Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

iPlanner: Imperative Path Planning

Fan Yang[1], Chen Wang[2], Cesar Cadena[1], and Marco Hutter[1]

***Abstract*—The problem of path planning has been studied for years. Classic planning pipelines, including perception, mapping, and path searching, can result in latency and compounding errors between modules. While recent studies have demonstrated the effectiveness of end-to-end learning methods in achieving high planning efficiency, these methods often struggle to match the generalization abilities of classic approaches in handling different environments. Moreover, end-to-end training of policies often requires a large number of labeled data or training iterations to reach convergence. In this paper, we present a novel Imperative Learning (IL) approach. This approach leverages a differentiable cost map to provide implicit supervision during policy training, eliminating the need for demonstrations or labeled trajectories. Furthermore, the policy training adopts a Bi-Level Optimization (BLO) process, which combines network update and metric-based trajectory optimization, to generate a smooth and collision-free path toward the goal based on a single depth measurement. The proposed method allows task-level costs of predicted trajectories to be backpropagated through all components to update the network through direct gradient descent. In our experiments, the method demonstrates around 4$\times$ faster planning than the classic approach and robustness against localization noise. Additionally, the IL approach enables the planner to generalize to various unseen environments, resulting in an overall 26-87% improvement in SPL performance compared to baseline learning methods.**

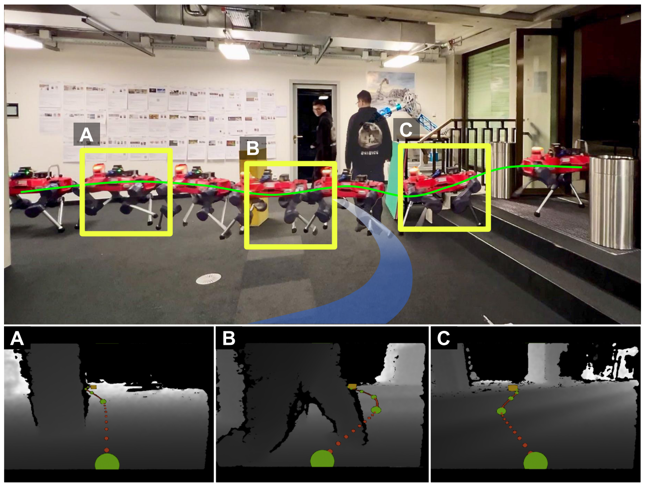


Fig. 1. Experiment of planning through static and dynamic obstacles with a legged robot, with A, B, and C representing three planning events. The goal is set outside the door from the start. The green curve shows the robot's trajectory as it (A) avoids a static obstacle, (B) avoids a moving human, and (C) climbs stairs to reach the goal. The blue curve represents human movement. The bottom images illustrate the depth measurements and the predicted trajectories from our method for these three events.

I. Introduction

Path planning is one of the significant tasks in the field of robotics. In structured environments, deploying a pre-built high-quality (HQ) map has pushed the limit and enabled robots to conduct daily tasks in environments shared with humans, e.g., autonomous driving [2]. However, for environments without a pre-built map, the robots can only rely on their onboard sensors for navigation. With limited sensor range, scene occlusions, and dynamic changes, the robots must react fast and re-plan efficiently to avoid collisions and navigate safely to their destinations.

The classic planning pipeline is based on a modular framework that includes perception, mapping, and path searching [5, 26]. This setup introduces latency as information is shared between modules and results in slower system response times. The planning performance is also limited by the information processed and preserved by each module, such as post-filtering and low-resolution perception. Maintaining a high-resolution map, on the other hand, can be computationally intensive and affect real-time performance. Additionally, if one module has errors or inaccuracies, these issues can be amplified and cause compounding effects by subsequent modules, potentially leading to system failure.

Researchers have been exploring end-to-end learning methods [23, 27] that can directly map sensory observations into trajectories or actions, as a way to improve the planning pipeline. Shah et al. [23] trains the planning policies to mimic reference trajectories using supervised learning. However, this method relies on having diverse data to function well in various scenarios, which is a common limitation of explicit supervised training. On the other hand, non-supervised approaches like reinforcement learning (RL) [27], which allows for random exploration, can lead to better generalizability. But, it struggles with low sampling efficiency and slow convergence, particularly in planning tasks with sparse task-level rewards. To address these issues, Pfeiffer et al. [21] combined supervised learning and RL, using prior demonstrations to improve training efficiency through better weight initialization using prior demonstrations. Recent studies [30, 32] have utilized auxiliary tasks to boost RL's training efficiency with external supervised signals. However, this approach still faces the risk of overfitting to the training data due to the use of explicit training labels. Additionally, the auxiliary losses, which may not align with the main navigation objective, can steer the policy toward suboptimal solutions.

To address this challenge, we introduce "imperative learn-

[1]Fan Yang, Cesar Cadena, and Marco Hutter are with Robotic Systems Lab, ETH Zurich, 8092 Zürich, Switzerland. Emails: {`fanyang1, cesarc, mahutter`}`@ethz.ch`
[2]Chen Wang is with Spatial AI and Robotics Lab, State University of New York at Buffalo, NY 14260, USA. Email: `chenwang@dr.com`

ing" (IL), a non-supervised approach aimed at improving the training efficiency and generalization of the policy. The policy includes network prediction and metric-based optimization, forming a BLO process during training. The IL approach trains the policy end-to-end without needing demonstrations or labeled references. Fig. 2 illustrates the overall process of the IL-based training for the planning policy. The core of IL lies in using differentiable metric-based optimization to direct network updates, resulting in an unsupervised "imperative" loss function. During training, a pre-built differentiable cost map provides traversability cost to guide the policy's behavior. When deployed, the policy decodes the traversability information directly from the input ego-centric observation and plans on top of it. Due to end-to-end training, the observation features extracted during deployment can be optimized specifically for the planning objective. Overall, the proposed IL approach can offer advantages over both end-to-end RL and supervised learning methods. In comparison to the end-to-end RL, instead of stochastically sampling the policy, the pre-computed cost map can guide the optimization process through direct gradient descent, improving training efficiency. In contrast to supervised learning, the IL approach utilizes task-level loss without any explicit labels or demonstrations, resulting in increased exploration of the action space, leading to better generalization. This paper presents the IL approach as a novel solution to overcome the limitations of previous methods for training a perceptive planning policy. Its main contributions are summarized as follows:

- A new non-supervised learning approach to train a perceptive planning policy with "imperative" supervision through direct gradient descent.
- An end-to-end training pipeline to map the single depth measurement to the trajectory by leveraging a BLO process with network update and metric-based trajectory optimization.
- Benchmarking the planning performance of learning-based methods with the classic approach in handling different types of unseen environments.

The iPlanner will be open-sourced[1] to promote research for learning navigation autonomy.

II. Related Work

Nowadays, classic planning frameworks, such as those done by Cao et al. [5], Wellhausen and Hutter [26], Yang et al. [29], have demonstrated effective and reliable performance in autonomous navigation in unseen terrains. For example, the navigation system, done by Cao et al. [5], incorporating mapping, terrain analysis, and motion-primitive path search[31], is deployed successfully during the DARPA Subterranean Challenge. The work of Wellhausen and Hutter [26] employs a pipeline with a learned terrain estimator and executes a lazy-PRM [12] path search, similar to the approach of Yang et al. [29] who uses a learned terrain cost and performs PRM* [11] with additional path optimization. Recently, there has been a shift towards end-to-end policies that directly map sensory inputs to planning paths or control actions [4, 8, 10, 14, 18, 20, 22, 23, 24, 27, 28, 30, 32]. This is due to the challenges posed by modularized systems, such as increased overall latency [18] and lack of robustness against noise or errors in real-world missions. However, the generalizability and reliability of these learning methods may still lag behind classic approaches, especially in scenarios that have not been encountered during training.

Supervised Learning: A common method for training an end-to-end policy is to copy expert or "ground truth" trajectories, such as works done by Bojarski et al. [4], Loquercio et al. [18], Pfeiffer et al. [20], Sadat et al. [22], Shah et al. [23], Xiao et al. [28]. The policy is trained to associate input sensory data with ground truth labels, which are obtained either by demonstrations from experts or statistical sampling. However, the explicitly supervised policy may not generalize well to diverse environments because of limited data diversity and richness. Shah et al. [23] uses 60 hours of navigation data from multiple robots and environments to improve generalization, but its performance is still limited by scene coverage. Moreover, relying on expert trajectories [22] or explicitly labeled references may limit the performance of the planner, leading it to converge towards sub-optimal solutions due to the limitations of the optimalities of the expert paths. Loquercio et al. [18] used simulated environments to train a planning policy for fast-flying drones. The simulated depth is estimated using stereo matching to reduce the gap between simulation and reality, allowing the network to be exposed to more training data with less effort in real-world data collection. Nevertheless, the use of a sampling-based planning algorithm to generate reference paths may limit the performance of the network by the optimality of the sampling-planning results.

Reinforcement Learning: Recently, deep reinforcement learning has gained popularity for training planning policies, as demonstrated in studies by Hoeller et al. [8], Kahn et al. [10], Kim et al. [14], Shi et al. [24], Wijmans et al. [27], Ye et al. [30], Zhu et al. [32]. As mentioned above, RL can utilize task-level loss. With simulated environments, using RL can save the effort of collecting and labeling data and generate various scenarios to improve the generalization of the policy. Moreover, the unsupervised nature of RL does not depend on the optimalities of expert labels and allows the network to explore the action space for better solutions. However, there are challenges in training with RL. One issue is the difficulty in creating a perfect simulator. To overcome the sim-to-real gap, researchers may need to collect data in the real world [10] or pre-process the simulated inputs based on heuristics [8]. Additionally, RL's low sample efficiency makes it challenging to train large policies end-to-end from dense inputs such as images or depth measurements. Training may take several days even with multiple GPUs [27]. To reduce the complexity of training, researchers have adopted strategies such as separating perception training from policy training [8] or using auxiliary tasks with heuristics-based loss functions [30, 32]. However, the success of these methods is contingent upon the accuracy

[1]iPlanner: https://github.com/leggedrobotics/iPlanner

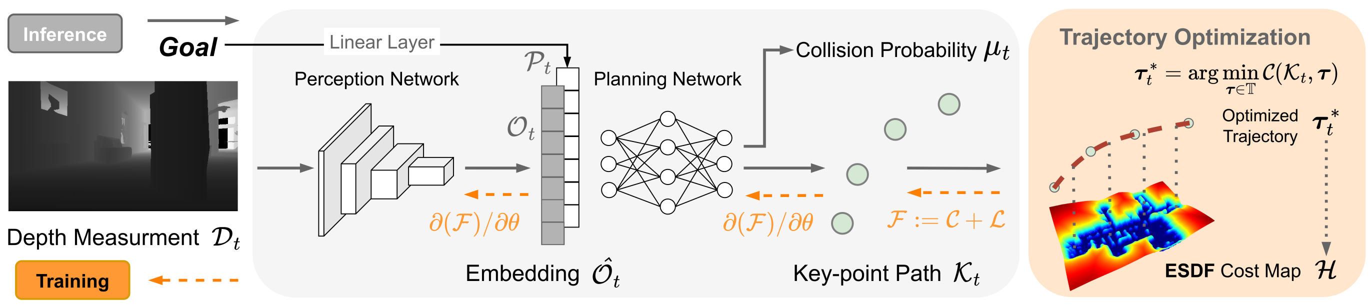

Inference
Goal
Linear Layer
Perception Network
$\mathcal{P}_t$
$\mathcal{O}_t$
Planning Network
Collision Probability μ_t
Trajectory Optimization
$\tau_t^* = \arg\min_{\tau\in\mathbb{T}} \mathcal{C}(\mathcal{K}_t, \tau)$
Optimized Trajectory τ_t^*
Depth Measurment $\mathcal{D}_t$
$\partial(\mathcal{F})/\partial\theta$
$\partial(\mathcal{F})/\partial\theta$
$\mathcal{F} := \mathcal{C} + \mathcal{L}$
Training
Embedding $\hat{\mathcal{O}}_t$
Key-point Path $\mathcal{K}_t$
ESDF Cost Map $\mathcal{H}$

Fig. 2. An overview of training the planning policy using IL. The pipeline consists of two parts, forming a BLO process with upper-level network update and lower-level trajectory optimization. During inference, the perception and planning network first encodes the depth measurement and goal position to predict a key-point path toward the goal with an associated collision probability. During training, the trajectory cost and task-level ”fear” loss are propagated back to provide direct gradients for updating both the perception and planning networks simultaneously.

and optimality of the heuristics employed.

This paper explores a novel non-supervised IL approach to train a planning policy from depth measurements end-to-end. This IL method uses task-level loss, eliminating the need for explicit labeling during training. By utilizing a differentiable cost map, the IL loss function can guide the network update through direct gradient descent to improve the training efficiency.

III. Methodology

Problem Definition: Define $\mathcal{Q} \subset \mathbb{R}^3$ as the workspace for the robot to navigate. Let the subset $\mathcal{Q}_{\text{obs}} \subset \mathcal{Q}$ represent the obstacles in the space that the robot cannot traverse through. Given a depth observation $\mathcal{D}_t \in \mathbb{R}^{H\times W}$ from the current time stamp t and a goal $\mathbf{p}_t^G \in \mathbb{R}^3$ in the robot frame, a trajectory $\boldsymbol{\tau}_t$ can be generated to guide the robot from the current position $\mathbf{p}_t^R \in \mathbb{R}^3$ towards to the goal $\mathbf{p}_t^G$ while avoiding obstacles $\mathcal{Q}_{\text{obs}}$. Here, we also denote a cost function $\mathcal{C}$ of a given trajectory $\boldsymbol{\tau}$, and an overall task-level loss function $\mathcal{F}$, such that $\mathcal{F} = \mathcal{C} + \mathcal{L}$, where $\mathcal{L}$ is an arbitrary non-negative function.

System Overview: The process of planning and imperative learning is illustrated in Figure 2. The pipeline consists of three modules. Firstly, a convolutional neural network (CNN) based [16] perception front-end transforms the depth input into an observation embedding. This embedding, combined with the goal waypoint feature, is then input into a planning network that predicts a key-point path, $\mathcal{K}_t$, towards the goal, which consists of n key points. Afterward, a metric-based trajectory optimizer further optimizes and interpolates the key-point path based on the cost $\mathcal{C}$ on the cost map $\mathcal{H}$ to generate the trajectory $\boldsymbol{\tau}_t$. The optimized cost $\mathcal{C}$, with other task loss $\mathcal{L}$, is then backpropagated through the network to update its parameters θ. Together, the network update and trajectory optimization form a nested BLO process. The aim of the pipeline is to enforce a trajectory optimization-based supervision (unsupervised optimization) on the perception and planning network f parameterized by θ. We hereby formulate the BLO model as follows:

$$\begin{aligned} &\min_{\theta} \mathcal{F}(f_\theta, \boldsymbol{\tau}^*) \\ &\text{s.t. } \boldsymbol{\tau}^* = \arg\min_{\boldsymbol{\tau}\in\mathbb{T}} C(f_\theta, \boldsymbol{\tau}), \end{aligned} \tag{1}$$

where $\boldsymbol{\tau}^*$ is the optimized trajectory under a constraint set $\mathbb{T}$, based on the objective $\mathcal{C}$. We next discuss the intuition of (1) in detail.

A. Perception and Planning Network

Perception Network: At each time stamp t, the robot receives a depth measurement $\mathcal{D}_t$ sampled from the space $\mathcal{Q}$. The front-head perception network encodes the observation $\mathcal{D}_t$ from the time t into a perception embedding $\mathcal{O}_t \in \mathbb{R}^{C\times M}$, preserving spatial and geometric information for the planning. Our method utilizes the backbone of the ResNet, proposed by He et al. [7], a classic and popular CNN architecture that is widely used as a feature extractor for image data. We adopt the efficient architecture: ResNet-18, to extract a feature representation for path planning from the perception input $\mathcal{D}_t$.

Planning Network: The planning network takes the perception embedding $\mathcal{O}_t \in \mathbb{R}^{C\times M}$, derived from the depth measurement $\mathcal{D}_t$, and the goal position $\mathbf{p}_t^G \in \mathbb{R}^3$ to find an collision-free to the goal. Here, C denotes the channel dimension of the embedding $\mathcal{O}_t$, and M represents the dimension of the feature space. The goal position, with a dimension of 3, is first mapped into a higher dimensional feature embedding $\mathcal{P}_t \in \mathbb{R}^{C^*\times M}$, where $C^* \geq 3$ using a linear layer. The perception embedding $\mathcal{O}_t$ and the expanded goal feature $\mathcal{P}_t$ are concatenated to form $\hat{\mathcal{O}}_t \in \mathbb{R}^{(C+C^*)\times M}$ as the observation input. The planning network uses a combination of CNN and MLP with ReLU activation to process $\hat{\mathcal{O}}_t$ and predict a key-point path $\mathcal{K}_t \in \mathbb{R}^{n\times 3}$ containing n key points. The path $\mathcal{K}_t$ is then interpolated and optimized by a metric-based trajectory optimizer.

B. Trajectory Optimization (TO)

To enforce the safety and smoothness of the trajectory generated by the planner, the key-point path $\mathcal{K}$ predicted by the planning network, is optimized by a trajectory optimizer. The objective is to minimize the trajectory cost $\mathcal{C}$ on a pre-defined cost map $\mathcal{H}$, which will be discussed in more detail later. The optimization takes the predicted key-point path $\mathcal{K}$ as the initial input and output the trajectory $\boldsymbol{\tau}^*$ under a constraint set $\mathbb{T}$, formulated as follow:

$$\boldsymbol{\tau}^* = \arg\min_{\boldsymbol{\tau}\in\mathbb{T}} \mathcal{C}(\mathcal{K}, \boldsymbol{\tau}), \text{ where } \mathcal{K} \leftarrow f_\theta \tag{2}$$

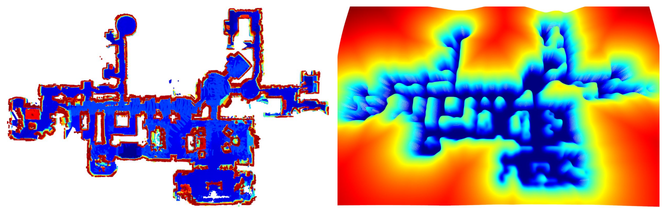

(a) Environment Point Cloud (b) ESDF Cost Map

Fig. 3. An illustration of a training environment and its ESDF cost map generated from the Matterport3D [6] dataset. (a) depicts the point cloud reconstructed from collected depth images within a Matterport3D room. (b) shows the smoothed ESDF cost map produced from the point cloud with Gaussian filtering.

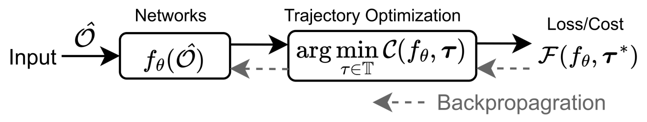

Input
$\hat{\mathcal{O}}$
Networks
$f_\theta(\hat{\mathcal{O}})$
Trajectory Optimization
$\arg\min_{\tau \in \mathbb{T}} \mathcal{C}(f_\theta, \tau)$
Loss/Cost
$\mathcal{F}(f_\theta, \tau^*)$
Backpropagration

Fig. 4. The mathematically BLO pipeline of the imperative training for the planning policy. The network function f_θ predicts a path as the input for the TO process. The TO minimizes the cost $\mathcal{C}$ with the optimized trajectory τ^*. The trajectory cost $\mathcal{C}$, combined with additional loss terms, forms the total training loss $\mathcal{F}$, and $\mathcal{F}$ is then backpropagated to update the network parameter θ.

Notice that the pre-built cost map is utilized only during training. During deployment, the TO interpolates and smooths the key-point path to generate a dynamically feasible trajectory $\boldsymbol{\tau}_t$ under a system constraint. Specifically, here, we use Cubic-Spline [19] to construct a third-order polynomial to pass through key points as a constraint. It interpolates m intermediate points in between every two key points in $\mathcal{K}_t$ and generates a dynamically feasible trajectory $\boldsymbol{\tau}_t \in \mathbb{R}^{(mn+1)\times 3}$ that passes through all the key points, with zero second derivatives on the start and endpoints, as per the natural boundary conditions. This optimization problem can be solved as a symmetric tridiagonal system with a linear solution [3]. Here, we create the cubic-spline function as a differentiable layer using PyPose [25] library to record the gradients of the output trajectory $\boldsymbol{\tau}_t$ to the input constraints $\mathcal{K}_t$ predicted by the network.

C. Optimization Objective and Training Loss

1) Trajectory Cost: The trajectory cost includes three differentiable terms that assess the quality of the output trajectory $\boldsymbol{\tau}_t$. The first term, obstacle cost $\mathcal{C}^{\mathcal{O}}$, checks if the trajectory collides or is too close to obstacles. To this end, before training, we reconstruct the environment offline based on the collected depth images, and their associated camera poses, as shown in Fig. 3(a). Then, based on the reconstructed environment, we build a Euclidean Signed Distance Field (ESDF) to label the distance to the nearest free (non-obstacle) space for each position in the environment. The ESDF is then smoothed with a Gaussian filter to make it locally differentiable and create a cost map $\mathcal{H}$ with non-negative cost values, shown in Fig. 3(b). To generate the cost $\mathcal{C}^{\mathcal{O}}$, each coordinate $\mathbf{p}_i$ on the trajectory $\boldsymbol{\tau}_t$ will be projected on this cost map $\mathcal{H}$ to get a cost value. The formulation of the obstacle cost $\mathcal{C}^{O}$ is shown as follows:

$$\mathcal{C}^{\mathcal{O}}(\boldsymbol{\tau}_t) = \sum_i \mathcal{H}(\mathbf{p}_i) \quad \mathbf{p}_i \in \boldsymbol{\tau}_t, |_{i=m\times n}, \tag{3}$$

where $\mathcal{H}(\mathbf{p}_i)$ represents the value of position $\mathbf{p}_i$ on the cost map $\mathcal{H}$. Secondly, we use the Euclidean distance from the final position of the trajectory $\boldsymbol{\tau}_t$ to the goal $\mathbf{p}_t^G$ as the second cost term $\mathcal{C}^{\mathcal{G}}$. It encourages the trajectory to be close to the destination and punishes the one that deviated away. The destination cost $\mathcal{C}^{\mathcal{G}}$ is hereby denoted as follow:

$$\mathcal{C}^{\mathcal{G}}(\boldsymbol{\tau}_t) = E(\mathbf{p}_i, \mathbf{p}_t^G) \quad \mathbf{p}_i \in \boldsymbol{\tau}_t \,|_{i=m\times n}, \tag{4}$$

where function E calculates the Euclidean distance between two given positions. Finally, the cost term $\mathcal{C}^{\mathcal{M}}$ is used to evaluate the motion smoothness of the trajectory. We assume that the same amount of time is taken to travel between two key points $\mathbf{p}_i^K, \mathbf{p}_{i+1}^K|_{i=0\ldots n-1}$ on the key-point path $\mathcal{K}_t$ and perform equal-time interpolation during the TO process. The objective of minimizing the cost $\mathcal{C}^{\mathcal{M}}$ is to reduce the difference in length of trajectory intervals on $\boldsymbol{\tau}_t$, thereby minimizing overall acceleration. The Euclidean distance from the current robot position $\mathbf{p}_t^R$ to the goal $\mathbf{p}_t^G$ is used to regulate the overall trajectory length and rewards a shorter motion. The motion cost $\mathcal{C}^{\mathcal{M}}$ is formulated as follow:

$$\mathcal{C}^{\mathcal{M}}(\boldsymbol{\tau}_t, \mathbf{p}_t^R, \mathbf{p}_t^G) = \sum_{i=0}^{n-2} \left| \frac{E(\mathbf{p}_t^R, \mathbf{p}_t^G)}{n-1} - E_{\boldsymbol{\tau}_t}(\mathbf{p}_i^K, \mathbf{p}_{i+1}^K) \right|, \tag{5}$$

where function $E_{\boldsymbol{\tau}_t}$ returns the length of a path interval between two given positions on the trajectory $\boldsymbol{\tau}_t$. The trajectory cost $\mathcal{C}$, as the objective of TO, is then formulated as a combination of $\mathcal{C}^{\mathcal{O}} \in \mathbb{R}_0^+$, $\mathcal{C}^{\mathcal{G}} \in \mathbb{R}^+$, and $\mathcal{C}^{\mathcal{M}} \in \mathbb{R}^+$:

$$\mathcal{C}(\boldsymbol{\tau}_t) = \alpha\, \mathcal{C}^{\mathcal{O}}(\boldsymbol{\tau}_t) + \beta\, \mathcal{C}^{\mathcal{G}}(\boldsymbol{\tau}_t) + \gamma\, \mathcal{C}^{\mathcal{M}}(\boldsymbol{\tau}_t, \mathbf{p}_t^R, \mathbf{p}_t^G), \tag{6}$$

where α, β, and $\gamma \in \mathbb{R}^+$ are hyperparameters used to balance the cost between different terms.

2) Fear Loss: In addition to the trajectory cost, the planning network predicts a collision probability μ_t for each trajectory to assess its risk of collision with obstacles. This is referred to as "fear loss". In some scenarios, the local planning policy can be trapped in a local minimum. Instead of setting large obstacle costs that could result in an over-conservative policy, having this task-level loss in addition to the trajectory cost provides the planner with the flexibility to escape from those scenarios and ensures safety. When deploying, the planner will only execute the trajectory with associated collision probability $\mu_t < 0.5$. The fear loss $\mathcal{L}(\boldsymbol{\tau}_t)$ is calculated using binary cross entropy (BCE):

$$\mathcal{L}(\boldsymbol{\tau}_t) = \begin{cases} \text{BCELoss}(\,\mu_t,\, 1.0\,) & \boldsymbol{\tau}_t \text{ collides w. } \mathcal{Q}_{\text{obs}} \\ \text{BCELoss}(\,\mu_t,\, 0.0\,) & \text{otherwise.} \end{cases} \tag{7}$$

The final training loss $\mathcal{F}$ is then formulated as the general summation of trajectory cost $\mathcal{C}$ and task-level fear loss $\mathcal{L}$:

$$\mathcal{F} = \mathcal{C}(\boldsymbol{\tau}_t) + \mathcal{L}(\boldsymbol{\tau}_t). \tag{8}$$

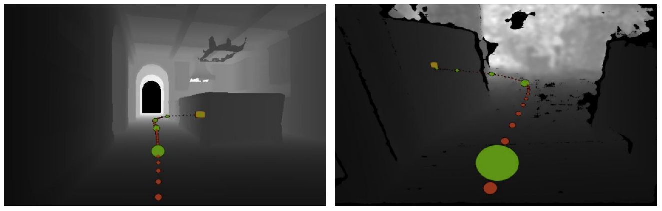


Fig. 5. Illustration of depth observation from simulation and the real world. (a) A depth image generated from the Gazebo simulation in the Matterport3D environment. (b) A depth observation obtained from Intel RealSense D435 during real-world experiments.

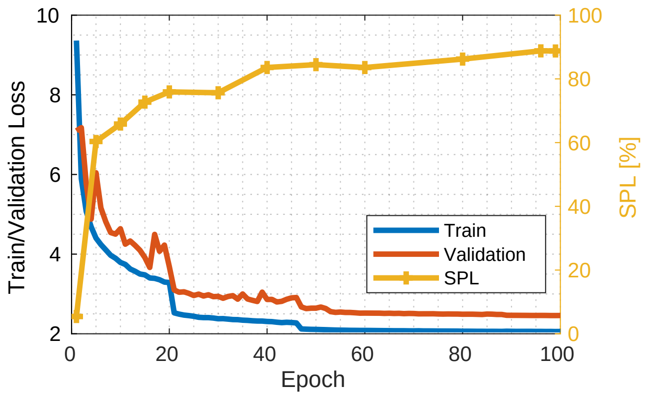


Fig. 6. Example of training and validation loss throughout the training process. The SPL (Success Weighted by Path Length) is evaluated in the garage simulation environment after specific epochs of training.

D. Training and Bi-Level Optimization (BLO)

As described above, the pipeline of the IL forms a nested BLO process, where the TO, as the lower-level task, takes the output of the network to generate the trajectory $\boldsymbol{\tau}$. Then, given the output $\boldsymbol{\tau}$, the upper-level process of BLO finds the network parameter θ to minimize the final training loss $\mathcal{F}$. Fig. 4 shows the pipeline in a mathematical formulation. To solve the BLO problem, we adopt the concept of Proxy-Based Explicit Gradient for Best-Response (EGBR) [17] by using approximated BR mapping. Specifically, instead of explicitly solving the optimization of the lower-level task TO, which can be computationally expensive, we approximate a sub-optimal solution $\boldsymbol{\tau}^{\text{sub}}$ by directly using the output of the cubic-spline interpolation and its projection cost on map $\mathcal{H}$ as the trajectory cost $\mathcal{C}^{\text{sub}}$:

$$\boldsymbol{\tau}^{\text{sub}} \sim \arg\min_{\boldsymbol{\tau}\in\mathbb{T}} \mathcal{C}(f_\theta, \boldsymbol{\tau}) \tag{9}$$

where $\boldsymbol{\tau}^{\text{sub}}$ is an approximation to the optimal solution $\boldsymbol{\tau}^*$. With the trajectory $\boldsymbol{\tau}^{\text{sub}}$, the corresponding cost $\mathcal{C}^{\text{sub}}$ of the TO objective, together with the task-level loss $\mathcal{L}^{\text{sub}}$, are propagated back to optimize the network parameter θ using gradient decent and solves the BLO problem iteratively:

$$\theta_{i+1} \leftarrow \theta_i - \omega \nabla_{\theta_i} \mathcal{F}(f_{\theta_i}, \boldsymbol{\tau}_i^{\text{sub}}), \tag{10}$$

where ω is a hyperparameter of the learning rate. The gradient of the training loss $\mathcal{F}$ to the network parameter θ is denoted

TABLE I
PLANNING PERFORMANCE ACROSS FOUR TYPES OF ENVIRONMENTS

SPL % - Success weighted by Path Length					
	Forest	Garage	Indoor	Matterport	Overall
MP [31] (LiDAR)	95.09	**89.42**	85.82	74.18	86.13
SL [18]	65.58	46.70	50.03	28.87	47.80
RL [8] (Tilt)	95.08	69.43	61.10	59.24	71.21
Ours (Tilt)	95.66	73.49	68.87	76.67	78.67
Ours	**96.37**	88.85	**90.36**	**82.51**	**89.52**

TABLE II
PLANNING LATENCY IN [MS] FOR DIFFERENT METHODS

SL [18]	RL [8]	MP [31]	Ours	Ours (*Jetson*)
13.5 ($\pm$1.8)	**3.6 ($\pm$0.3)**	24.9 ($\pm$4.2)	5.9 ($\pm$0.4)	11.4 ($\pm$0.6)

as follows:

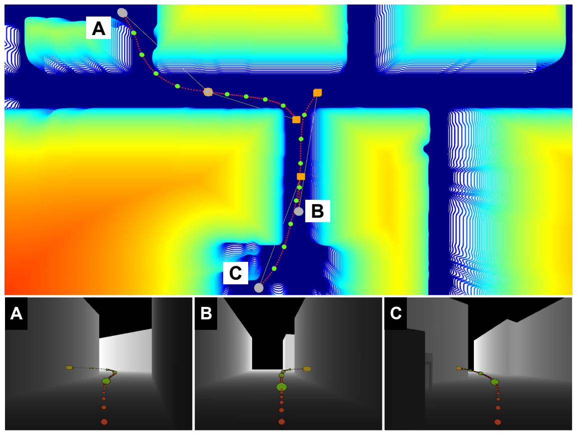
$$\nabla_\theta \mathcal{F} = \left(\frac{\partial \mathcal{F}}{\partial f} + \frac{\partial \mathcal{C}}{\partial f}\right)\frac{\partial f}{\partial \theta} + \frac{\partial \mathcal{C}}{\partial \tau}\frac{\partial \tau}{\partial \theta}. \tag{11}$$

IV. EXPERIMENTS

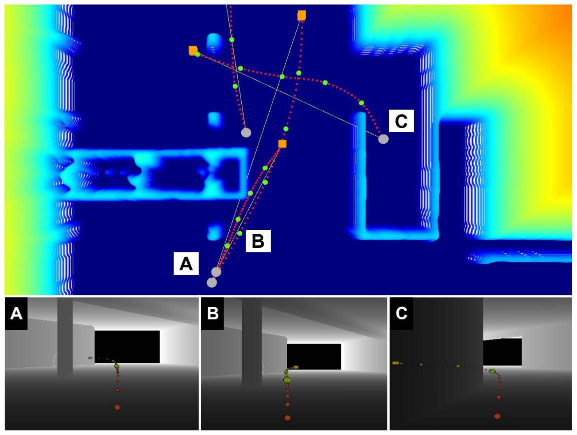
We assess our method through both simulated and real-world evaluations, comparing it to classic and learning-based baselines. The simulations are run on a 2.6GHz i9 laptop with an NVIDIA RTX 3080 GPU. Our real-world tests are carried out using the ANYmal legged robot [9] equipped with an NVIDIA Jetson Orin for the execution of our planning method.

The classic motion primitives (**MP**) planner, as proposed by Zhang et al. [31], integrates with a modularized pipeline [5] and uses a 360° view LiDAR. It is considered the state-of-the-art (SOTA) non-learning method in terms of success rate and efficiency, serving as a performance reference for learning methods. As learning-based approaches, the supervised learning (**SL**) method by Loquercio et al. [18] and the reinforcement learning (**RL**) method by Hoeller et al. [8] serve as baselines. Both our method and the learning baselines use a front-facing stereo-depth camera with a frame rate of 15Hz. The RL method requires a downward-tilted camera view of 30°, while the SL method requires a forward-looking camera. Our method is tested with both camera settings using the same policy weights.

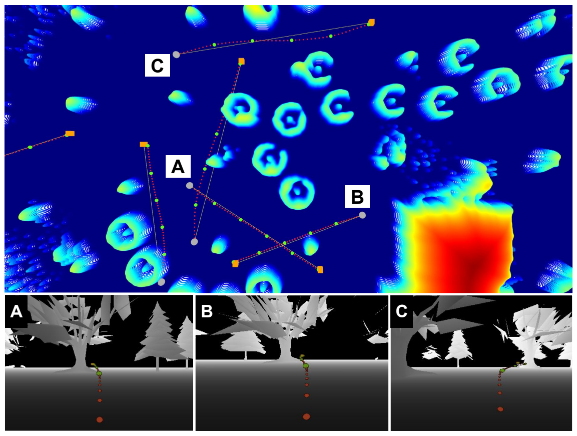
Our method is trained using a combination of data collected in both simulated and real-world environments. We gather approximately 20k depth images from various camera positions in the Matterport3D [6] environment as well as in simulated campus and tunnel [5] environments using Gazebo [15]. Additionally, we collect 10k images in real-world environments to adjust the policy for real-world perception noise. An example of depth observation in a simulated environment versus a real-world environment can be seen in Fig. 5. The training takes about 20 hours (100 epochs) using 30k images and a single NVIDIA 3090 Ti GPU, without pre-trained ResNet weights. After just 20 epochs (4 hours), the policy can demonstrate a high success rate in navigating through the unseen simulated environment, as shown in Fig. 6.



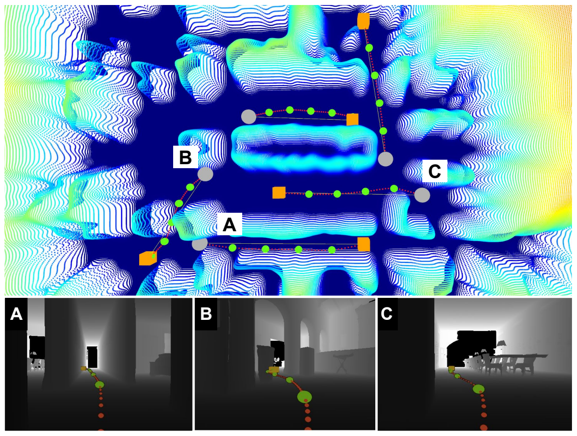
(a) Indoor



(b) Garage



(c) Forest



(d) Matterport

Fig. 7. Planning examples in simulation experiments: (a) Indoor, (b) Garage, (c) Forest, (d) Matterport3D. The bottom images show the depth measurements for the corresponding marked trajectories on the cost maps. The key points (green), along with the trajectories (red) shown in the depth images, or on cost maps, are generated by our method from the current robot positions (gray dot) to the goals (yellow/orange box), with the current depth images. The trajectories shown in the depth measurements are projected onto the image frame for visualization purposes.

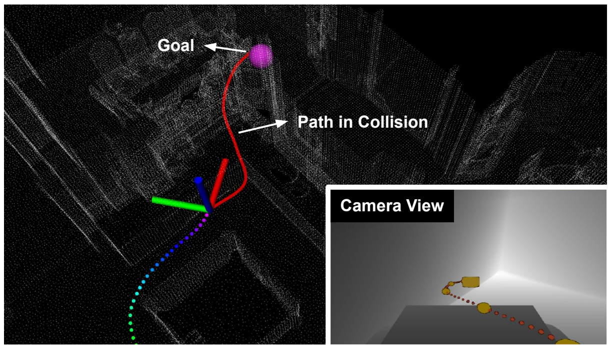
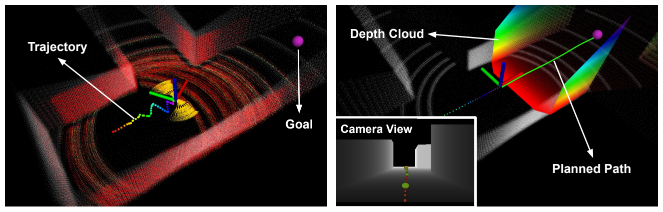


Fig. 8. An illustration of a local minimum scenario. The point cloud, in white, depicts the surrounding environment. The robot is trapped in a corner where no collision-free path to the goal (purple) exists within the FOV. The network predicts a path (red) with a high collision probability, which triggers the protective behavior to stop the robot.



(a) (b)

Fig. 9. Experiment against localization errors. (a) illustrates the compounding effect of the classic MP method with the modularized pipeline. The point cloud in red depicts the obstacles detected by the terrain analysis module. (b) shows the planning result of our method. In the same conditions, a feasible path (green) to the goal is planned based on the current depth observation displayed in the camera view image.

A. Simulation Experiments

The simulated experiments are conducted in four different types of environments: indoor, garage, forest, and a Matterport3D room, as shown in Fig. 7, using the Autonomous Exploration Development Environment [5]. The testing environments are not seen by the learning-based planning policies during training. In each environment, 30 pairs of start and goal positions are selected in traversable areas and given to the

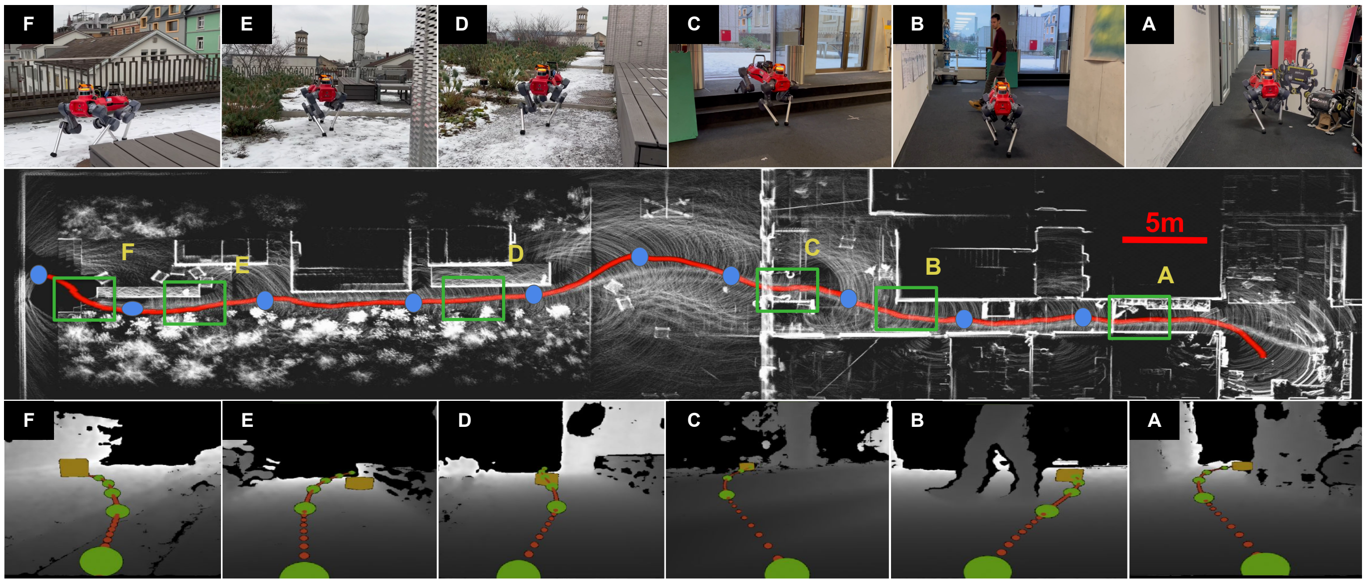


Fig. 10. Real-world experiment with the legged robot, with the red curve indicating the robot's trajectory starting from right to left. The robot begins inside a building and navigates to the outside. Green boxes A-F mark different planning events during the route. The robot follows a series of waypoints (shown in blue) and plans in between to: (A) pass through doorways, (B, D, E) circumvent static/dynamic obstacles, and (B, F) ascend and descend stairs.

robot. The robot uses onboard sensing to search for collision-free paths and navigate to reach the goal.

Table. I presents the results of experiments. The performance of the planning methods is evaluated using SPL [1] (Success weighted by Path Length). In the experiments, the SL method is found to be inferior in generalizing to the four unseen testing environments. On the other hand, the RL method performs well in the forest but struggles in the other three environments. This is due to its overfitted perception header trained with a downward-tilted camera, resulting in limited field-of-view (FOV) for planning effectively in complex environments. Furthermore, the perception header of the RL policy is trained using a self-supervised approach, not specifically optimized for planning. In contrast, our method demonstrates consistent and high performance across all different types of unseen environments and outperforms even the classic method with a 360° degree FOV LiDAR. Furthermore, our policy is capable of directly generalizing to different camera settings. Even with the same downward-tilted camera as the RL method, our policy can still achieve the best performance among all learning-based methods, shown in Table I. On the other hand, the robot's performance can be impacted by local minimal scenarios when its FOV is limited. For those cases, our network predicts "fear" values to detect trajectory collisions and stop the robot before crashing, as demonstrated in Fig. 8. In summary, our proposed planner has the capability to achieve, on average, 87% better performance than the SL method and 26% better performance than the RL method in terms of SPL to reach destinations.

The MP approach models the environment using a metric-based method and offers superior performance and generalization compared to learning-based baselines [8, 18] as shown in Table I. However, its modularized pipeline can result in high latency and sensitivity to noise, e.g., localization errors. To assess this, we add random localization noise with a standard deviation of 5cm to the system. This causes false-positive obstacle detections on the ground due to compounding effects between the mapping and terrain analysis modules, as shown in Fig 9. With these false detections, the MP planner cannot find a "collision-free" path to the goal. Our proposed method demonstrates high robustness, even in the presence of random localization noise, through its end-to-end pipeline that takes instantaneous sensor input for planning. Despite not using a panoramic sensor like LiDAR, our method offers similar generalization and even better performance than the classic MP method. Table II shows our method's computational efficiency and small planning latency, which is more than 4 times faster than the MP method but slightly slower than the RL approach.

B. *Real-World Experiment with Legged Robot*

Our experiment evaluates the effectiveness of our method in real-world scenarios using the legged robot ANYmal [9]. The robot is equipped with an Intel Realsense D435 front depth camera for depth measurement. The experimental setup involves both dynamic and static obstacles, with the robot navigating from indoor to outdoor environments following sequential waypoint commands from a human operator, as shown in Fig. 10. The planner navigates the robot through doorways, around both dynamic and static obstacles, and up and down stairs. The indoor and outdoor environments have different lighting conditions, which results in varying levels of noise in the depth measurements. Here, our planner relies on a localization method [13] to translate the goal into the robot frame for the planning network.

We also conducted experiments in various environmental settings to assess the generalization and efficiency of our

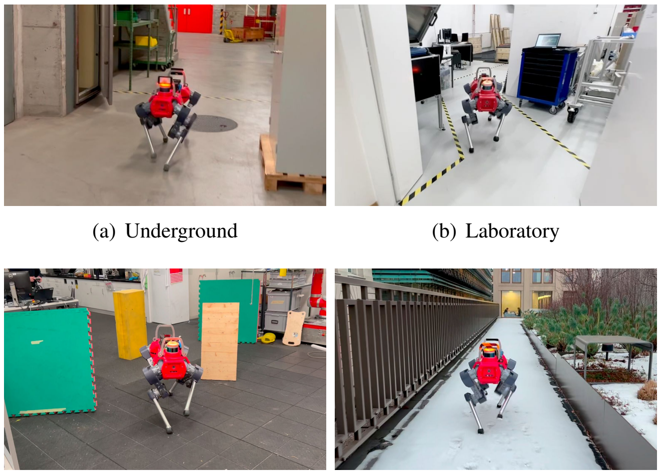

(a) Underground (b) Laboratory (c) Human-made "Maze" (d) Terrace

Fig. 11. Planning method testing in different unseen environments, indoor and outdoor, with various obstacle setups and lighting conditions.

planning policy, as illustrated in Fig. 11. In human-made mazes, the goal may be hidden from the start, requiring the robot to navigate around obstacles. Outdoors, the robot must traverse various terrains, such as snow, grass, and plants, to reach its destinations. In the underground environment, the robot encounters different lighting conditions and unfamiliar obstacle shapes. Despite these challenges, our method guides the robot successfully to its destinations while maintaining a low average latency of 11.4ms on the Nvidia Jetson Orin onboard computer. Furthermore, our method, which relies solely on a single depth frame, can operate independently of the localization module if the goal is set in the robot frame. We use a joystick command to set the goal in the robot frame as directional guidance, and the method then calculates safe paths in the local frame for the robot to follow.

V. Conclusion

In this paper, we present an end-to-end planning framework based on a novel imperative learning (IL) approach. The method involves a bi-level optimization (BLO) process that combines network update and metric-based trajectory optimization during training to produce smooth and collision-free trajectories using only a single depth measurement. The IL is able to utilize task-level loss and optimize through direct gradient descent. This allows the method to be trained in an efficient unsupervised manner, eliminating the need for explicit trajectory labels. In the experiment, we benchmark the performance of our planner with the SOTA classic non-learning method [31] and learning baselines [8, 18]. Our experiments demonstrate that the resulting policy has the capability to achieve efficient planning and generalize to various unseen environments compared to previous approaches. Further, we evaluate our method in various real-world settings, including dynamic environments and different camera settings. The results indicate the effectiveness of our method in real-world deployment, generalizing to novel environments and camera configurations, and being robust against perception and localization errors.

VI. Discussion

This paper presents a novel framework for training a perceptive planning policy that relies solely on depth measurement as input. While RGB images can offer supplementary information beneficial for planning, images collected from simulation may result in a significant difference between real environments. Thus, further research is required. Additionally, we realize that adding memory structure to the perception network could have improved the performance of the planning in the static environment. However, utilizing the current pipeline, the memory structure may compromise the safety of the planning to avoid dynamic obstacles. To address this issue, it may be necessary to incorporate a time-variable cost map in formulating the training loss. Overall, this work aims to demonstrate the innovative concept of using the imperative learning approach, and the result can be considered preliminary.

VII. Acknowledgement

This research was supported by: the Swiss National Science Foundation (SNSF) under project 188596, the National Centre of Competence in Research Robotics (NCCR Robotics), and the European Union's Horizon 2020 research and innovation program via grant agreements No. 101016970, No. 101070405, and No. 852044. Additionally, this work also benefited from the ETH Zurich Research Grant.

References

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[2] Zhibin Bao, Sabir Hossain, Haoxiang Lang, and Xianke Lin. High-definition map generation technologies for autonomous driving: a review. *arXiv preprint arXiv:2206.05400*, 2022.

[3] Richard H Bartels, John C Beatty, and Brian A Barsky. *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, 1995.

[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.

[5] Chao Cao, Hongbiao Zhu, Fan Yang, Yukun Xia, Howie Choset, Jean Oh, and Ji Zhang. Autonomous exploration development environment and the planning algorithms. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8921–8928. IEEE, 2022.

[6] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] David Hoeller, Lorenz Wellhausen, Farbod Farshidian, and Marco Hutter. Learning a state representation and navigation

in cluttered and dynamic environments. *IEEE Robotics and Automation Letters*, 6(3):5081–5088, 2021.

[9] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 38–44. IEEE, 2016.

[10] Gregory Kahn, Pieter Abbeel, and Sergey Levine. Badgr: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters*, 6(2):1312–1319, 2021.

[11] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[12] L Kavraki and R Bohlin. Path planning using lazy prm. In *International Conference on Robotics and Automation*, volume 1, pages 521–528, 2000.

[13] Shehryar Khattak, Huan Nguyen, Frank Mascarich, Tung Dang, and Kostas Alexis. Complementary multi–modal sensor fusion for resilient robot pose estimation in subterranean environments. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1024–1029. IEEE, 2020.

[14] Yunho Kim, Chanyoung Kim, and Jemin Hwangbo. Learning forward dynamics model and informed trajectory sampler for safe quadruped navigation. *arXiv preprint arXiv:2204.08647*, 2022.

[15] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[16] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[17] Risheng Liu, Jiaxin Gao, Jin Zhang, Deyu Meng, and Zhouchen Lin. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (12):10045–10067, 2021.

[18] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021.

[19] Sky McKinley and Megan Levine. Cubic spline interpolation. *College of the Redwoods*, 45(1):1049–1060, 1998.

[20] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, page 1527–1533. IEEE, 2017.

[21] Mark Pfeiffer, Samarth Shukla, Matteo Turchetta, Cesar Cadena, Andreas Krause, Roland Siegwart, and Juan Nieto. Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations. *IEEE Robotics and Automation Letters*, 3(4):4423–4430, 2018.

[22] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *European Conference on Computer Vision*, pages 414–430. Springer, 2020.

[23] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. Gnm: A general navigation model to drive any robot. *arXiv preprint arXiv:2210.03370*, 2022.

[24] Haobin Shi, Lin Shi, Meng Xu, and Kao-Shing Hwang. End-to-end navigation strategy with deep reinforcement learning for mobile robots. *IEEE Transactions on Industrial Informatics*, 16 (4):2393–2402, 2019.

[25] Chen Wang, Dasong Gao, Kuan Xu, Junyi Geng, Yaoyu Hu, Yuheng Qiu, Bowen Li, Fan Yang, Brady Moon, Abhinav Pandey, Aryan, Jiahe Xu, Tianhao Wu, Haonan He, Daning Huang, Zhongqiang Ren, Shibo Zhao, Taimeng Fu, Pranay Reddy, Xiao Lin, Wenshan Wang, Jingnan Shi, Rajat Talak, Kun Cao, Yi Du, Han Wang, Huai Yu, Shanzhao Wang, Siyu Chen, Ananth Kashyap, Rohan Bandaru, Karthik Dantu, Jiajun Wu, Lihua Xie, Luca Carlone, Marco Hutter, and Sebastian Scherer. PyPose: A library for robot learning with physics-based optimization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[26] Lorenz Wellhausen and Marco Hutter. Rough terrain navigation for legged robots using reachability planning and template learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6914–6921. IEEE, 2021.

[27] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

[28] Xuesu Xiao, Tingnan Zhang, Krzysztof Choromanski, Edward Lee, Anthony Francis, Jake Varley, Stephen Tu, Sumeet Singh, Peng Xu, Fei Xia, et al. Learning model predictive controllers with real-time attention for real-world navigation. *arXiv preprint arXiv:2209.10780*, 2022.

[29] Bowen Yang, Lorenz Wellhausen, Takahiro Miki, Ming Liu, and Marco Hutter. Real-time optimal navigation planning using learned motion costs. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9283–9289. IEEE, 2021.

[30] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectgoal navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16117–16126, 2021.

[31] Ji Zhang, Chen Hu, Rushat Gupta Chadha, and Sanjiv Singh. Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation. *Journal of Field Robotics*, 37(8): 1300–1313, 2020.

[32] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10012–10022, 2020.