Hybrid Imitative Planning with Geometric and Predictive Costs in Offroad Environments

Anonymous Author(s)
Affiliation
Address
email

Abstract

Mobile robots tasked with reaching user-specified goals in open-world outdoor environments must contend with numerous challenges, including complex perception and unexpected obstacles and terrains. Prior work has addressed such problems with geometric methods that reconstruct obstacles, as well as learning-based methods. While geometric methods provide good generalization, they can be brittle in outdoor environments that violate their assumptions (e.g., tall grass). On the other hand, learning-based methods can learn to directly select collision-free paths from raw observations, but are difficult to integrate with standard geometry-based pipelines. This creates an unfortunate "either-or" dichotomy – either use learning and lose out on well-understood geometric navigational components, or do not use it, in favor of extensively hand-tuned geometry-based cost maps. The main idea of our approach is reject this dichotomy by designing the learning and non-learning-based components in a way such that they can be easily and effectively combined and created without labeling any data. Both components contribute to a planning criterion: the learned component contributes predicted traversability as rewards, while the geometric component contributes obstacle cost information. We instantiate and comparatively evaluate our system in a high-fidelity simulator. We show that this approach inherits complementary gains from both components: the learning-based component enables the system to quickly adapt its behavior, and the geometric component often prevents the system from making catastrophic errors.

1 Introduction

How can we enable a robot to swiftly traverse open-world environments while minimizing heuristic and time-intensive hand-engineering, like those depicted in Fig. 1? The robot should receive coarse goal direction from a human supervisor, and use this direction, along with its sensor suite and prior experience, to make its own decisions about what actions to take to reach the destination. A solution to this problem would enable users to direct robots across unfamiliar territory without requiring them to significantly change or tune components of the system. For example, imagine a rescue worker tasking a search-and-rescue robot to quickly search a series of locations in an unmapped dense forest.

A major challenge to developing such a system is enabling it to both draw on prior experience and adapt its behavior to new environments. While learning is a powerful way to deal with open-world environments, most learning-based methods studied are difficult to integrate with non-learning-based navigation pipelines.

This creates an unfortunate "either-or" dichotomy – either use learning and lose out on well-understood navigational components, or do not use it, in favor of extensively hand-tuned geometry-based cost maps. The main idea of our approach is reject this dichotomy by designing the learning

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

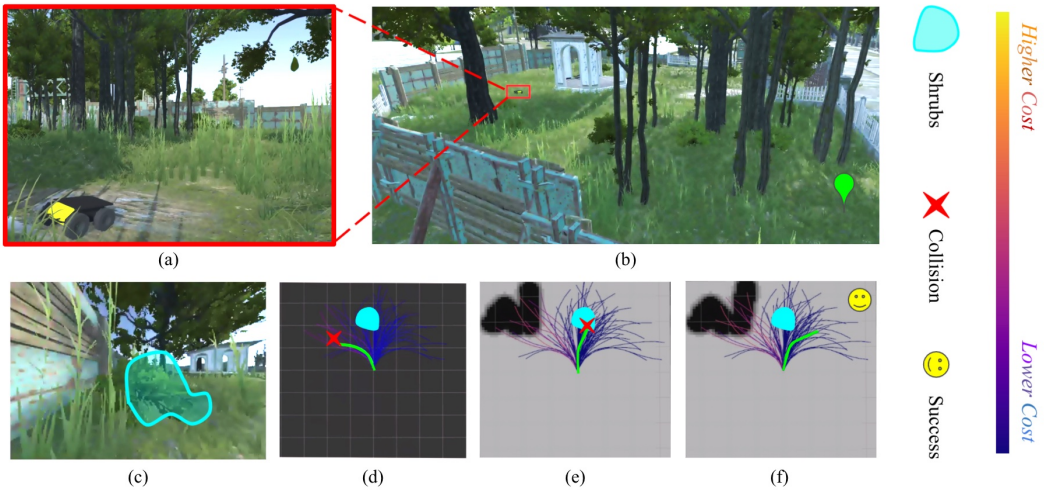

Figure 1: Our approach combines a learning-based method and a geometry-based planner to plan obstacle-free trajectories for a Husky robot in a deployment area with obstacles (trees, grass, bushes) as seen in **(a)**. An example navigation task is shown in **(b)**, where the robot (marked in red) must navigate to the goal (marked in green) while avoiding obstacles on its path. On its path to the goal, the robot often encounters a scenes like **(c)** where there are non-traversable elements like shrubs (highlighted in cyan) and novel obstacles like a wall. Learning-based methods successfully avoid previously seen obstacles but can struggle with novel obstacles, leading to a collision with the wall **(d)**. While a geometric approach can avoid this by predicting the collision, it is unable to reason about the traversability of large shrubs that are hard to identify **(e)**. Note that the cyan markers are shown for illustration only and are not available to the robot. Our approach, HIP, combines attributes from both these methods and successfully plans a collision-free trajectory **(f)**. Videos of our results are hosted at https://sites.google.com/view/hybrid-imitative-planning/.

and non-learning-based components in a way such that they can be easily and effectively combined and created without labeling any data. Fig. 1 depicts this synthesis.

One classic approach is to perform online geometric mapping and traversability estimation and then use these estimates, along with a hand-tuned cost function, for planning feasible paths [6, 28, 17, 18]. With careful design and incorporation of prior knowledge into the decision-making pipeline, these approaches, which we term "geometry-based" and "geometric costmap"-based, are a standard and performant approach. Their main drawback, however, is a general inability to automatically incorporate the robot's prior experience and improve its decision-making capability. For example, if the costmap assumes all densely-populated points correspond to rigid and untraversable obstacles, traversable tall grass may cause the robot to be too conservative and avoid the grass; if a height threshold is included to compensate for this effect, then small, untraversable obstacles like rocks and shrubs may cause the robot to be too aggressive and get stuck on the rocks or shurbs. Another approach to robot navigation is learning-based, e.g. goal-conditioned imitation learning and policy-based reinforcement learning, yet these are difficult to integrate with real-world robotic systems because they are inscrutable and directly output actions. Can we build on both lines of work by addressing their drawbacks in a principled way?

The main insight in our work is that, instead of trying to reconcile conflicting *actions* commanded by geometry-based and learning-based components, we can instead utilize both components to contribute terms to a shared navigational cost function. Essentially, this cost function represents the hypotheses of both methods about which future trajectories will or will not lead to collisions. Once a shared cost function combining geometry-based and learning-based reasoning has been produced, a standard planning method can decide on the best path to take informed by this cost. We summarize this idea, and the intuition for why the geometry-based and learning-based components might provide complementary strengths, in Fig. 2. We instantiate and comparatively evaluate our system in a high-fidelity simulator with a Husky robot. We show that this approach inherits gains from both components: the learning-based component enables the system to learn subtle associations between the high-dimensional sensor data and traversability, and the learning-free component reduces the frequency of catastrophic errors when the learner fails to generalizesites.google.com/view/hybrid-imitative-planning/.

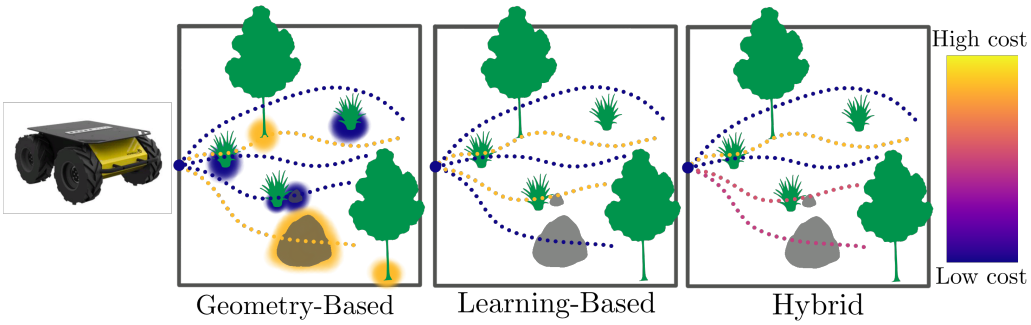

Figure 2: Illustration of example trajectory plan costs: The geometric costmap approach is adept at identifying obstacles like large trees and rocks, but it may fail to assign high costs to terrain with impediments that are only visually perceivable, since the LiDAR input often misses small obstacles. The learning-based planner uses its prior experience to reason about the traversability of objects and terrain with which it is experienced, such as traversable grass, small impassable objects, and trees. However, it may make mistakes when it encounters drastically novel obstacles, such as painted walls or automobiles. By designing each of these components in a way that is amenable to combination, the hybrid planner can accrue benefits from each of its components. In this example, it adeptly identifies large obstacles and incorporates the data-driven experience from the learned planner to estimate that the direct plan across the small, compliant obstacles is traversable.

2 Hybrid Imitative Planning with Geometric and Predictive Costs

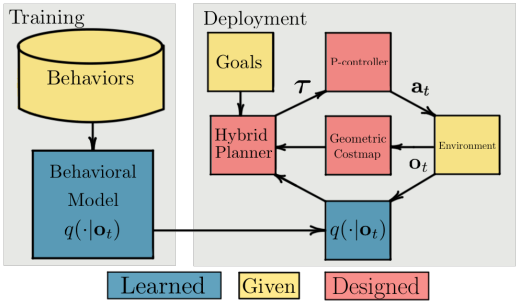

Figure 3: HIP system diagram: A dataset of trajectory demonstrations is used to learn a model of traversability and behavior. The model is combined with a geometric costmap into the hybrid planner, which plans a trajectory towards a received goal.

In our problem setting, a mobile robot receives high-dimensional sensory observations and global localization information, and is tasked with using these sources of information to navigate to a provided goal in an unmapped, offroad environment partly populated with untraversable terrain (e.g. large and small obstacles, steep hills). Let $\mathbf{o}_t = (\mathbf{i}_t, \mathbf{l}_t, \mathbf{x}_t)$ denote the observations that are available to the robot, where $\mathbf{x}_t \in \mathbb{R}^3$ is the robot's position estimated by odometry, $\mathbf{i}_t \in [0,1]^{H_i \times W_i \times 4}$ is an RGB-D image, and $\mathbf{l}_t \in \mathbb{R}^{N_t \times 3}$ is a point cloud provided by a LiDAR sensor. Let $\mathbf{d} \in \mathbb{R}^2$ denote the provided goal. Let $\boldsymbol{\tau} \in \mathbb{R}^{2H}$ denote a trajectory of potential future positions: $\boldsymbol{\tau} \doteq \mathbf{x}_{t+1:t+H}$, and $\boldsymbol{\tau}_i$ denote $\mathbf{x}_{t+i}$.

Our main idea is to design a system that allows learned models to *easily integrate* with standard geometric reconstruction and planning pipelines. Learned costs and represented as a *function of trajectories* and learned via density estimation to imitate collision-free data, and geometric costs are represented as a *function of positions*, which can be composed into a function of trajectories, and then combined into a combined single planning criterion. Finally, destination directives are represented as positions and included as a final cost. We term our method Hybrid Imitative Planning (HIP), and present a system diagram in Fig. 3. While we expect the capability to integrate these separate components to be generally useful, we have some intuition about how the utility may manifest. Geometric costmap-based planning may confer safety around large, easily-sensed objects, like boulders and trees, yet it cannot incorporate subtle visual cues, and more generally, cannot be improved from additional robot experience. Learning-based planning may offer the capability to identify terrain directly estimated as traversable from experience, such as grass and small bushes, yet it may be risky around objects that appear significantly different than those in the training data. In designing these two approaches into a single system, we expect the system to be zero-shot robust to objects that appear significantly different from the training data and able to identify subtle aspects of the terrain that afford or inhibit traversability. In this section, we present an abstract overview of how geometric traversability costs can be integrated with learned models, and then present a specific instantiation of a navigation system based on this design in Section 3.

Designing geometric costmaps. We assume the geometry-based component is available as a *costmap*: $C_t(\cdot) : \mathbb{R}^2 \mapsto \mathbb{R}$. This decision enables interpretable design of geometric heuristics $g(\mathbf{l}_t) = C_t$ that process the LiDAR into a function of positions. Let us denote the composition of the costmap generation and evaluation of it at a position $\mathbf{x}_t$ as $C_{\text{costmap}}(\mathbf{x}_t; \mathbf{o}_{\leq t}) \doteq g(\mathbf{l}_t)(\mathbf{x}_t)$. We defer discussion of how g is implemented to Section 3.

Designing learned costs. In order to combine geometric and learned costs, we must construct a learner that processes the high-dimensional sensor data to assigns costs to trajectories. Our learning-based costs aim to *predict whether a given trajectory would be collision-free* – whether it corresponds to a traversable path. Formally, we construct the learning-based component as a conditional *probability density function* of trajectories $\boldsymbol{\tau}$: $q(\boldsymbol{\tau}|\mathbf{o}_{\leq t}) : \mathbb{R}^{2H} \mapsto \mathbb{R}$, and learn q from data that excludes collisions. By satisfying this assumption and training q via a maximum likelihood objective, q should assign high values to a subset of the *traversable* trajectories and low values to this subset's complement, which will include *untraversable* trajectories. Given q, we can construct costs as $C_{\text{learned}}(\boldsymbol{\tau}; \mathbf{o}_{\leq t}) \doteq -\log q(\boldsymbol{\tau}|\mathbf{o}_{\leq t})$. Note that because q is a function of trajectories, not positions, it has a higher modeling capacity than if it were parameterized by a costmap. We defer further discussion of how we learn q to Section 3.

Designing directive costs. In order to direct the robot to the final desired destination ($\mathbf{d}$), we incorporate a cost that encourages forward progress towards the final destination. Denote this cost $C_{\text{directive}}(\boldsymbol{\tau}; \mathbf{o}_{\leq t}, \mathbf{d})$. We defer discussion of how we construct $C_{\text{directive}}$ to Section 3.

Combining geometric costmaps and learned costs. Given our goal of integrating learning-based and geometry-based planning, we design π to use a receding-horizon state-space planner as shown in Eq. (1):

$$\boldsymbol{\tau}^* \doteq \underset{\boldsymbol{\tau} \in \mathbb{R}^{2H}}{\operatorname{argmin}} \left[C_{\text{directive}}(\boldsymbol{\tau}; \mathbf{d}) + (1-\phi) C_{\text{learned}}(\boldsymbol{\tau}; \mathbf{o}_{\leq t}) + \phi \sum_{i=1}^{H} C_{\text{costmap}}(\boldsymbol{\tau}_i; \mathbf{o}_{\leq t}) \right] \tag{1}$$

Given a planned trajectory, $\boldsymbol{\tau}$, we use it to compute $\mathbf{a}_t$ as a simple position-tracking PID controller, written as $\mathbf{a}_t = f(\boldsymbol{\tau})$.

3 Instantiating the System

Implementing learned costs. As previously discussed, our learned costs are designed to be a conditional probability density function of future trajectories, $q(\boldsymbol{\tau}|\mathbf{o}_{\leq t})$ fit to data that excludes collisions. We implement q as a "Deep Imitative Model" [21]. We adapted the open-source PyTorch implementation of imitative models released in Filos et al. [7] and adapted the input to use RGB-D. This model uses a MobileNetV2 encoder [23]. During training, we follow the method of Rhinehart et al. [20] to induce an upper-bound, η, on q by perturbing the training trajectories with a Gaussian. We defer further data-dependent implementation details, including a full table of hyperparameters, to Section 5.

Implementing geometric costmaps. We use a LiDAR sensor to produce a local point cloud, from which a terrain traversal cost is computed. We use the "costmap_2d" implementation provided by ROS to compute a discrete 2D costmap for the ground plane, denote C'_{costmap} [19]. We apply a nonlinear transformation to C'_{costmap} so that its maximum value corresponds to η/H. This ensures $\sum_{i=1}^{H} C_{\text{costmap}}(\tau_i) \leq \max_{\boldsymbol{\tau}} q(\boldsymbol{\tau}|\mathbf{o}_{\leq t})$, which makes tuning the ϕ parameter simpler, as the costs are roughly in the same range. This transformation is given by $C_{\text{costmap}}((x,y)) = \alpha \cdot \exp C'_{\text{costmap}}((x,y)) / \sum_{x',y'} \exp C'_{\text{costmap}}((x',y'))$.

Implementing directive costs. Similar to the region goals described by Rhinehart et al. [21], we designed a directive cost that penalizes $\boldsymbol{\tau}$ with cost δ if it does not end within a particular region a short distance away from the robot, in the direction of $\mathbf{d}$. We construct this region as a rectangle with width 2m, center axis in the direction of $\mathbf{d}$, offset towards $\mathbf{d}$ by 3m.

Fast planning. In order to quickly solve the optimization problem in Eq. (1), we employ a trajectory library with size with $K = 200$. The future trajectories (described in Section 5, were clustered using k-means, and the centroid, $\bar{\boldsymbol{\tau}}^k$, of each cluster was appended to a library, $\mathcal{K} = \{\bar{\boldsymbol{\tau}}^k\}_{k=1}^{K}$. During inference, we perform approximation optimization of Eq. (1) using $\operatorname{argmin}_{\boldsymbol{\tau} \in \mathcal{K}} L(\boldsymbol{\tau}; \mathbf{o}_{\leq t}, \mathbf{d})$.

System and environment overview. We instantiate our system on a Clearpath Husky UGV deployed in a photorealistic outdoor navigation simulator. The default sensor suite on the Husky includes a 6-DoF IMU, a GPS unit for approximate global position estimates and wheel encoders to estimate local odometry. We added a forward-facing wide field-of-view RGB-D camera and a LiDAR sensor. During data collection and evaluation, we heuristically detect collisions using IMU and odometry data. The simulator consists of a Unity backend and is tightly integrated with the ROS stack on the robot. The environment consists of an obstacle-rich, enclosed geofence where dense rigid trees, bushes, and traversable tall grass are scattered throughout. While the geometric costmap may correctly identify trees as hazardous, it can fail to disambiguate traversable grass and untraversable bushes. Furthermore, it does not reason about other terrain properties that may lead to collision, such as the terrain slope. In order for the robot to succeed, it must carefully navigate through dense obstacles over traversable terrain.

4 Related Work

Geometry-based goal-directed navigation. Existing methods for traversability estimation include geometry-based, appearance-based, and proprioceptive systems, as categorized by Papadakis [18]. Geometry-based methods build a 2.5D or 3D terrain map that is used to extract features, such as the maximum, minimum, and variance of the height and slope of the terrain [15, 10]. Planning algorithms for such methods can take into account the stability of the robot on the terrain [11, 28]. Since sensor and localization uncertainty can play a large role in the construction of environment maps, various methods exist for estimating the probability distributions of states based on the kinematic model of the vehicle and the terrain height uncertainty [17, 9]. For example, Fan et al. [6] construct distributions of traversability costs for risk-aware planning. These approaches rely on hand-crafted models to determine risks to the robot when traversing over various terrains. However, these models often rely on simplifying assumptions, and may not consider the compressibility or compliance of geometric features, especially with respect to vegetation [16]. In contrast, our hybrid approach allows us to leverage geometry-based goal-directed navigation in tandem with a learned model, which enables our approach to overcome misspecifications of the geometry-based heuristics.

Learning-based goal-directed navigation. A variety of learning-based methods have studied the acquisition of goal-directed behavior, broadly classed as imitation learning (IL) [4, 2, 3, 9, 25] and reinforcement learning (RL) [13, 24, 22]. Goal-conditioned IL typically requires goals to be specified during training and do not extend well beyond demonstrations. A drawback to these more general IL and RL systems is that they are difficult to interpret and incorporate into existing geometry-based goal-directed navigation pipelines. Planning and navigation in unstructured environments can be greatly improved by learning environment traversability using prior experience, but previous approaches to explicitly representing environment traversability require expensive human supervision or traversability heuristics [14, 1, 12, 5]. Recent progress in using topological graphs to implicitly reason about traversability [8, 25, 27, 26] gives a promising way to learn from prior experience but has not been demonstrated for long-range navigation in complex, unstructured environments. In contrast, our hybrid approach employs a form of self-supervised learning-based explicit *trajectory* traversability estimation in tandem with geometry-based *positional* traversability estimation, which enables our approach to more robustly deal with previously-unseen complex obstacles and terrains.

5 Experiments

We designed our experiments to answer the following questions: **Question 1:** Can our combined approach achieve collision-avoidance and navigation performance superior to its constituent components? **Question 2:** How does varying input modality and component weighting affect HIP?

To answer these questions, we measure the rate of navigational success for randomized start-goal pairs in two areas in the simulator (which was described in Section 3). The first area is the original environment in which data was collected, and the second area is an out-of-distribution environment containing previously-unseen obstacles. In each environment, a method's success rate quantifies the performance of the model when globally directed to a determined goal; specifically, how often it succeeds in reaching the goal unharmed. We report a normalized version of this metric relative to the performance of a planner operating with a coarse-grained global obstacle map, which achieved a success rate of 0.72 in the first area, and 0.66 in the second area.

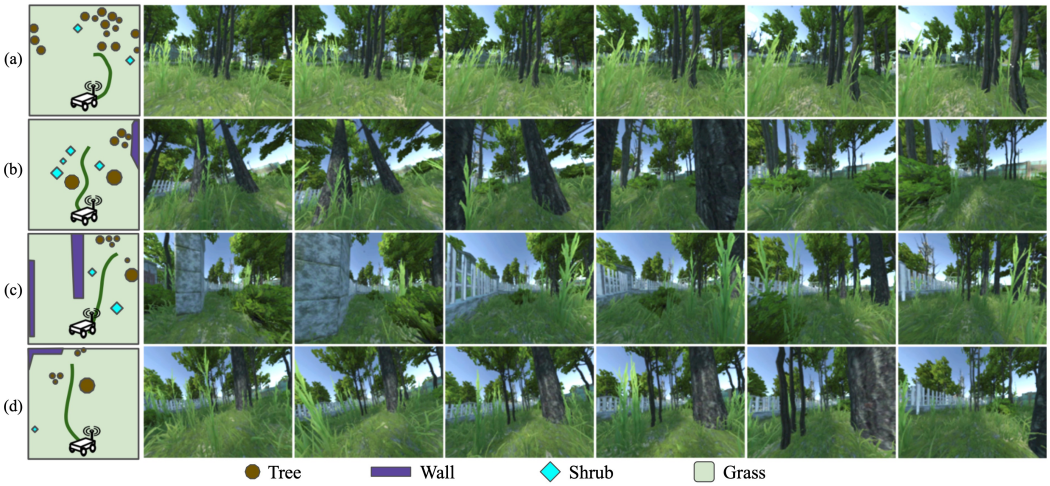

Figure 4: Example HIP rollouts: We show egocentric rollouts of our method in four challenging scenes illustrated on the left (note that this view is not available to the robot). **(a)** The robot navigates through grass and avoids trees. **(b)** The robot navigates between trees and shrubs. **(c)** The robot avoids a wall and a shrub while traversing tall grass. **(d)** The robot traverses grass and between trees.

This global planner failed sometimes due to leading the robot onto hills on which it capsized and into inescapable dead-end areas with small obstacles. This "oracle" planner violates the problem assumption of deployment to an unmapped environment, as it has access to a map.

Towards **Question 1**, we expect the geometric costmap method to struggle with smaller untraversable obstacles, thus affecting its capability to efficiently navigate given a global direction. We expect the learner, on the other hand, to be able to generate sequences of movement through areas that are perceptually similar to traversable scenes in the training data. Since any learned model is susceptible to errors under distributional shift, we might expect the learner to sometimes fail to produce collision-free trajectories when it encounters obstacles that are visually very different from those seen in the training data. Finally, we expect our proposed hybrid approach to harness the strengths of its components to outperform them with minimal hand-engineering effort. We expect the improvement to result primarily from superior obstacle-avoidance behavior.

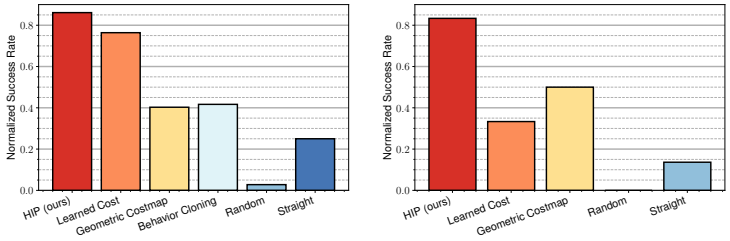

Figure 5: Goal-direction navigation success rates: HIP, which provides a simple interface to combine learned costs with a geometric costmap, improves the task performance to the next-best approach by over 10% in the in-distribution environment (*Left*) and 33% in the out-of-distribution environment (*Right*). The "Random" policy uniformly samples actions from a discretized action space. The "Straight" policy drives straight towards the goal.

Towards **Question 2**, we first investigate the effect of ϕ, which controls the importance of the component costs on the planning criterion (Eq. (1)). By varying ϕ to identify the optimal value, ϕ^*, we determine if they complement each other ($\phi^* \in (0, 1)$), or whether one dominates ($\phi^* \in \{0, 1\}$). Furthermore, through the ablation, we can understand the importance of each sensor; we expect the learner's performance to be maximized when all sensor modalities are included.

Baselines. Beyond the components of our method, we include three other baselines to further contextualize our system's performance. **Behavior Cloning (BC) baseline:** We trained a goal-conditioned BC baseline using the same data the learner used, as well as a nearly identical neural-network architecture to that of the learner. The BC baseline is trained to directly predict a control given a provided goal, rather than a probability density function over trajectories. **Straight baseline:** In order to contextualize the importance of reactivity to perceptual cues, we include a baseline that

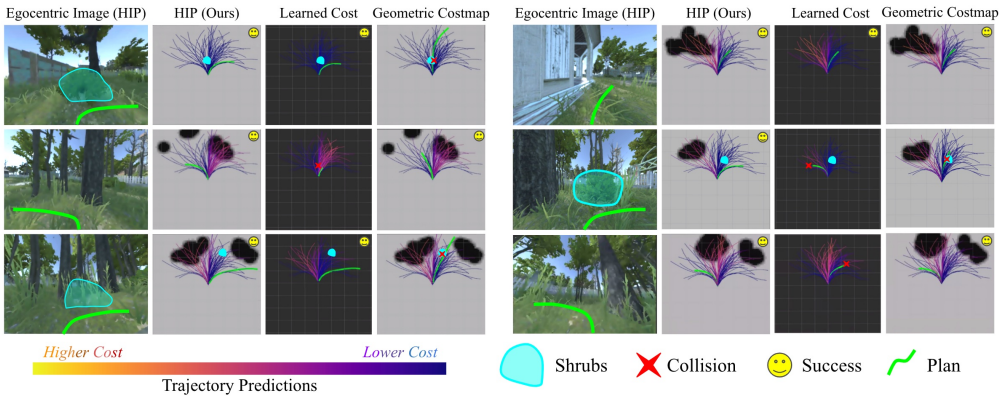

Figure 6: Qualitative results on six example scenes: We visualize the trajectory library predictions for the different methods and highlight HIP's plan (green) and shrubs (cyan) on the left. Methods which utilize a geometric costmap have plans overlaid on top of the map where black represents high cost and gray represents low cost. Since the geometric costmap cannot localize shrubs, it often plans through them and results in a collision. The planner using learned cost fails to reason in cases with multiple obstacles (like several trees) and can cause collision. Our hybrid planner combines the benefits of a reliable costmap with learned traversability to successfully plan around walls, trees and bushes.

plans an action to track the straight-line segment from the robot to the goal. This baseline does not process the perceptual data, and therefore cannot react to obstacles. **Random baseline:** Finally, we include a baseline that uniformly samples a random action from the robot's discretized action space, which further contextualizes the difficulty of the problem.

Data collection and model learning. We performed the following steps to automatically collect training data in the simulator. First, we created a geofence and a generated a set of random starting poses within it. Next, we randomized the robot's starting pose among this set and drove it with "sticky" (executed for multiple frames) random actions. As per our method's data requirements described in Section 2, we automatically removed sequences that resulted in collisions. Although the explicit collision data could be employed to refine the model, we found this method to be effective even when the collision data was discarded entirely. Collisions were heuristically identified in three different ways, and the robot was randomly respawned after any collision. The first heuristic is a *stuck collision*, which triggers when the robot is stationary for over 4 seconds. The second is a *trapped collision*, which triggers when the robot does not move over 3m over a 10s period, which prevents tight circular movement or very slow motion. The final heuristic is a *capsized collision*, which uses the IMU to identify when the robot is capsized. As part of each recorded positional trajectory, RGB-D and odometry data was recorded. To create the set of examples for training, the data was postprocessed by subsampling at 5Hz. Each example uses 10 past timesteps and a one RGB-D visual input, and is trained to predict $H = 10$ timesteps of future positions via a maximum likelihood objective. Over $200k$ examples were collected, with which our Learner was trained to maximize the likelihood of the ground truth odometry positions conditioned on the past odometry and visual information: $\max_\theta \mathbb{E}_{(\mathbf{o}_{\leq t}, \tau)} \log q_\theta(\boldsymbol{\tau}|\mathbf{o}_{\leq t})$. Table 1 summarizes the hyperparameters of our method.

Hyperparam.	Value	Meaning
f_{Sim}	30Hz	Simulator framerate
$\lvert\mathcal{D}\rvert$	$\sim$ 200k	Dataset size
f_τ	5Hz	Trajectory framerate
H	10	Learner's prediction horizon
H_{past}	10	Learner's input horizon
$H_i \times W_i$	100×100	Cropped RGB-D image dimensionality
B	32	Minibatch size
ϵ	0.001	Learning rate
σ	0.01	Scale of the training perturbation
η	64	σ-induced upper-bound of $q(\boldsymbol{\tau}\lvert\mathbf{o}_{\leq t})$
α	6.4	Costmap scaling parameter
ϕ^*	0.75	Final planner cost balance
K	200	Trajectory library size
f_L	1Hz	Replanning frequency

Table 1: Hyperparameters used in our experiments.

Method	Normalized Success Rate
Learned Cost-only, RGB-D input	0.76
Learned Cost-only, RGB input	0.60
Learned Cost-only, Depth input	0.65

Table 2: Ablation results: We observe that both RGB and Depth information are helpful to the Learner.

Experimental setup. The simulator we employed in our experiments was described in Section 3. To measure success rate, $\boldsymbol{s}_{rate}^{goal}$, the robot was randomly spawned 300 times in the geofence with a goal at least 10m away. A constant seed was utilized for generating start and goal points across each method.

Results analysis. Our primary results are shown in Fig. 5. Videos of our results are hosted at https://sites.google.com/view/hybrid-imitative-planning/. The naïve baselines – Random and Straight – illustrate the difficulty of the problem. We find our method to be the most performant. The Learner significantly outperforms the Behavioral Cloning baseline, as it represents multiple possible futures and defers goal-conditioning until test time, when the planner can select the most appropriate trajectory for the given goal. The Costmap outperforms the naïve baselines, and performs similarly to BC. It cannot independently model traversability for rigid objects shorter than grass, such as bushes or rocks, but is adept at identifying areas with larger, more distinguishable untraversable objects. While the performance ordering in the in-distribution environment is HIP $>$ Learner $>$ Costmap, the performance ordering in the out-of-distribution environment is HIP $>$ Costmap $>$ Learner. In the out-of-distribution environment, the Costmap is more robust than the Learner to out-of-distribution obstacles, yet our method still is able to leverage the Learner's performance to outperform the Costmap method. While the costmap confers robustness to previously-unseen obstacles, we observed it to fail in some cases by exhibiting oscillatory planning behavior near some obstacles until the robot reached unrecoverable positions, whereas HIPrarely produced oscillatory planning behavior, enabling it to avoid these types of unrecoverable positives. Table 2 contains the result of the ablation analysis. The RGB-D Learner outperformed both RGB and Depth alone, thus illustrating the significance of both modalities in determining traversability. In Fig. 7, the results of the ϕ hyperparameter sweep are presented. These results affirm the efficacy of both components of our method, show that multiple ϕ values are performant. In Fig. 4, we show example rollouts from our method navigating dense obstacles; we observed that it is capable of winding through complex areas with many obstacles if an open path exists. In Fig. 6, a set of qualitative examples for our method, the Learner, and the Costmap are depicted. In these examples, we often observe the Costmap to identify a subset of the impassable obstacles, while the Learner refines the remaining viable paths to account for previous odometer positions and objects perceived in the RGB-D image (which may be undetected by LiDAR). This results in the most effective navigation around visually perceivable obstacles.

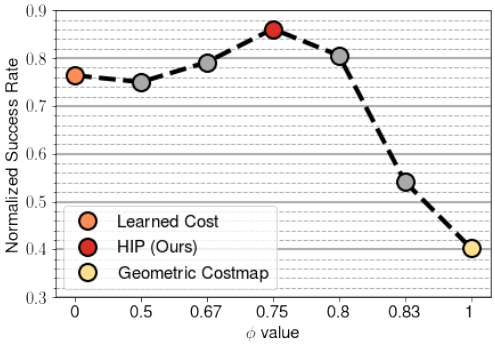

Figure 7: Success rate on varying ϕ: When no cost is incorporated ($\phi = 0$), our method purely relies on learning; when the learned component is removed ($\phi = 1$), our method purely relies on the costmap. Empirically analyzing a range of ϕ suggests that combining the components with $\phi = 0.75$ results in the best performing system.

6 Discussion

We proposed HIP, a method designed to flexibly incorporate learning-based and geometry-based components into a single cost function for goal-directed navigation in open-world off-road environments. We evaluate HIP in a high-fidelity simulator, and find that HIP shows significant improvement over both of its constituent components, as well as baselines. Ablations of sensory inputs confirm the efficacy of both RGB and depth data to the system. A hyperparameter sweep of the primary parameter illustrates the existence of multiple performant values. The effectiveness of our approach illustrates that both learning-based and geometry-based components for autonomous navigation can be integrated effectively if they contribute distinct but complementary terms to a shared cost function. That is, instead of sharing *control* between different types of methods, we simply add their contributions to a cost function used by a standard model-predictive control method. This approach suggests promising directions for future work, integrating other types of learned models, as well as additional sensory modalities. Furthermore, since our learned costs can be integrated with arbitrary goal representations into a standard planner, a promising direction is to further study other types of objectives that can be accomplished with our method.

References

[1] R. Omar Chavez-Garcia, Jerome Guzzi, Luca M. Gambardella, and Alessandro Giusti. Learning ground traversability from simulations. *IEEE Robotics and Automation Letters*, 3(3):1695–1702, Jul 2018. ISSN 2377-3774. doi: 10.1109/lra.2018.2801794. URL http://dx.doi.org/10.1109/LRA.2018.2801794.

[2] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning, 2018.

[3] Yiming Ding, Carlos Florensa, Mariano Phielipp, and Pieter Abbeel. Goal-conditioned imitation learning. *arXiv preprint arXiv:1906.05838*, 2019.

[4] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future, 2017.

[5] David D Fan, Ali-akbar Agha-mohammadi, and Evangelos A Theodorou. Learning risk-aware costmaps for traversability in challenging environments. *arXiv preprint arXiv:2107.11722*, 2021.

[6] David D Fan, Kyohei Otsu, Yuki Kubo, Anushri Dixit, Joel Burdick, and Ali-Akbar Agha-Mohammadi. Step: Stochastic traversability evaluation and planning for safe off-road navigation. *arXiv preprint arXiv:2103.02828*, 2021.

[7] Angelos Filos, Panagiotis Tigas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? In *International Conference on Machine Learning (ICML)*, 2020.

[8] A. Francis, A. Faust, H. T. L. Chiang, J. Hsu, J. C. Kew, M. Fiser, and T. W. E. Lee. Long-Range Indoor Navigation With PRM-RL. *IEEE Transactions on Robotics*, 2020.

[9] S. Ghosh, K. Otsu, and M. Ono. Probabilistic kinematic state estimation for motion planning of planetary rovers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5148–5154, 2018.

[10] Steven B Goldberg, Mark W Maimone, and Larry Matthies. Stereo vision and rover navigation software for planetary exploration. In *IEEE Aerospace Conference*. IEEE, 2002.

[11] Alain Haït, Thierry Simeon, and Michel Taïx. Algorithms for rough terrain trajectory planning. *Advanced Robotics*, 16(8):673–699, 2002.

[12] A. Howard, M. Turmon, L. Matthies, Benyang Tang, A. Angelova, and E. Mjolsness. Towards learned traversability for robot navigation: From underfoot to the far field. *J. Field Robotics*, 23: 1005–1017, 2006.

[13] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993.

[14] Ioannis Kostavelis, Lazaros Nalpantidis, and Antonios Gasteratos. Supervised traversability learning for robot navigation. In Roderich Groß, Lyuba Alboul, Chris Melhuish, Mark Witkowski, Tony J. Prescott, and Jacques Penders, editors, *Towards Autonomous Robotic Systems*, pages 289–298, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-23232-9.

[15] Jinhan Lee, Charles Pippin, and Tucker Balch. Cost based planning with rrt in outdoor environments. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 684–689. IEEE, 2008.

[16] Michael McCullough, Paramsothy Jayakumar, Jean Dasch, and David Gorsich. The next generation nato reference mobility model development. *Journal of Terramechanics*, 73:49–60, 2017.

[17] Kyohei Otsu, Guillaume Matheron, Sourish Ghosh, Olivier Toupet, and Masahiro Ono. Fast approximate clearance evaluation for rovers with articulated suspension systems. *Journal of Field Robotics*, 37:768–785, 2020.

[18] Panagiotis Papadakis. Terrain traversability analysis methods for unmanned ground vehicles: A survey. *Engineering Applications of Artificial Intelligence*, 26(4):1373–1385, 2013.

[19] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[20] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018.

[21] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. In *International Conference on Learning Representations*, 2020.

[22] Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real Single-Image Flight Without a Single Real Image. In *Robotics: Science and Systems (RSS)*, 2017.

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[24] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal Value Function Approximators. In *International Conference on Machine Learning (ICML)*, 2015.

[25] Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. ViNG: Learning Open-World Navigation with Visual Goals. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[26] Dhruv Shah, Benjamin Eysenbach, Nicholas Rhinehart, and Sergey Levine. Rapid Exploration for Open-World Navigation with Latent Goal Models. *Proceedings of the Conference on Robot Learning*, 2021.

[27] D. Singh Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta. Neural topological slam for visual navigation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[28] Rohan Thakker, Nikhilesh Alatur, Michael Paton, Kyohei Otsu, Olivier Toupet, and Ali-akbar Agha-mohammadi. Autonomous off-road navigation over extreme terrains with perceptually-challenging conditions. In *Experimental Robotics: The 17th International Symposium*, page 161. Springer Nature.