Safe Local Motion Planning with Self-Supervised Freespace Forecasting

Peiyun Hu[1], Aaron Huang[1], John Dolan[1], David Held[1], 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

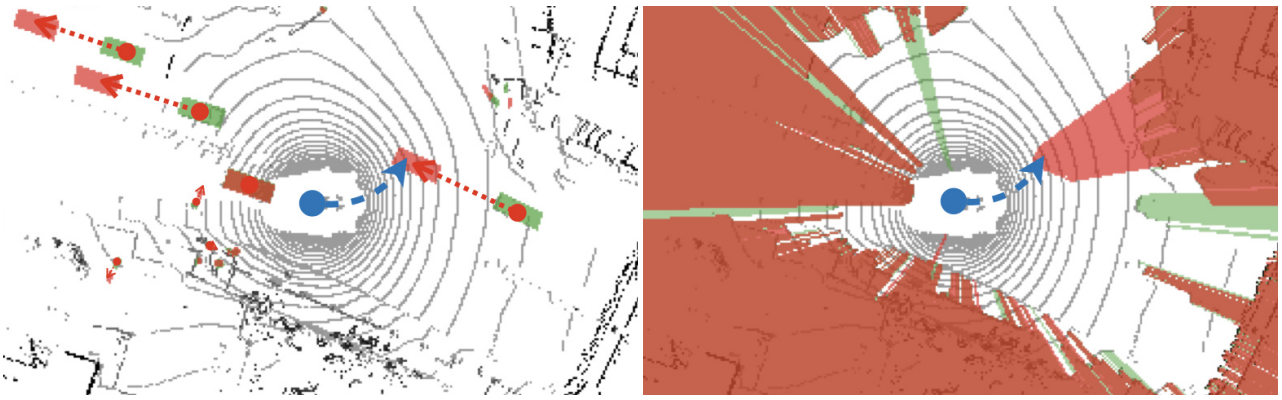


Figure 1: What are good 3D representations that support planning in dynamic environments? We visualize a typical urban motion planning scenario from a bird's-eye view, where an autonomous vehicle (AV) awaits an unprotected left turn. We highlight a candidate plan with a blue arrow, whose endpoint represents where the AV will be in 1s. An *object-centric* representation (left), as adopted by standard perception stacks, focuses on object properties (their shape, orientation, position, etc.) both at the current time step and the future. Alternatively, a *freespace-centric* representation directly captures the freespace of the surrounding scene and can be readily obtained by raycasting measurements from a depth (e.g., LiDAR) sensor. Forecasting a future version (in 1s) of either representation could help the AV identify a potential collision associated with the candidate plan, however at wildly different annotation costs. Forecasting future *object* trajectories requires a massive amount of object and track labels to train perceptual modules. Instead, we explore *future freespace*, whose forecasting can be naturally self-supervised by simply letting time move forward and raycasting future sensor measurements. We propose approaches to *planning with forecasted freespace* and *learning to plan with future freespace*.

Abstract

Safe local motion planning for autonomous driving in dynamic environments requires forecasting how the scene evolves. Practical autonomy stacks adopt a semantic object-centric *representation of a dynamic scene and build object detection, tracking, and prediction modules to solve forecasting. However, training these modules comes at an enormous human cost of manually annotated objects across frames. In this work, we explore* future freespace *as an alternative representation to support motion planning. Our key intuition is that it is important to avoid straying into occupied space regardless of what is occupying it. Importantly, computing ground-truth future freespace is* annotation-free. *First, we explore* freespace forecasting *as a self-supervised learning task. We then demonstrate how to use forecasted freespace to identify collision-prone plans from off-the-shelf motion planners. Finally, we propose future freespace as an additional source of* annotation-free *supervision. We demonstrate how to integrate such supervision into the learning-based planners. Experimental results on nuScenes and CARLA suggest both approaches lead to a significant reduction in collision rates.*[1]

1. Introduction

Motion planning in dynamic environments requires forecasting how the scene imminently evolves. What representation should we forecast to support planning? In practice, standard autonomy stacks forecast a semantic *object-centric* representation by building perceptual modules such as object detection, tracking, and prediction [42]. However, in the context of machine learning, training these modules comes at an enormous annotation cost, requiring massive

[1]Code will be available at https://github.com/peiyunh/ff

amounts of data manually annotated with *object* labels, including both 3D trajectories and semantic categories (e.g., cars, pedestrians, bicyclists, etc). With autonomous fleets gathering petabytes of data, it's impossible to label data at a rate that keeps up with the rate of data collection.

To avoid the need for such costly annotations, and to enable learning at scale, we explore an alternative *freespace-centric* representation to support motion planning (Fig. 1). We believe this is effective for two primary reasons. First, freespace is a natural cue for safe planning - it is generally important to avoid straying into occupied space, regardless of what is occupying it. Second, gathering training data for freespace forecasting is *annotation-free* given LiDAR scans recorded from an autonomous vehicle.

In this work, we propose two approaches for using a *freespace-centric* representation to assist with planning. First, we explore *freespace forecasting* as a self-supervised learning task. We point out essential modeling choices for building an effective predictor that forecasts freespace. Then, given an off-the-shelf black-box motion planner, we demonstrate that self-supervised future freespace predictions can be used to identify candidate plans that are likely to collide with objects in the near future.

Lastly, we propose using *future freespace* as an additional source of supervision when learning to plan. Many planners learn from expert demonstrations, and for example, learn to imitate good habits like maintaining a wide safety margin when approaching pedestrians in the street. However, it is difficult for the learner to know which other actions are bad, since there may have been multiple reasonable actions that could have been taken. We use *future freespace* to identify a subset of other actions that are clearly poor because they collide with an obstacle. We empirically show that imitative learning-based planners with such additional supervision produce motion plans that are far safer and less likely to induce collisions.

Contributions: We explore a self-supervised *freespace-centric* representation as an alternative to the predominantly supervised *object-centric* representation. We are the first to integrate self-supervised freespace predictions with an existing planner and demonstrate promising results. We also propose simple modifications to existing learning approaches to planning that allow future freespace to be used as an additional source of self-supervision. Finally, we demonstrate promising results on planning benchmarks.

2. Related work

Geometric planning: Classic planning algorithms such as A* [17] D* [40], PRM [23], and RRT* [22] usually assume static scene geometry and focus on efficiently finding the shortest collision-free path within the navigable freespace. A common workaround for motion planning in dynamic environments is to replan at high frequencies and reactively avoid moving objects [41, 43, 31, 3]. To avoid reactive planning, one must be able to forecast future evolution of geometry. This is typically done by building a modular perception pipeline that contains components for object detection, tracking, and forecasting. However, massive amounts of training data and annotated labels are required to train perception modules for discrete object classes. Purely geometric planning approaches also commonly suffer from an ambiguous interpretation of geometry (e.g. aggressively avoiding leaves blowing in the wind) or may not pick up on semantic cues (e.g. driving onto an empty opposite lane).

Behavior cloning: End-to-end learned approaches for autonomous driving have emerged as simple alternatives to modular autonomy stacks, with imitation learning methods showing particular promise [35, 8]. Imitation learning is generally split into two major classes: behavioral cloning and inverse optimal control (inverse reinforcement learning) [29]. Behavioral cloning refers to methods that learn a direct mapping from observations to actions using expert demonstrations. ALVINN [32] is a classic example of behavioral cloning for road following which uses a neural network to learn mapping from image to steering angle. More recently, [4] used a deep convolutional network to demonstrate real-world vehicle control in a variety of driving scenarios. Another approach uses video game driving demonstrations to train a network that maps images to driving affordances that can be directly used for control [5]. [7] trains a network to produce steering and acceleration commands from input images while also conditioning on a high level-command. More recently, [6] proposes a new approach to behavior cloning. Their results suggest that a privileged imitative learner, despite performing worse than the expert it learns from, may serve as a better teacher to non-privileged imitative learners, by providing richer supervision.

Inverse optimal control: Inverse optimal control (or inverse reinforcement learning) attempts to recover an unknown cost function from a set of expert demonstrations which can then be used for planning. [1] developed a seminal approach that cast the cost/reward model as a linear function of state features whose feature weights could be learned from expert demonstrations. Maximum Margin Planning (MMP) [34] is another classic approach that used a structured margin loss to learn a cost map that can produce expert-like trajectories via dynamic programming. [47] use the maximum-entropy principle to select solutions that show the least commitment to the training data, avoiding ambiguities that may arise if expert demonstrations are imperfect. This approach has recently been improved by using neural networks to approximate the underlying cost model [44]. Similarly, [46] extends MMP by using a deep neural network trained end-to-end with a multi-task loss to produce cost maps for trajectory scoring.

Additional supervision: One well-known challenge for

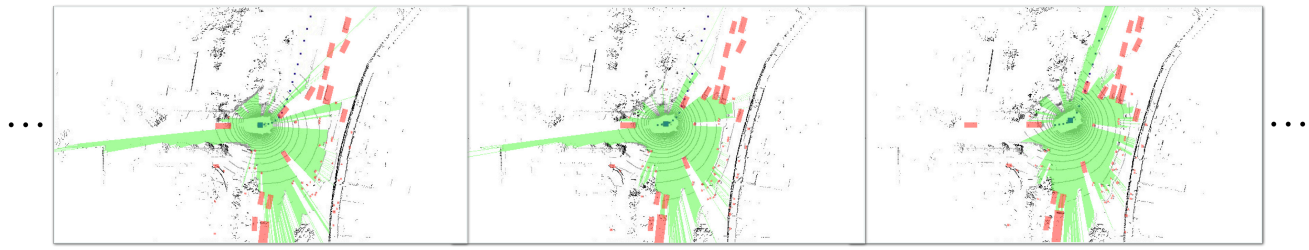


Figure 2: We illustrate a raw data log collected by an AV. In practice, such logs come in abundance. The crucial question is: *how do we use them to support local planning?* The widely-adopted approach tries to provide planners with knowledge of where objects are and will be (red). We try to provide knowledge of what future freespace looks like (green). We discuss two ways of how existing planners can use future freespace. First, we develop freespace forecasting models and demonstrate that their *predicted* future freespace can be used to improve off-the-shelf planners. Second, we show how to use *ground-truth* future freespace as additional supervision while learning to plan and demonstrate improvements by doing so.

imitation learning is figuring out how to recover from mistakes - commonly referred to as the "compounding error" problem [32, 37, 4]. [37] show that an effective solution is to have the expert interactively provide feedback by correcting the actions executed by a policy that is learning online. However, this is potentially dangerous to implement in the real world and may result in only sparse feedback. A more widely adopted strategy is to instead learn offline from historical (non-interactive) driving logs. Many researchers have used the CARLA simulator to record driving logs at scale using the built-in autopilot system [7, 8, 35, 12]. Simulation is particularly attractive because one has access to ground-truth labels of objects and the environment, which greatly simplifies learning [6]. However, transferring policies trained in simulation to the real world remains an active area of research. [2] perturb mid-level representations of real-world historical data to simulate nontrivial driving scenarios and were able to deploy their model on a real car. Our freespace forecasting approach can learn from both real historical data and simulated data and works directly with raw sensor data instead of needing object labels.

Self-supervised learning: Self-supervised learning has recently emerged as an effective approach for many robotic manipulation tasks [27, 14, 45, 24]. However, its usage in mobile robotics is far less prevalent, with most methods aimed at solving perception tasks like road detection [9], aerial image analysis [38], and lidar/camera depth completion [25]. One early application of self-supervised learning to robot navigation developed by [39] learned mappings from both online and offline perceptual data to planning costs, demonstrating navigation on the Crusher robot. More recently, [21] showed that a navigation system based on a generalized computation graph trained with self-supervised deep reinforcement learning (DRL) was able to outperform standard DRL approaches in both simulation and real-world RC car experiments. In this work, we pose future freespace forecasting as a scalable source of self-supervision and show that it is effective for motion planning.

Freespace as a representation: A few works have explored the question of *estimating* freespace. [16] estimate a top-view probabilistic occupancy map to track people in an indoor setting with a multi-camera setup. [18] estimate indoor freespace from a single image by leveraging "boxy" object detectors. Thanks to the progress in 3D sensing, recent works have been building upon freespace *measured* through depth sensors (e.g. LiDAR). [28, 10, 15, 20, 26] pose occupancy grid maps (OGMs) prediction as a self-supervised learning task and explore effective neural net architectures for this task. Our work on forecasting freespace is similar to prior works on predicting OGMs in learning with *self-supervision*. However, our work extends beyond the forecasting task itself in three meaningful aspects. First, we demonstrate how off-the-shelf planners can use forecasted freespace. Second, we demonstrate how to learn planners with future freespace as additional supervision. Third, we demonstrate improvements in terms of planning performance. Most recently, [36] learn to predict semantic occupancy maps. [11] learn to predict *future* semantic occupancy maps as a representation that supports downstream planning and demonstrate improvement in planning performance. Our work differs in that our *freespace-centric* representation is annotation-free and therefore more scalable.

3. Method

Raw logs of autonomous fleets naturally provide an abundance of *aligned* sensor data sequences $\mathbf{x}$ and ego vehicle trajectories $\mathbf{y}$, represented as collections of $\{(\mathbf{x}, \mathbf{y})\}$. We provide an example of such logs in Fig. 2. How do we make use of such data to learn representations that support planning? In the sections to follow, we first introduce the definition of freespace and how to compute it. Then we describe a self-supervised approach to forecasting freespace. Finally, we describe approaches to planning with forecasted freespace and learning to plan with future freespace.

3.1. Computing freespace

We define freespace as space free of obstacles as observed by a LiDAR sensor at a particular time instance. Given a sequence of *aligned* sensor data and ego-vehicle trajectory $(\mathbf{x}, \mathbf{y})$, let us write

$$\text{ray}(\mathbf{u}; \mathbf{x}, \mathbf{y}) \in \{0, -1, 1\}, \quad \mathbf{u} = (x, y, t), \quad \forall \mathbf{u} \in \mathbf{U} \quad (1)$$

to denote the freespace state of voxel $\mathbf{u}$ in the spacetime voxel grid $\mathbf{U}$, which can be *unknown* (0), *free* (-1), or *occupied* (1) respectively. Note that the spatial index of voxel $\mathbf{u}$ is 2D because we assume the local motion planners we work with operate on a ground plane and x, y represents a bird's-eye-view spatial location. When the freespace state of voxel $\mathbf{u}$ is *unknown*, the *true* state can be either *occupied* or *free* but is unobserved due to for example occlusion.

We compute freespace via raycasting. Given a 3D point cloud, we compute a 2D bird's-eye-view freespace following two steps. First, we identify LiDAR returns from the ground via a robust ground segmentation algorithm [19]. After we discard ground returns, we compute 2D freespace via a 2D visibility algorithm known as wall tracking [30]. We show example results in Fig. 2. This computation is automatic and does not require human annotators in the loop.

3.2. Forecasting freespace

Suppose we split each sensor trajectory pair $(\mathbf{x}, \mathbf{y})$ into a historical pair $(\mathbf{x_1}, \mathbf{y_1})$ and future pair $(\mathbf{x_2}, \mathbf{y_2})$, our goal is to learn a model that predicts freespace computed over $(\mathbf{x_2}, \mathbf{y_2})$ given freespace computed over $(\mathbf{x_1}, \mathbf{y_1})$. Crucially, we can compute ground-truth future freespace via raycasting for free (without human annotations)!

We train a convolutional neural network $f_\theta(\mathbf{u}; \mathbf{x_1}, \mathbf{y_1})$ with parameters θ to predict future freespace given the historical sequence $(\mathbf{x_1}, \mathbf{y_1})$ by minimizing the following loss:

$$\min_\theta \text{BCE}\Big(\sigma\big(f_\theta(\mathbf{u}; \mathbf{x_1}, \mathbf{y_1})\big), \text{ray}(\mathbf{u}; \mathbf{x_2}, \mathbf{y_2})\Big), \forall \mathbf{u} \in \mathbf{U} \quad (2)$$

where σ represents the sigmoid function and BCE stands for Binary Cross Entropy. Here, we use $\mathbf{U}$ to represent the voxel grid with future timestamps. The neural network produces *logits*, which are then converted to probabilities of voxels belonging to freespace through sigmoid.

Here, we formulate freespace forecasting as a binary classification and do not represent an *unknown* state as a separate state. This is because, as we have mentioned, when the ground-truth freespace state of a voxel future is *unknown*, its *true* state is either *occupied* or *free*. When computing the binary cross-entropy loss, we ignore such voxels with an ambiguous freespace state.

Residual forecasting: In most scenarios, future freespace does not look much different from historical freespace. This means we may be able to predict a majority of future freespace through interpolating historical freespace. Therefore, we decompose our freespace forecasting model into two parts, i.e., linear extrapolation and non-linear residual.

$$f_\theta(\mathbf{u}; \mathbf{x_1}, \mathbf{y_1}) = \overbrace{f_\alpha(\text{ray}(\mathbf{u_1}; \mathbf{x_1}, \mathbf{y_1}))}^{\text{linear extrapolation}} + \overbrace{f_{\tilde{\theta}}(\mathbf{u}; \mathbf{x_1}, \mathbf{y_1})}^{\text{non-linear residual}} \quad (3)$$

where f_α represents a linear extrapolation over spatially-aligned historical freespace and $f_{\tilde{\theta}}$ represents a non-linear predictor that forecasts *residual logits*. As we will show in Tab. 1, residual forecasting (3) is crucial to good accuracy.

3.3. Planning with forecasted freespace

Now we have introduced a self-supervised approach to freespace forecasting, how can an off-the-shelf planner work with forecasted freespace? We answer this question in the context of planners learned via both behavior cloning (BC) and inverse optimal control (IOC).

Behavior cloning (BC): A behavior cloning planner takes sensor data and ego-trajectories $\mathbf{x_1}, \mathbf{y_1}$ as input and predicts an expert-like future trajectory $\hat{\mathbf{y}}_\mathbf{2}$. The planner needs to know if the ego-vehicle can safely traverse each space-time voxel along the future trajectory. Our freespace forecasting model is designed to answer such queries, with one *caveat*: the model is trained to output a *soft* probability. We have to introduce a threshold that turns *soft* probabilities into *hard* decisions, similar to the fact that we have to pick a confidence score threshold for object detectors in standard autonomy stacks. Let τ be the threshold, we can test if a candidate future trajectory $\mathbf{y} = \{\mathbf{u}\}$ is safe by

$$q = \wedge_{\mathbf{u} \in \hat{\mathbf{y}}_\mathbf{2}} \left[f_\theta(\mathbf{u}; \mathbf{x_1}, \mathbf{y_1}) \leq \tau\right] \quad (4)$$

The planner passes the test of predicted future freespace when q is true. When it fails, we override the plan with a fall-back option, such as emergency braking maneuvers.

Inverse optimal control (IOC): An inverse optimal control approach to planning learns a cost map that scores potential trajectories, where the best one is found through optimization. We define the cost of a candidate trajectory $\hat{\mathbf{y}}_\mathbf{2}$ to be the sum of costs at its spacetime points:

$$C_\psi(\hat{\mathbf{y}}_\mathbf{2}; \mathbf{x_1}, \mathbf{y_1}) = \sum_{\mathbf{u} \in \hat{\mathbf{y}}_\mathbf{2}} \text{cost}_\psi(\mathbf{u}; \mathbf{x_1}, \mathbf{y_1}) \quad (5)$$

where cost_ψ is a spacetime *cost map* generated by a neural net, structurally similar to the freespace forecaster from Sec. 3.2. It is important that any candidate trajectory $\hat{\mathbf{y}}_\mathbf{2}$ maintains consistent and smooth dynamics with its immediate past $\mathbf{y_1}$ hence the conditioning. When integrating forecasted freespace into IOC planners, we directly modify the cost map to ensure that voxels predicted to be likely occupied incur very large costs,

$$C_{\psi,\theta}(\hat{\mathbf{y}}_\mathbf{2}; \mathbf{x_1}, \mathbf{y_1}) = \sum_{\mathbf{u} \in \hat{\mathbf{y}}_\mathbf{2}} \left[(\text{cost}_\psi + \gamma f_\theta)(\mathbf{u}; \mathbf{x_1}, \mathbf{y_1})\right] \quad (6)$$

where γ is a predefined cost w.r.t. future freespace violation.

Implementation: An IOC planner may search over an exponentially large number of potential trajectories with dynamic programming [34], explore a local set of trajectories through gradient-based optimization [33], or evaluate a set of sampled trajectories through exhaustive search [46]. We make use of the latter. Following [46], instead of modeling smoothness as part of the cost map, we enforce them as a constraint by restricting the space of viable trajectories $\mathbf{Y}(\mathbf{y_1})$ that is searched. This is formally equivalent to assigning trajectories not in $\mathbf{Y}(\mathbf{y_1})$ to be infinite cost.

$$\min_{\mathbf{y}\in\mathbf{Y}} C_\psi(\mathbf{y};\mathbf{x_1},\mathbf{y_1}) = \min_{\mathbf{y}\in\mathbf{Y}(\mathbf{y_1})} \sum_{\mathbf{u}\in\mathbf{y}} \text{cost}_\psi(\mathbf{u};\mathbf{x_1}) \tag{7}$$

When the space of viable trajectories $\mathbf{Y}(\mathbf{y_1})$ is available, we found it useful to restrict the freespace forecasting loss to the set of spacetime voxels reachable by the ego-vehicle. This results in a *sparse* loss in contrast to the original *dense* one. We will refer to freespace forecasting learned with the *sparse* loss as *planning-aware* freespace forecasting.

$$\min_\theta \text{BCE}\Big(\sigma\big(f_\theta(\mathbf{u};\mathbf{x_1},\mathbf{y_1})\big), \text{ray}(\mathbf{u};\mathbf{x_2},\mathbf{y_2})\Big), \forall \mathbf{u}\in\mathbf{Y}(\mathbf{y_1}) \tag{8}$$

3.4. Learning to plan with future freespace

We have discussed how an off-the-shelf planner can work with forecasted freespace. In particular, we show one can modify an IOC planner's cost map based on predicted future freespace. A follow-up question is: can we use ground-truth future freespace to learn a cost map that *naturally* reflects future freespace?

An IOC planner learns a neural net to predict a spacetime cost map. The network will be trained to ensure that ground truth future trajectory $\mathbf{y_2}$ has a lower cost than others:

$$C_\psi(\mathbf{y_2};\mathbf{x_1},\mathbf{y_1}) \leq \min_{\mathbf{y}\in\mathbf{Y}} C_\psi(\mathbf{y};\mathbf{x_1},\mathbf{y_1}) \tag{9}$$

Because not all alternative trajectories are equally bad, one often introduces a penalty that ensures the ground-truth *dominates* over those trajectories that lie far away by a margin $l(\mathbf{y},\mathbf{y_2})$ [34, 46]:

$$C_\psi(\mathbf{y_2};\mathbf{x_1},\mathbf{y_1}) \leq \min_{\mathbf{y}\in\mathbf{Y}} \big(C_\psi(\mathbf{y};\mathbf{x_1},\mathbf{y_1}) - l(\mathbf{y},\mathbf{y_2})\big) \tag{10}$$

The margin term $l(\mathbf{y},\mathbf{y_2})$ is often chosen to be a measure of dissimilarity between $\mathbf{y}$ and $\mathbf{y_2}$, for example, Euclidean distance:

$$l(\mathbf{y},\mathbf{y_2}) = \text{Dist}(\mathbf{y},\mathbf{y_2}) \tag{11}$$

One can rewrite the constraint from (10) as a loss that penalizes the cost of the ground-truth while maximizing the cost of the worst-offender:

$$\text{loss}(\psi) = \Big[C_\psi(\mathbf{y_2};\mathbf{x_1},\mathbf{y_1}) - \Big(\min_{\mathbf{y}\in\mathbf{Y}} C_\psi(\mathbf{y};\mathbf{x_1},\mathbf{y_1}) - l(\mathbf{y},\mathbf{y_2})\Big)\Big]^+ \tag{12}$$

where $[\cdot]^+ = \max(\cdot, 0)$. The minimization reaches the minimum at 0 when (10) is satisfied.

Importantly, [46] query additional supervision in the form of object's bounding boxes. These bounding boxes in future frames are converted to a binary object occupancy grid, denoted by O, as visualized by red in Fig. 2. If $O[\mathbf{u}] = 1$, there is an object occupying spacetime voxel $\mathbf{u}$. Any trajectory $\mathbf{y}$ that appears at such spacetime voxels should bear an additional *margin* cost for collisions:

$$l(\mathbf{y},\mathbf{y_2}) = \text{Dist}(\mathbf{y},\mathbf{y_2}) + \gamma_o \sum_{\mathbf{u}\in\mathbf{y}} O[\mathbf{u}] \tag{13}$$

where γ_o is a predefined cost of object collision. Instead of relying on human annotations, our approach extracts supervision from raycasted future-freespace, as visualized by green in Fig. 2.

$$l(\mathbf{y},\mathbf{y_2}) = \text{Dist}(\mathbf{y},\mathbf{y_2}) + \gamma \sum_{\mathbf{u}\in\mathbf{y}} [\text{ray}(\mathbf{u};\mathbf{x_2},\mathbf{y_2})]^+ \tag{14}$$

where γ is a predefined cost of future freespace violation.

4. Experiments

We use CARLA to evaluate freespace forecasting as a task itself. We use both NoCrash and nuScenes to evaluate planning performance. On one hand, NoCrash offers an interactive environment where an agent's action has lasting consequences, allowing for on-policy evaluation; on the other hand, nuScenes offers real-world sensor data and driving scenarios, allowing for realistic off-policy evaluation.

CARLA and NoCrash: CARLA is an open-source urban driving simulator [13] and NoCrash is the latest planning benchmark on CARLA [8]. On NoCrash, an agent succeeds if it completes a predefined route in time without collisions. NoCrash features various towns, weather conditions, and traffic densities. Town 1 is for training and Town 2 is for testing. A subset of weather conditions is also held out for testing. An agent has access to a sensor suite and needs to produce control signals to apply to motors.

nuScenes: nuScenes is one of the latest real-world driving datasets collected by autonomous fleets. We choose nuScenes because its unique release of CAN bus data makes it possible to implement our baseline motion planner [46]. Since the official server does not evaluate planning, we create a protocol for nuScenes to evaluate planning. We randomly split the 850 annotated scenes into training (550), validation (150), and test sets (150), which amounts to about 17K, 5K, 4K frames respectively.

We refer readers to the supplementary materials for implementation details such as neural net architectures. We plan to make our implementation publicly available.

α, θ	Town 1(val)			Town 2(test)		
	BCE	F1	AP	BCE	F1	AP
$(\alpha_0, \mathbf{0})$	0.594	0.659	0.453	0.377	0.740	0.561
$(\alpha, \mathbf{0})$	0.101	0.688	0.610	0.089	**0.786**	0.730
$(\mathbf{0}, \theta)$	0.043	0.687	0.752	0.099	0.406	0.364
(α, θ)	**0.034**	**0.772**	**0.830**	**0.047**	0.755	**0.773**

Table 1: Ablation studies on residual forecasting. $(\alpha_0, \mathbf{0})$ replicates the latest frame as the future prediction, without any learning. $(\alpha, \mathbf{0})$ learns to linearly predict from historical freespace observations, dramatically improving accuracy on towns used for training, but reducing performance in new towns. $(\mathbf{0}, \theta)$ directly learns a nonlinear predictor without residual, which appears to heavily overfit to the training town. (α, θ) learns a nonlinear residual that is added to the linear prediction, which outperforms all variants.

#Logs	Town 1(val)			Town 2(test)		
	BCE	F1	AP	BCE	F1	AP
25	0.051	0.692	0.707	0.051	0.759	0.752
50	0.046	0.710	0.740	0.049	**0.760**	0.760
100	0.042	0.731	0.773	0.048	**0.760**	0.768
200	0.037	0.754	0.805	0.048	0.757	0.772
400	**0.034**	**0.772**	**0.830**	**0.047**	0.755	**0.773**

Table 2: Ablation studies on increasing the amount of training data. We see a dramatic improvement on towns used for training and a slower improvement on new towns in AP.

4.1. Freespace Forecasting

Setup: We let a driving agent roam around in Town 1 under an autopilot policy [8] to collect 400 trajectories for training and 100 trajectories for validation. We follow the same practice to collect 100 trajectories in Town 2 for testing. In total, we have about 164K frames for training, 39K for validation, and 31K for testing.

Evaluation: Under the binary classification formulation, two classes in freespace forecasting are highly imbalanced. Compared to occupied space, freespace constitutes a vast majority of freespace states in the future at a rate of 35 to 1. Therefore, we plot a precision-recall curve w.r.t. occupied space and compute average precision. We also evaluate the maximum F1 score on the PR curve.

Residual forecasting: We test the idea of residual forecasting, including four variants as shown in Tab. 1. Our results suggest residual forecasting is highly effective. Please refer to the caption for details.

Scalability: Learning to forecast freespace requires no human annotation. We evaluate the performance of freespace forecasting models trained with an increasing amount of data. Our results in Tab. 2 suggest performance in training towns improves dramatically as we increase the amount of training data. The slower improvement in new towns suggests we should collect data in new towns as well. This would be particularly viable for freespace forecasting as it does not need additional human annotations.

4.2. Planning on NoCrash

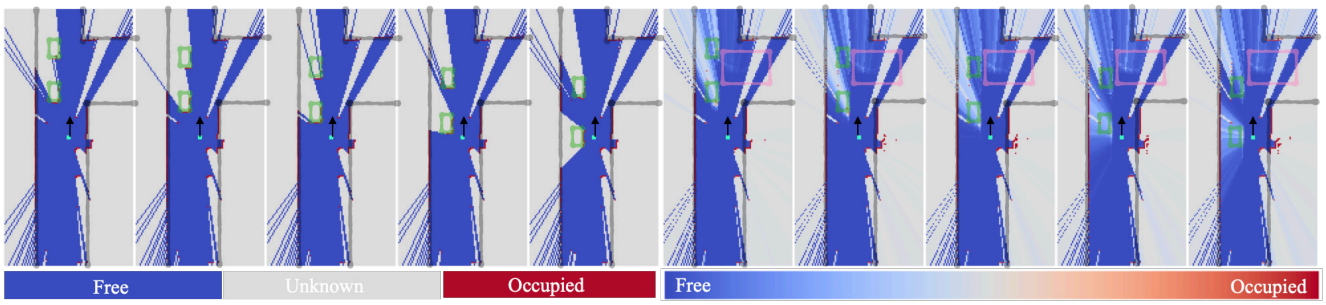
Baseline: LBC [6] is the state-of-the-art planner on NoCrash. At test time, an LBC agent receives sensor data and a high-level instruction (*turn-left, turn-right, go-straight, follow-lane*) as input every 0.1s and outputs a trajectory in the form of a series of bird's-eye-view waypoints. In particular, there are 5 waypoints from 0.5s to 2.5s at every 0.5s. The planner then uses heuristics to select one waypoint and translates it via pure-pursuit controllers to control signals that can be applied to motors for a duration of 0.1s before re-planning when new sensor data and instructions arrive.

Results: We learn a residual forecasting model that takes historical freespace from the past 2s and predicts future freespace up to 2.5s. We incorporate future freespace predicted by this forecasting model into an off-the-shelf LBC planner in a *post hoc* fashion. Based on the predicted bird's-eye-view future freespace (Fig. 3), we check if LBC's waypoint passes the test according to Eq. (4). We identify relevant voxels by drawing an oriented box for the ego-vehicle centered at the selected waypoint. When the test fails, we override LBC's plan with a trajectory that represents the action of staying still, which translates to control signals that correspond to an emergency brake through the controllers.

As Tab. 3 shows, such *post hoc* integration significantly improve the overall success rate on most testing suites compared to LBC's off-the-shelf performance. Also, our freespace-forecasting-based intervention decreases jerk in training weathers but increases jerk in testing weathers, indicating that our current evasive maneuver might not be optimal and soft braking might be preferred. Finally, our approach tends to achieve higher route completion rates compared to the baseline, suggesting freespace forecasting does not simply halt movement to reduce collision rates. We further break down the remaining failures in Fig. 4. When incorporating forecasted freespace *post hoc*, we dramatically reduce the number of collisions, converting some to timeouts. This suggests avoiding imminent collisions is not enough to guarantee successful long-term planning.

4.3. Planning on nuScenes

Baselines: Neural motion planner (NMP) [46] is a state-of-the-art planner on real-world driving data. Since there is no official implementation for NMP, we reimplement it based on details from the paper and with help from the authors. The planner first samples a list of plausible trajectories based on the ego-vehicle's kinematic state. Then it



(a) Ground-truth future freespace

(b) Predicted future freespace

Figure 3: Freespace forecasting qualitative result (bird's-eye-view). On the left, we visualize ground-truth future freespace. occupied is red, unknown is gray, freespace is blue. On the right, we visualize predicted future freespace. The model does not treat the unknown as a separate class, instead, it predicts a probability for every voxel. We highlight two vehicles in the opposing lane with green boxes. Notice the predicted freespace tracks the first (bottom) vehicle. Also, it predicts that the second vehicle could have turned left, shown by the predicted freespace in the pink box. We urge readers to view a video version of this figure (and other results) in our supplement.

Task	Weather	Success rate (%)			Jerk (m/s^3)		Route completion(%)	
		PV [6]	LBC [6]	LBC+FF	LBC [6]	LBC+FF	LBC [6]	LBC+FF
Empty	Train	100 ± 0	**100** ± 1	99 ± 1	**6.9** ± 0.0	**6.9** ± 0.1	**100** ± 0	99 ± 0
Regular	Train	95 ± 1	94 ± 3	**96** ± 2	7.8 ± 0.1	**7.5** ± 0.2	97 ± 2	**99** ± 1
Dense	Train	46 ± 8	51 ± 3	**57** ± 4	10.3 ± 0.3	**8.1** ± 0.2	79 ± 2	**80** ± 2
Empty	Test	100 ± 0	**70** ± 4	66 ± 3	**7.3** ± 0.1	7.5 ± 0.2	**86** ± 3	83 ± 2
Regular	Test	93 ± 1	62 ± 2	**73** ± 1	**8.4** ± 0.6	9.5 ± 0.7	82 ± 2	**87** ± 2
Dense	Test	45 ± 6	39 ± 6	**44** ± 5	**11.0** ± 0.8	12.0 ± 0.8	67 ± 3	**71** ± 4

Table 3: Planning results on CARLA NoCrash (test town). PV: privileged agent (see [6]). LBC: Learning By Cheating (our baseline planner on CARLA). LBC+FF: we combine a learned freespace forecaster (FF) with LBC and override any plan of LBC that "collides" with the predicted future freespace. By using forecasted freespace, we significantly improve the state-of-the-art planner LBC's overall success rates on most test suites, in particular, on those test suites that have moving objects. On empty towns, using forecasted freespace leads to slightly worse performance, likely due to false positives in freespace forecasting. Fig. 4 further breaks down failures into collisions versus timeouts, demonstrating that freespace is even more beneficial for avoiding safety-critical collisions.

Gray : new approaches from this work.

takes sensor data from the last 2 seconds as input and predicts a space-time bird's-eye-view cost map for the next 3 seconds, including 6 time-slices from 0.5s to 3.0s at every 0.5s. Finally, it scores every sampled trajectory according to the predicted cost maps and pick the best scoring one.

We implement three NMP baselines. First, we implement a *vanilla* NMP, where it penalizes trajectories based on how much they deviate from expert trajectories, as described in (11). Second, we implement an *object-supervised* NMP, where it applies additional penalties to trajectories that collide with object box occupancy in space-time, as described in (13). This baseline serves as a faithful reimplementation of [46] given what is available on nuScenes ([46] also applies penalties based on the real-time traffic light status which is not available on nuScenes). Third, we implement a NMP with improved object supervision. Specifically, we modify binary object occupancy grid O as in (13) by performing raycasting over O. This gives us a bird's-eye-view occlusion patterns imposed by object occupancy, based on a heuristics that the ego-vehicle should learn to stay away from not only the spacetime voxels that are *occupied* by objects but also those that are *occluded* by objects.

Evaluation: We evaluate planned trajectories within a 3s horizon at every 0.5s. We focus on two evaluation metrics: L2 error and collision rate. First, we compute the difference between the planned and the expert trajectory by the average L2 error between corresponding waypoints. Second, we evaluate how often the ego vehicle would collide with other objects. We place oriented bird's-eye-view boxes that represent the ego-vehicle at every waypoint on the planned trajectory and detects if there is any collision with other annotated boxes in the scene. One caveat for this evaluation is that we assume the scene plays out as recorded.

Results (L2P): We compare different learning-to-plan

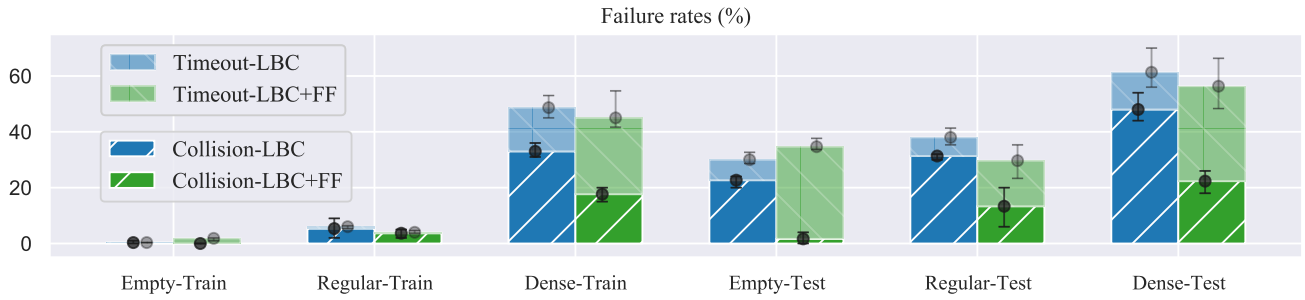


Figure 4: Breaking down NoCrash failures. Our approach of incorporating freespace forecasting into off-the-shelf black-box planners reduces overall failure rates in most scenarios. Most importantly, it dramatically reduces the collision rates by converting many collisions to timeouts. This suggests that avoiding imminent collisions is not enough to assure the success of long-term goal-reaching planning. For example, an autonomous vehicle would not want to drive on the opposing lane even if it does not lead to any collision in the near term.

Learn to plan	L2 (m)			Collision (%)		
	$1s$	$2s$	$3s$	$1s$	$2s$	$3s$
vanilla	**0.50**	**1.25**	**2.80**	0.68	0.98	2.76
+ object	0.61	1.44	3.18	**0.66**	0.90	2.34
+ object*	0.61	1.40	3.16	0.71	**0.81**	**1.45**
+ freespace	0.57	1.28	2.94	**0.66**	0.87	2.17
+ freespace*	0.59	1.35	3.07	0.74	0.93	1.65

Table 4: We compare learning-to-plan approaches with different additional supervision (+) on the held-out test set of nuScenes. The *vanilla* approach learns with ego-vehicle trajectories as the only source of supervision, achieving the lowest L2 errors but the largest collision rates. Learning with additional *object* (especially *object** – improved) supervision significantly reduces the collision rates. Learning with additional *future freespace* supervision reduces collision rates without requiring human annotations. Note that vanilla+object represents a faithful reimplementation of neural motion planning (NMP) [46].

Red: approaches that need human annotations

Plan w/ FF	L2 (m)			Collision (%)		
	$1s$	$2s$	$3s$	$1s$	$2s$	$3s$
vanilla	**0.50**	**1.25**	**2.80**	0.68	0.98	2.76
→ FF (dense)	0.57	1.34	3.18	0.66	0.98	2.43
→ FF (sparse)	0.56	1.27	3.08	**0.65**	**0.86**	**1.64**

Table 5: We evaluate planning-with-forecasted-freespace approaches on the test set of nuScenes. We compare two loss functions for freespace forecasting: *dense* (2) and *sparse* (8). The *sparse* loss is strictly better than the *dense* loss, suggesting freespace forecasting is more effective at helping planning when it is aware of what is reachable space for the planner. Also, post-processing a *vanilla* baseline with *planning-aware* freespace forecasting can largely bridge the gap toward learning with *object*-supervision* (Tab. 4).

approaches. As Tab. 4 shows, the *vanilla* baseline achieves the lowest L2 errors but larger collision rates compared to *object-supervised* baselines. Learning with *improved* object-supervision leads to the lowest collision rates, suggesting staying away from object occlusion is a good heuristics when learning cost maps for planning. Finally, learning with future freespace reduces collision rates compared to the *vanilla* baseline without requiring human annotations.

Results (Plan w/ FF): We evaluate approaches that post-process the *vanilla* baseline with forecasted freespace. We explore both *dense* and *sparse* loss for learning the freespace forecasting model. As Tab. 5 shows, post-processing with planning-aware freespace forecasting (FF) greatly reduces the *vanilla* baseline's collision rates, and largely bridge the gap toward the best *object-supervised* baseline (Tab. 4). Interestingly, post-processing with planning-aware freespace forecasting (8) turns out more effective than learning to plan with future freespace. We posit that max-margin learning does not take full advantage of future freespace by penalizing only the worst offender.

Conclusion: Standard approaches to planning in a dynamic environment usually require an object-centric perception system that is trained to forecast the future evolution of the scene. Providing object annotations is an expensive venture that cannot scale to the magnitude of data generated by autonomous fleets. We introduce self-supervised *future freespace forecasting* as an *annotation-free*, scalable representation for safe, expert-like motion planning and show that it serves as an effective augmentation to standard methods. In practical settings, future freespace forecasting is versatile because it can (1) be directly incorporated into existing learning-based approaches for motion planning or (2) be used as an additional post hoc predictive collision-checking step on top of an existing motion planner.

Acknowledgments: This work was supported by the CMU Argo AI Center for Autonomous Vehicle Research.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004. 2

[2] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018. 3

[3] Fethi Belkhouche. Reactive path planning in a dynamic environment. *IEEE Transactions on Robotics*, 25(4):902–911, 2009. 2

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 2, 3

[5] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015. 2

[6] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020. 2, 3, 6, 7

[7] Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018. 2, 3

[8] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9329–9338, 2019. 2, 3, 5, 6

[9] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary R Bradski. Self-supervised monocular road detection in desert terrain. In *Robotics: science and systems*, volume 38. Philadelphia, 2006. 3

[10] Julie Dequaire, Peter Ondrúška, Dushyant Rao, Dominic Wang, and Ingmar Posner. Deep tracking in the wild: End-to-end tracking using recurrent neural networks. *The International Journal of Robotics Research*, 37(4-5):492–512, 2018. 3

[11] Pranaab Dhawan and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. 3

[12] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016. 3

[13] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017. 5

[14] Frederik Ebert, Chelsea Finn, Alex X Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. *arXiv preprint arXiv:1710.05268*, 2017. 3

[15] Nico Engel, Stefan Hoermann, Philipp Henzler, and Klaus Dietmayer. Deep object tracking on dynamic occupancy grid maps using rnns. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3852–3858. IEEE, 2018. 3

[16] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):267–282, 2007. 3

[17] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. 2

[18] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering free space of indoor scenes from a single image. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2807–2814. IEEE, 2012. 3

[19] Michael Himmelsbach, Felix V Hundelshausen, and H-J Wuensche. Fast segmentation of 3d point clouds for ground vehicles. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 560–565. IEEE, 2010. 4

[20] Stefan Hoermann, Martin Bach, and Klaus Dietmayer. Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2056–2063. IEEE, 2018. 3

[21] Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018. 3

[22] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010. 2

[23] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996. 2

[24] Michelle A Lee, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8943–8950. IEEE, 2019. 3

[25] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3288–3295. IEEE, 2019. 3

[26] Nima Mohajerin and Mohsen Rohani. Multi-step prediction of occupancy grid maps with recurrent neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3

[27] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2146–2153. IEEE, 2017. 3

[28] Peter Ondruska and Ingmar Posner. Deep tracking: Seeing beyond seeing using recurrent neural networks. *arXiv preprint arXiv:1602.00991*, 2016. 3

[29] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018. 2

[30] AJ Patel. 2d visibility: wall tracking, 2012. 4

[31] Stéphane Petti and Thierry Fraichard. Safe motion planning in dynamic environments. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2210–2215. IEEE, 2005. 2

[32] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989. 2, 3

[33] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pages 489–494. IEEE, 2009. 5

[34] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736, 2006. 2, 5

[35] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*, 2018. 2, 3

[36] Thomas Roddick and Roberto Cipolla. Predicting semantic map representations from images using pyramid occupancy networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11138–11147, 2020. 3

[37] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011. 3

[38] Young-Woo Seo, Nathan Ratliff, and Chris Urmson. Self-supervised aerial images analysis for extracting parking lot structure. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009. 3

[39] Boris Sofman, Ellie Lin, J Andrew Bagnell, John Cole, Nicolas Vandapel, and Anthony Stentz. Improving robot navigation through self-supervised online learning. *Journal of Field Robotics*, 23(11-12):1059–1075, 2006. 3

[40] Anthony Stentz. Optimal and efficient path planning for partially known environments. In *Intelligent Unmanned Ground Vehicles*, pages 203–220. Springer, 1997. 2

[41] Anthony Stentz et al. The focussed dˆ* algorithm for real-time replanning. In *IJCAI*, volume 95, pages 1652–1659, 1995. 2

[42] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008. 1

[43] Jur P Van Den Berg and Mark H Overmars. Roadmap-based motion planning in dynamic environments. *IEEE Transactions on Robotics*, 21(5):885–897, 2005. 2

[44] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015. 2

[45] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4238–4245. IEEE, 2018. 3

[46] 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. 2, 5, 6, 7, 8

[47] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008. 2