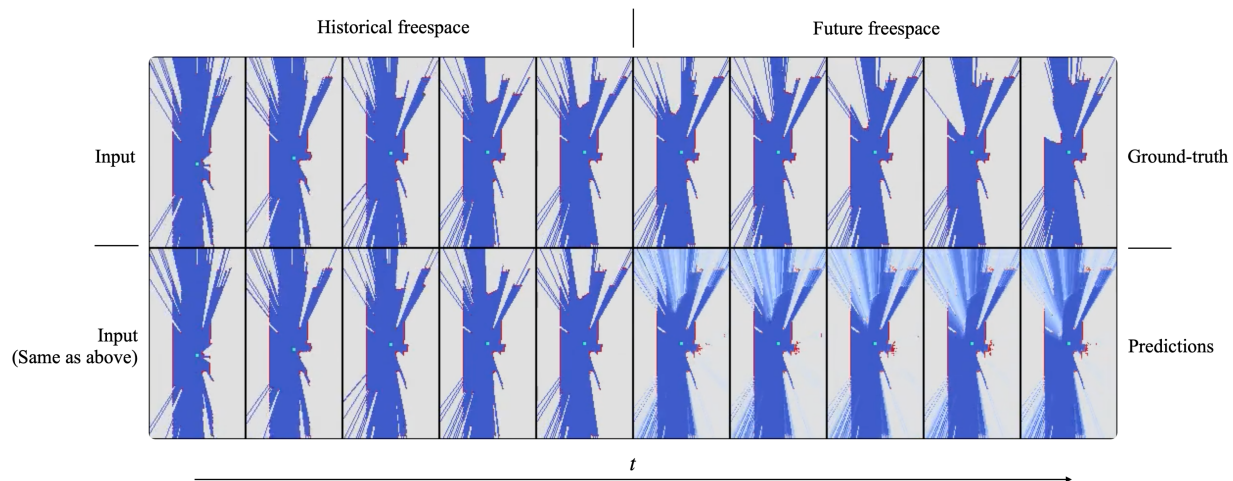
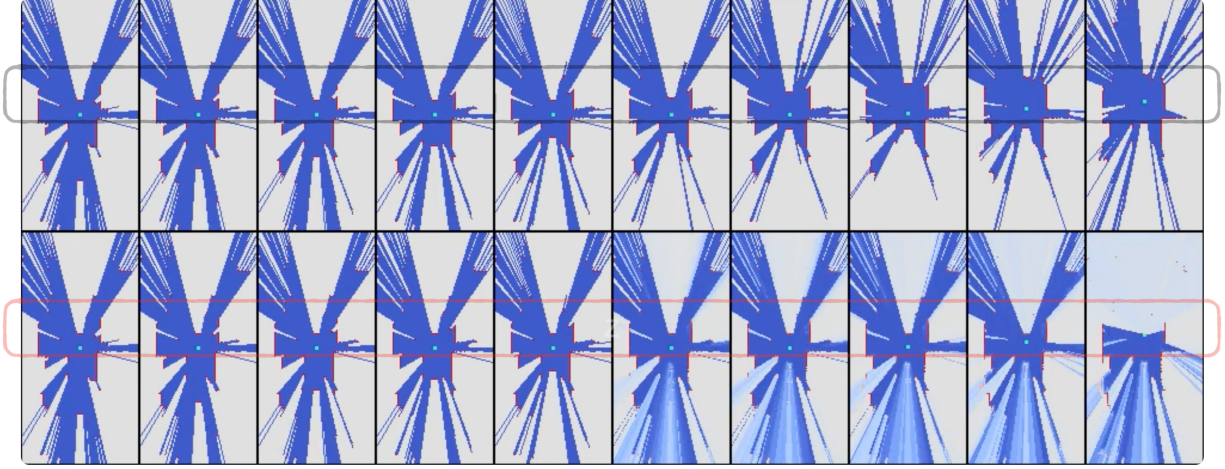
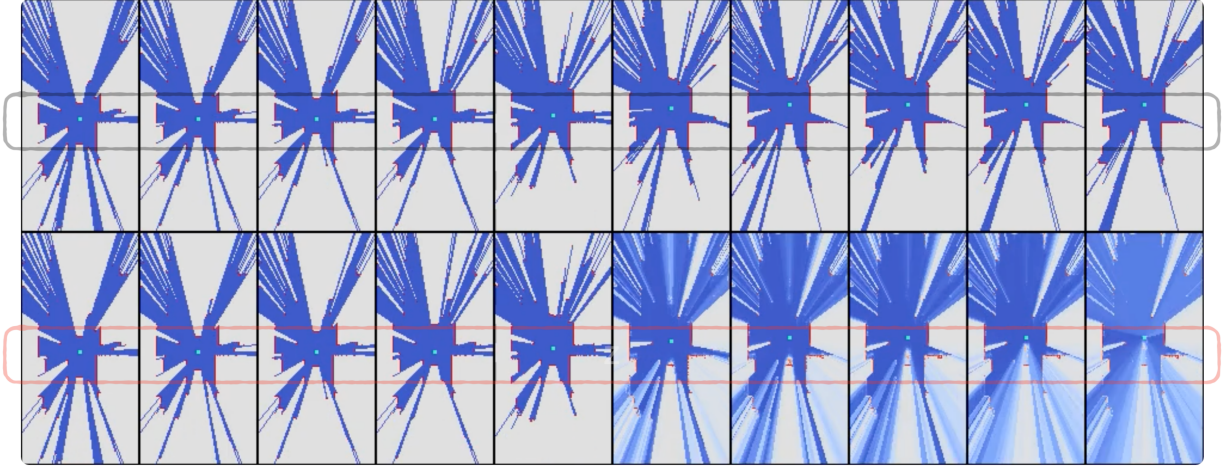


Figure 2: We visualize the input historical, ground-truth, and predicted future freespace. Notice that we perform *soft* raycasting when visualizing predicted future freespace. Green dots represent future locations of the ego vehicle. Here, we highlight a scenario where freespace forecasting captures the future motion of a left-turning vehicle from the opposing lane. Please find the complete video in [carla.turn.mp4](#).

Failure cases: We include a video to highlight typical scenarios where freespace forecasting tends to fail. We include two screenshots from the video in Fig. 3. Please refer to the caption.



(a) fail to foresee the acceleration of the vehicle in front



(b) fail to foresee the de-acceleration of the vehicle behind

Figure 3: We identify two failure modes for freespace forecasting. At top (a), we visualize a scenario where the freespace forecasting model fails to anticipate the acceleration of the vehicle in front. As a result, the ego vehicle will not start moving as soon as possible. At bottom (b), we visualize a scenario where the freespace forecasting model fails to anticipate the de-acceleration of the vehicle behind. As a result, the ego vehicle may perceive a rear-end accident. Such failure modes reveal the limited capacity of our freespace forecasting model at capturing the second-order motion. Please find the complete video in [carla.failures.mp4](#).

2 Additional diagnostic experiments

Loss function: We explore alternative loss functions for freespace forecasting. Since the classification formulation features imbalanced classes, we experiment with focal loss [2]. We adopt off-the-shelf parameters for focal loss, i.e., $\gamma = 2, \alpha = 0.25$. As we see in Tab. 1, training with focal loss could improve a freespace forecasting model's cross-town generalization performance.

Loss	Town 1(val)		Town 2(test)	
	F1	AP	F1	AP
BCE	0.772	0.830	0.755	0.773
FL	0.727	0.770	0.773	0.782

Table 1: Freespace forecasting validation performance with different loss functions, including Binary Cross Entropy (BCE) and Focal Loss (FL). We use the default default hyper-parameters, $\gamma = 2$, $\alpha = 0.25$, for FL. BCE achieves better performance on the validation set, which is collected in the same town where the training set is collected. FL achieves slightly better performance on the test set, which is collected in a different town. The results suggest FL may help cross-town generalization for freespace forecasting.

Feature encoding: We explore alternative ways to encode occupancy as an input representation to the freespace forecasting network. The simplest way to use raycasted freespace state. We refer to this encoding as *raw occupancy*. Voxels with varying freespace states through time likely matter more to planning in a dynamic environment. We add additional channels to encode changes in the freespace state. In addition, to focus on transitions *measured* sensors, when computing changes, if a voxel has an unknown state at time t , we default to its last known state up to time t . We refer to these additional channels as *delta occupancy*. As we see in Tab. 2, there is a small but consistent improvement from including *delta occupancy* for freespace forecasting.

Encoding	Town 1(val)		Town 2(test)	
	F1	AP	F1	AP
Raw	0.682	0.745	0.390	0.341
Delta	0.687	0.752	0.406	0.364

Table 2: Freespace forecasting validation performance with different input encodings. The results suggest additional the delta occupancy encoding contributes a small but consistent improvement to freespace forecasting. Here we compare two models without the design of residual forecasting hence the much worse test town performance.

Prediction horizon: We examine precision recall curves for each future timestamp. We normally compute one average precision for predictions from all future timestamps. Here, we plot a precision-recall curve for each future timestamp. As we see in Fig. 4, as the forecasting model predicts further into the future, there is a consistent decline in the average precision.

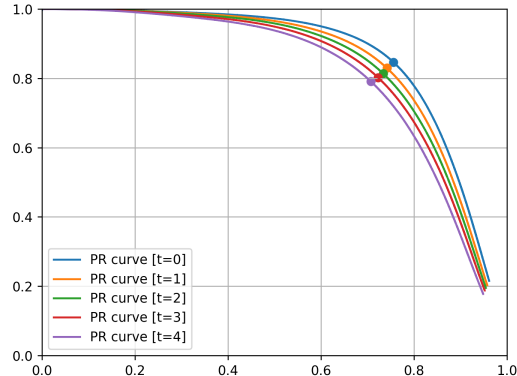


Figure 4: Freespace forecasting validation precision recall curves for each future timestep. We see performance gradually deteriorates as the model predicts further into the future. The highlighted dot on each curve corresponds to the maximum F1 score.

3 Additional implementation details

3.1 Planning on CARLA

Network architecture: We adopt a ResNet-18 backbone for our freespace forecasting model on CARLA. Our network architecture resembles the one of the privileged agent from Learning by Cheating [1]. Importantly, our network takes spacetime occupancy as input, rather than privileged bird’s-eye-view map information. The network input is a $L \times W \times D$ bird’s-eye-view spacetime occupancy from -2s to 0s. We use L and W to represent length and width under a bird’s-eye view. Here we focus on a spatial region of size $[-39.2\text{m}, 39.2\text{m}] \times [-19.2\text{m}, 19.2\text{m}]$ centered around the ego vehicle. We discretize at every 0.4m, therefore, $L=97$ and $W=193$. For *raw occupancy*, $D=5$. For *delta occupancy*, $D=18$, including additional channels to capture changes. The network output is a $L \times W \times T$ probabilistic spacetime occupancy from 0.5s to 2.5s, where $T=5$. We build 4 deconv layers (256, 128, 64, 5) on top of the ResNet-18 backbone as a decoder. We train the network for 90 epochs with a batch size of 512. We start with an initial learning rate of 0.2 and decay the learning rate by 0.1 every 30 epochs.

3.2 Planning on nuScenes

Trajectory sampling: Our trajectory sampler takes the current state of the ego-vehicle (y_1) and generates a set of plausible future trajectories, i.e., $\mathbf{Y}(y_1)$. Each trajectory is randomly generated using a two-step procedure. First, we choose to model the trajectory as a line, circle, or clothoid curve [4] with a probability of 0.5, 0.25, and 0.25 respectively. We use the instantaneous steering angle α (provided by nuScenes as part of CAN bus data release) to compute the initial curvature κ of a clothoid curve, following an approximate bicycle model [3]: $\kappa \approx 2\alpha/d$, where d is the distance between the front and rear axle (2.59m for nuScenes). Second, we sample a velocity profile to determine how fast the ego-vehicle travels along the geometric curve using an initial velocity and constant acceleration. Following [6], we encourage more diversity among the sampled trajectories by adopting the logged instantaneous velocity v and acceleration a 20% of the time. Otherwise, we sample the initial velocity and acceleration from prior distributions that fit urban driving ($v \sim U[0\text{m/s}, 15\text{m/s}]$ and $a \sim U[-5\text{m/s}^2, 5\text{m/s}^2]$). In total, we sample $|\mathbf{Y}(y_1)| = 200$ trajectories per example during training and 1000 trajectories during testing.

Network architecture: We adopt the same neural net architecture for both forecasting future freespace and learning a cost function for planning. The only architectural difference between the two networks is the addition of a sigmoid function to the forecasting network to produce probabilities. The input to the network is an ego-centric voxelized representation of the I most recent LiDAR sweeps which each have dimension $L \times W \times H$. We transform previous scans into the current LiDAR frame and use points within $[-70.4\text{m}, 70.4\text{m}] \times [-40\text{m}, 40\text{m}] \times [-2\text{m}, 3.4\text{m}] \times [-1\text{s}, 0\text{s}]$. We discretize spacetime at every 0.2m and every 0.1s. Therefore, $L=704$, $W=400$, $H=27$, $T=10$. We stack the I and H dimensions for a final input size of $L \times W \times (HI)$ to make our input amenable to 2D convolutions. Our network architecture resembles the architecture of PIXOR [5], a fully convolutional neural net designed for dense 3D object detection. The network outputs a space-time volume of dimension $L \times W \times T$ from 0.5s to 3.0s at every 0.5s. All baselines have the same network architecture and are trained for 15 epochs with a batch size of 12 over 4 GPUs. When learning to plan, following [6], we set the cost of collision to $\gamma_o = 200$. For simplicity, we also set non-freespace trespassing to $\gamma = 200$.

References

- [1] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020. 5
- [2] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 3
- [3] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016. 5

- [4] Mihail Pivtoraiko and Alonzo Kelly. A study of polynomial curvature clothoid paths for motion planning for car-like robots. 2004. 5
- [5] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018. 5
- [6] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019. 5