PROBE: Proprioceptive Obstacle Detection and Estimation while Navigating in Clutter

Dhruv Metha Ramesh[1], Aravind Sivaramakrishnan[2], Shreesh Keskar[1], Kostas E. Bekris[3], Jingjin Yu[1], Abdeslam Boularias[1]

***Abstract*— In critical applications, including search-and-rescue in degraded environments, blockages can be prevalent and prevent the effective deployment of certain sensing modalities, particularly vision, due to occlusion and the constrained range of view of onboard camera sensors. To enable robots to tackle these challenges, we propose a new approach, Proprioceptive Obstacle Detection and Estimation while navigating in clutter (PROBE), which instead relies only on the robot's proprioception to infer the presence or absence of occluded rectangular obstacles while predicting their dimensions and poses in SE(2). The proposed approach is a Transformer neural network that receives as input a history of applied torques and sensed whole-body movements of the robot and returns a parameterized representation of the obstacles in the environment. The effectiveness of PROBE is evaluated on simulated environments in Isaac Gym and with a real Unitree Go1 quadruped robot. The project webpage can be found at https://dhruvmetha.github.io/legged-probe/.**

I. Introduction

In a dark environment, humans can effectively navigate by relying on mechanical interactions with the fixed and movable obstacles present. While navigating in such an environment, humans construct in real-time a mental representation of these objects, including their dimensions and poses, entirely through *contact force-based interactive sensing*. Similarly, modeling the movability of objects on the fly is an important skill for autonomous robots deployed in open environments, such as those encountered in search-and-rescue (SAR) operations, unmanned exploration missions, and debris removal [1]–[7]. These environments can contain objects with diverse physical properties in terms of shape and materials. Therefore, some level of on-demand object property identification can help guide navigation.

Inferring the shapes of objects in cluttered scenes is extremely challenging because of occlusions, which only permit a partial view of the surrounding environment. For instance, in a degraded environment, objects can be hidden behind rubble and thus cannot be observed directly. Visual input may also be extremely limited or impossible in certain critical situations, such as navigating through dense smoke.

This work addresses the problem of partial scene reconstruction for a legged robot using only proprioception data. We consider a setup (Fig 1) where a legged robot is deployed in an unknown environment and tasked to navigate to a target location without access to visual inputs. A learned high-level navigation policy generates desired velocities for the robot, unaware of any obstacles present in the environment, and a learned low-level locomotion controller translates them into joint positions. A Transformer-based Obstacle Reconstruction Module (ORM) receives as inputs the history of the robot's SE(2) pose, the desired velocity commands, its joint positions, velocities, and torques at each time-step. It returns a reconstructed environment representation at each time-step, which is gradually fine-tuned as the robot navigates in its environment. Using only this proprioceptive feedback, the ORM can predict the existence of planar rectangular obstacles, their 2D dimensions and positions. In addition, the ORM can detect whether the encountered obstacles are fixed or movable, and in the latter case, further detect other obstacles hidden behind, and what their properties are. The capability to sense objects hidden behind movable large objects cannot be attained through vision. This information can enable the robot to use a high-level reasoning framework to navigate among movable objects [8] by clearing the path unlocking regions to explore.

In summary, this work brings forth the following main contributions:

- The construction of a neural architecture, PROBE, capable of rapidly predicting an actionable representation of the robot's environment using only the history of a robot's proprioceptive states during navigation in that environment, without any vision sensors, and
- A comprehensive empirical study of the proposed system

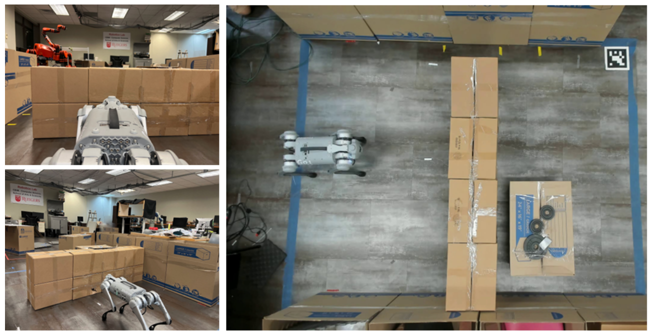
Fig. 1: The setup considered in PROBE involves a Go1 robot dog and obstacles that can potentially obstruct its path. Some planar obstacles, such as the long frontal box in the image, are movable, while others are fixed to the ground. A transformer network reconstructs the locations and sizes of the obstacles, including the occluded ones, from a history of proprioceptive data that the robot receives while exploring the scene without vision.

[1]Dept. of Computer Science, Rutgers University, NJ, USA. [2]A.S. is affiliated with Amazon.com Inc. but the work related to this paper was performed while at Rutgers University. [3]K. E. Bekris holds concurrent appointments as a Professor at Rutgers University and as an Amazon Scholar. This paper describes work performed at Rutgers and is not associated with Amazon. This work is partly supported by NSF awards 1846043 and 2132972. Corresponding author e-mail: dhruv.metha@rutgers.edu.

in simulation and on a real quadruped, demonstrating the effectiveness of PROBE.

II. Related Work

Object/Scene Reconstruction has been extensively studied using vision-based methods that rely on RGB-D images for volumetric shape completion of partially occluded objects [9]–[11]. To infer 3D shapes in clutter without vision, prior physics-based methods like [10] perform computationally expensive searches over large shape spaces based on their observed interactions. The proposed ORM can rapidly predict planar obstacle properties directly from proprioceptive data, circumventing intensive computations. Other related work focusing on scene reconstruction from force sensing [12] do not deal with *nested* object interactions considered in this paper's problem setup.

Contact Sensing is crucial for scene reconstruction from force measurements. Early attempts solve the force-moment balance and surface equations while assuming a soft finger and convex shape of the contacted object [13]. These assumptions were relaxed in subsequent works [14], [15]. Some works addressed the problem of locating contact points on manipulators [16], which is closer to the presented work on legged robots.

Mapping and Navigation involves building a map of a mobile robot's unknown environment while the robot is navigating an environment. The simultaneous localization and mapping (SLAM) problem is extensively studied in mobile robotics, with most solutions requiring visual inputs in the form of RGB or LIDAR images [17]–[19]. In this paper, the robot navigates under uncertainty, only observing the environment through contact, a variant of the *Blindfolded Traveler's Problem* [20]. Proposed solutions assume rigid and fixed obstacles [21]–[23], which is different from our setup, which contains both fixed and movable obstacles.

Legged Robot Navigation utilizes proprioceptive feedback to adapt locomotion control on challenging terrains in real-time [24]–[26]. Unlike methods that implicitly encode environmental conditions, the proposed approach leverages proprioception to reconstruct the environment by inferring the presence and properties of obstacles. This shifts the use of proprioception from internal adaptation to external environment mapping, addressing a different aspect of autonomous navigation in complex environments.

III. Problem and Environment Setup

We address the problem of Obstacle Prediction while navigating in clutter. A legged robot negotiates a 2D workspace to reach a goal region (Fig. 2) without visual inputs. The environment contains varying numbers of box-shaped obstacles that are either mobile (e.g., movable by pushing) or immobile. The locations and sizes of the obstacles in the environment are unknown *a priori*. While navigating, the robot manipulates or comes into contact with a subset of the obstacles in its workspace. Given the robot's proprioceptive state history, the environment is partially reconstructed by predicting the locations, sizes, and mobility of the obstacles it has encountered, either directly or indirectly, through nested interactions. Nested interactions refer to cases when the robot pushes a moving obstacle, which in turn pushes a second obstacle that is behind it.

Robotic Platform. This work uses the Unitree Go1-EDU legged robot. Locomotion is performed by controlling its twelve joints. The proprioception signal at time-step t is $s_t = \{(q_i, \dot{q}_i, \tau_i)\}_{i=1}^{12}$, where $(q_i, \dot{q}_i)$ are the position and velocity of joint i, and τ_i is the torque applied on joint i. The pose and velocity of the robot at time t is $\rho_t = (x_t, y_t, \theta_t)$ and $\dot{\rho}_t = (\dot{x}_t, \dot{y}_t, \dot{\theta}_t)$, where $(x_t, y_t, \theta_t) \in \mathtt{SE}(2)$ is the 2D translational and rotational transformations w.r.t. the robot's initial position A, and $(\dot{x}_t, \dot{y}_t, \dot{\theta}_t)$ are the linear and angular velocity components.

Both movable and static obstacles are placed randomly in the workspace and have unknown friction, mass, and size. PROBE's scene estimation module represents each obstacle as $\mathcal{O}_i = (\mathbb{I}_i^{\text{static}}, x_i, y_i, \theta_i, w_i, l_i)$. The binary variable $\mathbb{I}_i^{\text{static}}$ indicates whether the obstacle is static (1) or movable (0). $(x_i, y_i, \theta_i) \in \mathtt{SE}(2)$ defines the pose of the obstacle relative to the robot's initial position A (typically the origin), and (w_i, l_i) defines the box's width and length respectively. The set of all obstacles in the environment with N obstacles is $\mathcal{O} = \{\mathcal{O}_i\}_{i=1}^{N}$.

IV. Proposed Method

A. Low-level Locomotion Controller

The locomotion controller π_{loc} [27] is a neural network trained in an obstacle-free environment using Proximal Policy Optimization (PPO) [28]. π_{loc} tracks velocity and gait commands, enabling the robot to walk at desired speeds while maintaining a commanded gait. The velocity commands v^{des} are specified within the robot's $\mathtt{SE}(2)$ frame. Gait commands b include the robot's stepping frequency, body height, and stance. The input observations o^{loc} to the policy are the gravity vector g in the robot's frame, the robot's proprioceptive states – joint positions q, joint velocities $\dot{q}$, the previous action $a_{\text{prev}}^{\text{loc}}$, and a latent physics parameter z^{phys}. The commands $c = (v^{\text{des}}, b)$ are also provided as input. The output action a^{loc} is the target joint position for each of the robot's joints. These target joint positions are converted to joint torques τ using the robot's built-in PD controller. π_{loc} uses a student-teacher framework [24], to adapt to different environment conditions as encoded in z^{phys}. In summary, the neural-network policy π_{loc} returns the following action at each time t: $a_t^{\text{loc}} = \pi_{\text{loc}}(g_t, q_t, \dot{q}_t, z_t^{\text{phys}}, a_{t-1}^{\text{loc}}; c_t)$.

B. High-level Navigation Policy

A navigation policy π_{nav} is trained using PPO in cluttered corridors to provide a sequence of velocity commands $v^{\text{des}} = (v_x, v_y, v_\omega) \in \mathbb{R}^3$ to the low-level controller π_{loc} so that the robot successfully navigates from its initial position to the goal region. The commanded gait b is fixed to a trotting gait due to its simplicity and stability. π_{nav} takes as input a history H of observations o^{nav} and outputs a desired velocity command v^{des}. The observation contains the $\mathtt{SE}(2)$ pose of the robot ρ, the robot's velocity vector $(\dot{x}, \dot{y})$, and

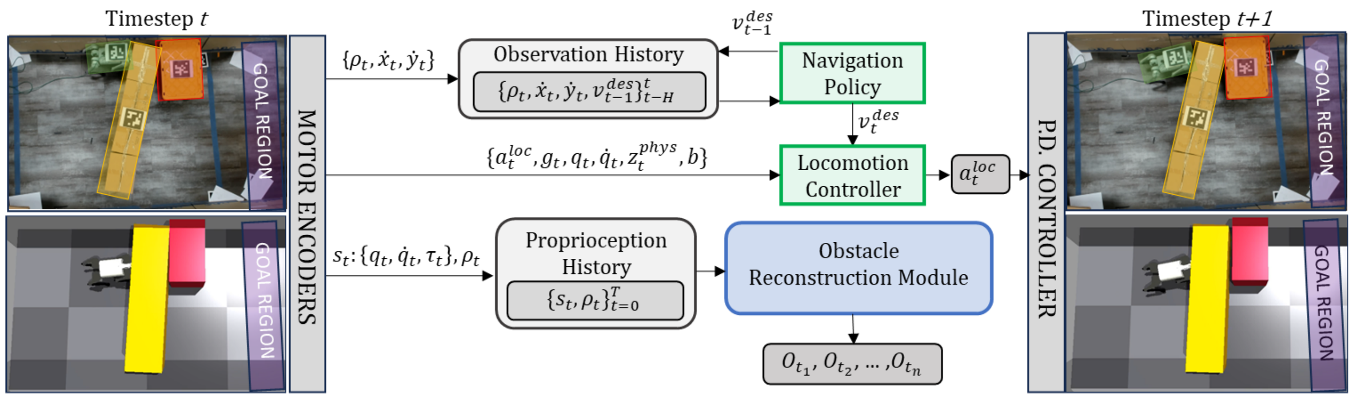
Timestep t
GOAL REGION
GOAL REGION
MOTOR ENCODERS
$\{\rho_t, \dot{x}_t, \dot{y}_t\}$
Observation History
$\{\rho_t, \dot{x}_t, \dot{y}_t, v_{t-1}^{des}\}_{t-H}^{t}$
v_{t-1}^{des}
Navigation Policy
v_t^{des}
$\{a_t^{loc}, g_t, q_t, \dot{q}_t, z_t^{phys}, b\}$
Locomotion Controller
a_t^{loc}
s_t: $\{q_t, \dot{q}_t, \tau_t\}, \rho_t$
Proprioception History
$\{s_t, \rho_t\}_{t=0}^{T}$
Obstacle Reconstruction Module
$O_{t_1}, O_{t_2}, \dots, O_{t_n}$
PD. CONTROLLER
Timestep $t+1$
GOAL REGION
GOAL REGION

Fig. 2: (Left) Environment setup in reality (top) and simulation (bottom), best viewed in color. The robot's workspace is bounded in a box of dimensions $w_{\text{env}} \times l_{\text{env}}$, and the goal region (purple) is defined as the set of all locations that satisfy $\{x > K\}$, i.e., all locations that are beyond K meters in the direction of the robot's initial orientation. The yellow long obstacle is movable, while the red obstacle is static. The robot has to move the yellow box in front of it to reach the goal region. (Middle-Right) A hierarchical control policy is executed to navigate the robot so it may explore the environment's properties and successfully reach the goal region. Concurrently, the proposed obstacle reconstruction module (ORM) is a Transformer neural network that uses localization and proprioception history from the robot to predict object position and dimensions through interaction.

the previous action $v_{\text{prev}}^{\text{des}}$ output from the policy. At any time-step t, the high-level navigation policy outputs: $v_t^{\text{des}} = \pi_{\text{nav}}(\langle \rho_t, \dot{x}_t, \dot{y}_t, v_{t-1}^{\text{des}} \rangle_{t-H}^{t})$. The history H of observations (positions and velocities of the robot) enables the policy to interpret and remember interactions with the obstacles, which in turn aids in navigation. The reward function fuses (a) *goal reaching reward*: a sparse reward for the robot on reaching the goal region; (b) *distance penalty*: a penalty on the robot's distance from the goal region; (c) *time penalty*: incurred for every control step the robot does not reach the goal; (d) *wall collision penalty*: when the robot gets close to the walls; (e) *heading penalty*: if the robot's heading angle is beyond the threshold θ_{thresh} with respect to its initial heading.

C. Dataset Curation

The Obstacle Reconstruction Module (ORM) is trained on a dataset collected by running the navigation policy π_{nav} in a randomly generated simulation environment (see Sec. V for details). The execution is terminated when the robot reaches the goal region, or after $T_{\max}$ time-steps. The time-step when the goal is reached is denoted by t_{goal}. The robot's proprioceptive state s and pose ρ, as well as the N obstacle states $\mathcal{O}$, are recorded at every time step. A *trajectory* $\gamma = \{s_t, \rho_t, \mathcal{O}^t\}_{t=0}^{t=T}$ is defined as a sequence of observations, where $T = \min(t_{\text{goal}}, T_{\max})$. A dataset $\mathcal{D}$ is a collection of trajectories γ_j under M different navigation policies π_{nav}^i, $i \in \{1, 2, .., M\}$ in randomly generated cluttered environments.

Multiple sources of bias may arise in collecting $\mathcal{D}$:

Navigation policy bias. In an environment with multiple homotopic paths, π_{nav} may be biased towards a single homotopy class (e.g., trajectories that favor the right-hand side of the environment). This may lead to recorded trajectories having limited robot-obstacle interactions.

Early termination bias. In different trajectories, the robot interacts with different numbers of obstacles in the scene. The trajectories collected from randomly generated environments are not uniformly divided with respect to the obstacle interaction modes, leading to an unbalanced dataset. For example, trajectories terminate faster in an environment with a single movable obstacle than in environments with multiple obstacles, increasing the number of trajectories recorded for the former.

Contact mode bias. Each trajectory can be categorized into different contact modes, e.g., no contact with any obstacle, direct contact with a static/movable obstacle, or direct contact with a movable obstacle along with indirect static obstacle contact. Imbalances between the modes can lead to bias.

To ensure a high-quality dataset, the curation procedure collects trajectories from M different navigation policies to help cover multiple possible homotopic classes, increasing the diversity of interactions with the obstacles. Pruning trajectories based on contact mode frequency helps mitigate the early termination and contact mode biases.

D. Obstacle Reconstruction Module (ORM)

ORM is a neural network consisting of a causal Transformer encoder [29] followed by a fully-connected `MLP` decoder that predicts a sequence of obstacle parameters for reconstruction from a sequence of proprioception inputs (Fig 3). This network is trained on trajectories in $\mathcal{D}$. The input is a sequence of proprioception data $(s_t, \rho_t)_{t=0}^{T}$ – joint positions, joint velocities, torques applied on the joints, and the robot pose in the world frame. For any trajectory in $\mathcal{D}$, let the time-steps of the first and last contact on obstacle $\mathcal{O}_i$ be denoted by t_i^{first} and t_i^{final}, respectively. Obstacle $\mathcal{O}_i$'s contact window is defined as $\Delta_i^{\text{con}} = [t_i^{\text{first}}, t_i^{\text{final}}]$. $t^{\text{final}} = \max_i(t_i^{\text{final}})$ denotes the time-step of the final contact with any obstacle in $\mathcal{O}$. The obstacle parameters $\mathcal{O}_i$ (see Section III) are augmented to include the contact window information, $\tilde{\mathcal{O}}_i = (\mathbb{I}_i^{\text{contact}}, \mathbb{I}_i^{\text{static}}, x_i, y_i, \theta_i, w_i, l_i)$. The binary variable $\mathbb{I}_i^{\text{contact}}$ indicates whether the obstacle is in its contact window (1), i.e. if the robot or another obstacle $\mathcal{O}_j$ is currently pushing against the obstacle $\mathcal{O}_i$ (where $i \neq j$), or not (0). This augmented $\tilde{\mathcal{O}}^t$ is the ground-truth label for the learning process. The module outputs predictions $\{\hat{\mathcal{O}}^t\}_{t=0}^{t=T}$, where $\hat{\mathcal{O}}^t = \{\hat{\mathcal{O}}_i^t\}_{i=1}^{i=N}$, n is the number of obstacles in the scene and the prediction at time-step t is a function of only the history of inputs up to time-step t in the trajectory. The

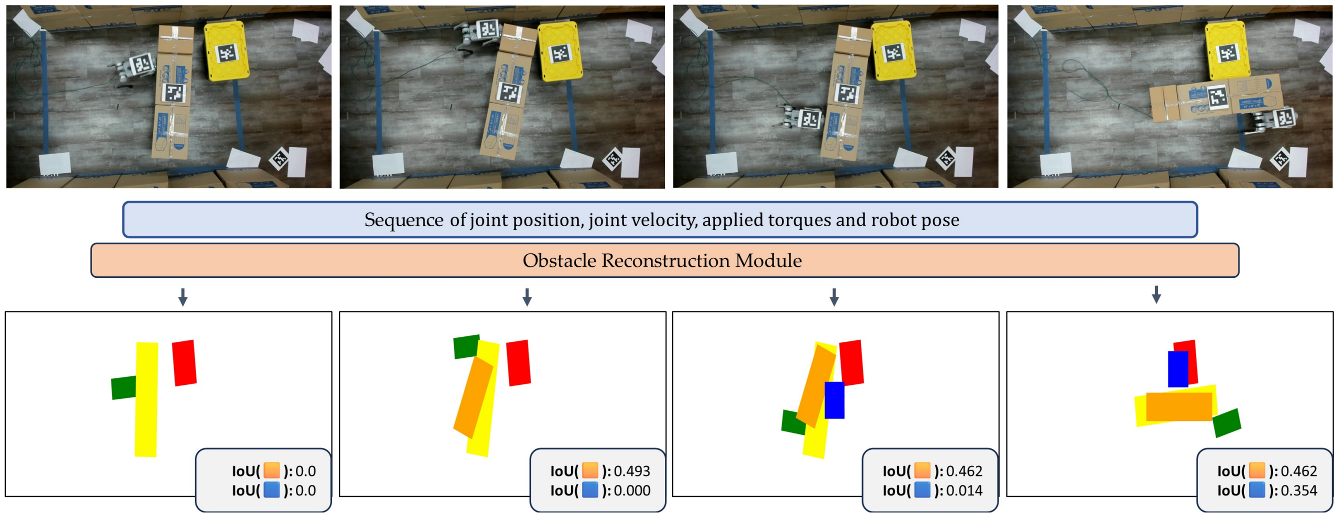
Sequence of joint position, joint velocity, applied torques and robot pose
Obstacle Reconstruction Module
IoU(🟧): 0.0
IoU(🟦): 0.0
IoU(🟧): 0.493
IoU(🟦): 0.000
IoU(🟧): 0.462
IoU(🟦): 0.014
IoU(🟧): 0.462
IoU(🟦): 0.354

Fig. 3: Given as input a sequence of the robot's joint positions, velocities, applied torques, and poses (Top), the Obstacle Reconstruction Module (ORM, middle) is a Transformer-based neural network that outputs the sizes and poses for the different movable (ground truth visualized in yellow) and static (ground truth visualized in red) obstacles in the scene (Bottom, best viewed in color). The reconstruction for the movable and static obstacles are visualized in orange and blue, respectively, and their corresponding Rotated Intersection Over Union (See: Section V-C) values are reported. Higher values represent more accurate reconstructions.

segments of the output sequence not in the contact window for each obstacle $\mathcal{O}_i$ are masked, so the network predicts $\mathcal{O}_i$'s parameters only during its contact window. Contact windows contain rich contact information for obstacle prediction.

The training loss for ORM is a combination of scaled binary cross-entropy BCE losses and a mean squared error MSE supervised learning objective. The loss function for a single sequence with $n \leq N$ interacted obstacles is:

$$\mathcal{L} = \sum_{i=1}^{n} \frac{1}{|\Delta_i^{\text{con}}|} \sum_{t=t_i^{\text{first}}}^{t^{\text{last}}} [\alpha_1 \ \texttt{BCE}(\mathbb{I}_{t,i}^{\text{contact}}, \hat{\mathbb{I}}_{t,i}^{\text{contact}}) + \alpha_2 \ \texttt{BCE}(\mathbb{I}_{t,i}^{\text{static}}, \hat{\mathbb{I}}_{t,i}^{\text{static}}) + \alpha_3 \ \texttt{MSE}((x_{t,i}, y_{t,i}, \theta_{t,i}), (\hat{x}_{t,i}, \hat{y}_{t,i}, \hat{\theta}_{t,i})) + \alpha_4 \ \texttt{MSE}((w_{t,i}, l_{t,i}), (\hat{w}_{t,i}, \hat{l}_{t,i}))].$$

E. System Integration

Offline Training Locomotion policy π_{loc} is first trained using PPO to follow velocity and gait commands. Once trained, π_{loc} is used only in inference mode with a fixed gait for navigation. M navigation policies π_{nav}^i, $i \in \{1, 2, ..., M\}$ are then trained using PPO in unknown cluttered environments using the low-level control policy π_{loc}. A balanced dataset $\mathcal{D}$ of trajectories is collected from M different navigation policies in randomly generated cluttered environments. The ORM is then trained with a supervised objective to partially reconstruct the environments using only proprioception data.
Online Inference For navigating through environments with unknown obstacles, π_{nav} operates in inference mode, internally commanding π_{loc} for robot locomotion. Each control step of the navigation process captures and records the robot's proprioception state in a sequence, preserving a complete history from the trajectory's start. This sequence feeds into the ORM, enabling partial, on-the-fly scene reconstruction. This proposed approach provides an novel method for real-time 2D-scene understanding during navigation.

V. Experiments

Environment Setup For the evaluation, a 12 DoF Unitree Go1-EDU quadruped starts at one end of a walled 4m×2m corridor and must navigate to the goal region located at the other end of the corridor. *Offline, during training*, obstacles $\mathcal{O}_i, i \in [1, N_{\text{max}}]$, are generated in IsaacGym by sampling their physical properties for each obstacle. The obstacles are then placed in random $\texttt{SE}(2)$ configurations with their orientations fixed to 0 with respect to the robot's initial orientation. This may or may not block the path(s) to the goal. When $N_{\text{max}} \geq 2$, the movable obstacle is placed in front of one or more static obstacles. No obstacles are spawned in the goal region. *Domain randomization* of the obstacle's physical parameters aims to reduce the sim2real gap of both the navigation policy and the obstacle prediction module.

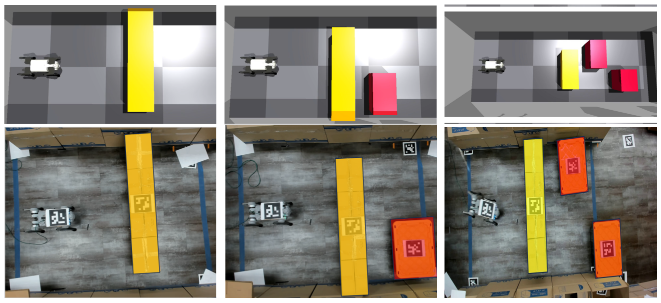

Fig. 4: Examples of Easy (Left), Medium (Middle) and Hard (Right) simulated (top) and real-world (bottom) scenarios considered in the evaluation.

Online, during evaluations, the environments are categorized into *easy*, *medium* and *hard* with respect to the navigation task, where the number of obstacles N is set to 1, 2, and 3 respectively. In the *easy* environments, the robot may only directly interact with the obstacle, whereas, in the *medium* and *hard* ones, the robot may have both direct and indirect obstacle interactions.

Training Details The locomotion and navigation **policies**, π_{loc} and π_{nav}, are trained on 4096 parallel IsaacGym environments for 30K and 1K iterations, respectively. π_{loc} is trained to follow velocity commands in the range $[-0.4, 0.4]m/s$ at a step frequency of 50Hz. Outside this range, the robot is prone to falling when interacting with obstacles while executing π_{nav}. The input to π_{nav} is the history of observations and actions, where the history is a moving window with a maximum length of $H = 30$s and the maximum episode length $T_{\text{max}} = 60$s. π_{nav} provides commands to π_{loc} at a step frequency of 25Hz. For better sim-to-real transfer of π_{nav}, the robot parameters are randomized during training. Multiple versions of π_{nav} are trained on different seeds using different combinations of reward functions mentioned in Section IV-B.

A single **Obstacle Reconstruction Module** (ORM) is trained on a dataset $\mathcal{D}$ of $180K$ trajectories curated from three different navigation policies $\pi^i_{\text{nav}}, i \in \{1, 2, 3\}$. The maximum number of obstacles spawned in the environment $N_{\text{max}} = 3$. The `Transformer` Encoder comprises four self-attention blocks with two attention heads each, followed by a two-layer `MLP` decoder. The inputs to the network are projected to learnable embedding of 512 dimensions with positional information. This network is trained for 20 epochs on an NVIDIA RTX A4500 with 20GB of GPU memory.

Evaluation Metrics The following metrics measure the performance of the ORM. They are calculated for each obstacle $\mathcal{O}_i$ independently at the final time-step t_{final} in the contact window Δ^{con}_i (See Section IV-D).

Rotated Intersection over Union (IoU) evaluates how well the predicted obstacle representation overlaps with the ground-truth. For a single obstacle $\mathcal{O}_i$ at time-step t, it is defined as,

$$\text{IoU}(\mathcal{O}^t_i, \hat{\mathcal{O}}^t_i) = \frac{\text{Area}(\text{Geom}(\mathcal{O}^t_i) \cap \text{Geom}(\hat{\mathcal{O}}^t_i))}{\text{Area}(\text{Geom}(\mathcal{O}^t_i) \cup \text{Geom}(\hat{\mathcal{O}}^t_i))},$$

where $\text{Geom}(\mathcal{O}_i)$ is the geometry of the obstacle $\mathcal{O}_i$ in the workspace. The IoU for obstacle $\mathcal{O}_i$ at the final time-step t_{final} in the contact window Δ^{con}_i is defined as $\text{IoU}^{\text{final}}_i = \text{IoU}(\mathcal{O}^{t_{\text{final}}}_i, \hat{\mathcal{O}}^{t_{\text{final}}}_i)$.

Absolute Error For each obstacle $\mathcal{O}_i$ at time-step t, (i) *Obstacle pose error* is the the absolute error of the predicted $\mathtt{SE}(2)$ pose of the obstacle. (ii) *Obstacle shape error* is the absolute error for the dimensions of the obstacle. The absolute error at the final time-step t_{final} in the contact time window Δ^{con}_i is defined as $\mathcal{E}^{\text{final}}_i = |\mathcal{O}^{t_{\text{final}}}_i - \hat{\mathcal{O}}^{t_{\text{final}}}_i|$. It corresponds to the individual absolute errors in the position of the object in each axis, in its rotation, and length.

VI. Results

A. Simulation Results

Table I reports evaluations of the ORM using the aforementioned metrics on 1000 *Easy*, 1000 *Medium* and 1000 *Hard* benchmarks. Across the benchmarks, the ORM returns a reasonably accurate estimation of the geometry of the obstacles evidenced by the average IoU at the end of the robot's contact as demonstrated qualitatively in Fig. 5. In *Medium* scenarios with a static obstacle, the robot may not be in direct or indirect contact with the static obstacle for a sustained time-period while completing the navigation task. Thus, the IoU reported in this case is relatively lower for the static obstacle. The reported reconstruction for the movable obstacle is surprisingly more accurate in the *Medium* and *Hard* scenarios. The reason is that the robot is in direct contact with the obstacles for longer in the *Medium* and *Hard* benchmarks, which leads to more accurate predictions. The static obstacle's measured IoU is higher in the *Hard* benchmark than the *Medium* benchmark because there is a higher chance of direct interaction with one or the other static obstacle while navigating. The absolute errors further emphasize the trend observed in the IoU. Since the orientation of the static obstacles is fixed across the different benchmarks, their orientation error is omitted.

Category	Type	Metrics				
		IOU ↑	Pose Errors ↓			Shape ↓
			x	y	θ	
Easy	Movable	0.473	0.135	0.101	0.198	0.183
	Static	0.501	0.087	0.104	-	0.172
Medium	Movable	0.496	0.115	0.095	0.201	0.162
	Static	0.331	0.430	0.169	-	0.186
Hard	Movable	0.481	0.128	0.108	0.214	0.172
	Static 1	0.432	0.091	0.138	-	0.117
	Static 2	0.404	0.094	0.151	-	0.120

TABLE I: Quantitative results evaluating the proposed method in simulation. Performance outcomes are averaged across 1000 trials of *Easy*, *Medium*, and *Hard* benchmarks that each contain $N_{\text{max}} = 1, 2, 3$ obstacles, respectively.

B. Real Robot Results

Table II evaluates the ORM on independent trials of 20 *Easy*, 20 *Medium* and 5 *Hard* benchmarks on the real-world setup. For each independent trial, a different navigation policy is uniformly sampled from $\{\pi^i_{\text{nav}}\}^{i=3}_{i=1}$. In the real-world setup, ArUco tags are used to estimate the ground-truth poses of the objects for computing the evaluation metrics. The robot's pose is used as input to the ORM, and can be replaced with onboard localization, e.g., by using an IMU.

Categ.	Type	Metrics				
		IOU ↑	Pose Errors ↓			Shape ↓
			x	y	θ	
Easy	Movable	0.271	0.495	0.151	0.333	0.341
	Static	0.449	0.095	0.162	-	0.099
Medium	Movable	0.277	0.340	0.195	0.407	0.303
	Static	0.212	0.185	0.230	-	0.225
Hard	Movable	0.371	0.203	0.106	0.182	0.282
	Static 1	0.210	0.295	0.408	-	0.192
	Static 2	0.198	0.220	0. 494	-	0.157

TABLE II: Quantitative evaluation of `PROBE` on the real robot. Performance outcomes are averaged across 10 independent trials with a movable obstacle and 10 trials with a static obstacle in the *Easy* category, followed by 20 trials with both obstacles in the *Medium* category, and 5 trials with three obstacles in the *Hard Category*.

The results from the real-world trials confirm the trends observed in the simulation trials. However, multiple factors contribute to a certain decline in the measured metrics – the physical properties of the obstacles, e.g., mass, friction, and restitution, are assumed to be uniformly distributed across the obstacle geometry in simulation, but this is not the case in the real-world setup. The real-world obstacles are also not perfectly box-shaped, contributing to a distribution mismatch

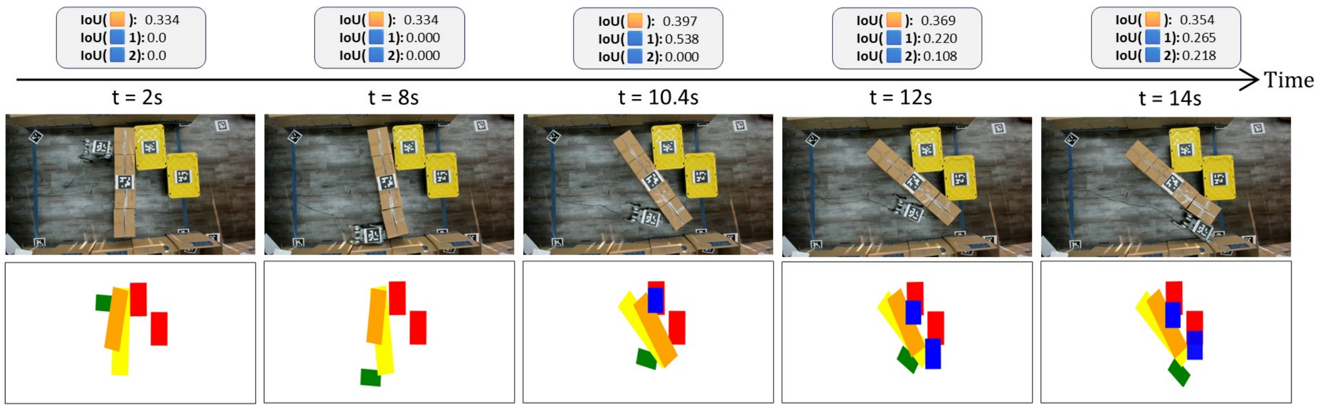

Fig. 5: An example execution of PROBE with a real Unitree Go1, best viewed in color. (Top) Snapshots of the experiment at different timestamps. (Bottom) Obstacle reconstruction returned by PROBE. As the experiment progresses, the robot (green) comes into direct contact with the movable obstacle (ground truth pose in yellow) and indirect contact with the static obstacles (ground truth poses in red). The predictions during the contact window for the movable and static obstacles are visualized in orange and blue, respectively.

between the obstacles and the robot compared to those in the simulation. The *Hard* benchmark contains trajectories that have very minimal direct interaction with the static obstacles in the environment and this reflects in the measured IoU. This is comparable to the evaluated IoU for the *Medium* benchmark.

Fig 5 showcases an example execution of PROBE from a real robot trial on a *Hard* benchmark. The robot makes first contact with the movable obstacle at $t = 2s$ and returns an initial estimate of its pose and shape. At $t = 8s$, π_{nav} directs the robot to the other end of the movable box, creating an opening. As the robot continues pushing the movable box, it enables the ORM to hypothesize about a possible static obstacle at $t = 10.4s$. When the movable box can no longer be pushed at around $t = 12s$, the ORM reconstructs the second static obstacle nested behind the first one. However, since it does not have enough information about how it has manipulated the movable obstacle, the corresponding IoU decreases. The final reconstructions are obtained at the end of the contact window ($t = 14s$).

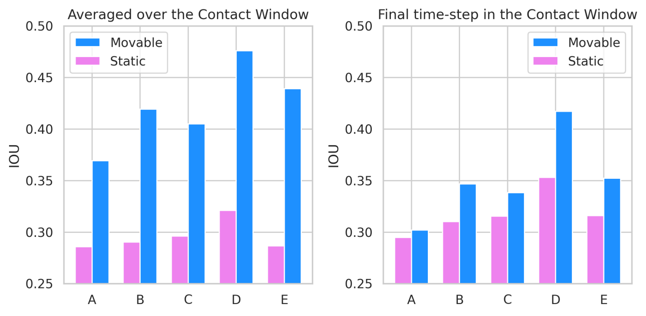

Fig. 6: A: $\{q\}$, B: $\{q, \dot{q}\}$, C: $\{q, \dot{q}, \tau\}$, **D:** $\{q, \dot{q}, \tau, \rho\}$, E: $\{\tau, \rho\}$. The plots show an ablation study on the ORM. The ORM trained with different inputs affects the reconstruction performance measured with the IoU metric.

Ablation Studies. Fig 6 evaluates the importance of the different inputs provided to the ORM using the IoU metric on the same evaluation dataset as Table I. For the ablation study the ORM is trained on environments with $N_{\text{max}} = 2$. The networks A, B, C, D receive as input the history of joint positions q, joint velocities $\dot{q}$, commanded joint torques τ, and the robot pose ρ incrementally. Network E receives as input only the commanded joint torques and the robot's pose. For the movable obstacles, whose features are indicated by the changes in the robot's pose relative to the commanded joint torque, networks D and E perform best in accurately reconstructing them. The IoU of the static obstacles displays a marked improvement when the joint torques are introduced as input. Removing the joint position and velocity information significantly declines reconstruction performance, emphasizing their importance.

VII. Conclusion

Robots that navigate unknown environments benefit from reconstructing their surroundings for planning their actions accordingly. While standard reconstruction methods rely on vision, PROBE is a novel method for 2D scene reconstruction for legged robots which uses only a history of proprioceptive data. This sensing modality is useful in places where vision is not helpful, such as those encountered in search and rescue missions or when obstacles are partially or fully occluded. PROBE employs a transformer-based obstacle reconstruction module, trained in simulation, to map the history of proprioceptive data into an encoded representation of the obstacle geometries in the environment. Experiments are carried out to first evaluate PROBE in simulation and then with a real Unitree Go1 robotic dog navigating an environment with static and movable objects. The evaluation indicates that PROBE can not only detect object locations and sizes but also whether they are movable or static. It can also detect the properties of fully occluded objects by interacting with the occluding frontal objects.

PROBE, however, has some limitations that are worth investigating in future works. Firstly, all experiments were conducted in environments that contained only box-shaped objects. It is interesting to extend and evaluate PROBE on other forms of objects. Furthermore, PROBE was not tested on identifying other physical properties of objects beyond their geometries and whether they are movable. Finally, combining PROBE with reconstruction methods that rely on other modalities, such as touch sensing, can improve the ability to reconstruct scenes.

References

[1] A. Boularias, J. A. Bagnell, and A. Stentz, "Efficient optimization for autonomous robotic manipulation of natural objects," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 2014, pp. 2520–2526. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8414

[2] G. C. Haynes, D. Stager, A. Stentz, J. M. Vande Weghe, B. Zajac, H. Herman, A. Kelly, E. Meyhofer, D. Anderson, D. Bennington, J. Brindza, D. Butterworth, C. Dellin, M. George, J. Gonzalez-Mora, M. Jones, P. Kini, M. Laverne, N. Letwin, E. Perko, C. Pinkston, D. Rice, J. Scheifflee, K. Strabala, M. Waldbaum, and R. Warner, "Developing a Robust Disaster Response Robot: CHIMP and the Robotics Challenge," *J. Field Robot.*, vol. 34, no. 2, pp. 281–304, Mar. 2017. [Online]. Available: https://doi.org/10.1002/rob.21696

[3] C. Atkeson, B. P. Wisely Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. Dedonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. P. Graff, P. He, A. Jaeger, J. Kim, K. Knoedler, L. Li, C. Liu, X. Long, and X. Xinjilefu, "What Happened at the DARPA Robotics Challenge Finals," *Springer Tracts in Advanced Robotics*, pp. 667–684, 04 2018.

[4] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, *et al.*, "Team IHMC's lessons learned from the DARPA robotics challenge trials," *Journal of Field Robotics*, vol. 32, no. 2, pp. 192–208, 2015.

[5] J. Luo, Y. Zhang, K. Hauser, H. A. Park, M. Paldhe, C. G. Lee, M. Grey, M. Stilman, J. H. Oh, J. Lee, *et al.*, "Robust ladder-climbing with a humanoid robot with application to the darpa robotics challenge," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2792–2798.

[6] G. Pratt and J. Manzo, "The DARPA Robotics Challenge," *IEEE Robotics & Automation Magazine*, vol. 20, no. 2, pp. 10–12, 2013.

[7] C. Song and A. Boularias, "Learning to slide unknown objects with differentiable physics simulations," in *Proceedings of Robotics: Science and Systems (RSS), Corvallis, Oregon*, 2020.

[8] M. Stilman, "Navigation among movable obstacles," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, October 2007.

[9] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. K. Allen, "Shape completion enabled robotic grasping," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2442–2447, 2016.

[10] C. Song and A. Boularias, "Inferring 3d shapes of unknown rigid objects in clutter through inverse physics reasoning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 2, 2019.

[11] X. Yan, L. Lin, N. J. Mitra, D. Lischinski, D. Cohen-Or, and H. Huang, "Shapeformer: Transformer-based shape completion via sparse representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[12] J. Bimbo, A. S. Morgan, and A. M. Dollar, "Force-based simultaneous mapping and object reconstruction for robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4749–4756, 2022.

[13] A. Bicchi, J. K. Salisbury, and D. L. Brock, "Contact sensing from force measurements," *The International Journal of Robotics Research*, vol. 12, pp. 249 – 262, 1990.

[14] N. Kurita, S. Sakaino, and T. Tsuji, "Whole-body force sensation by force sensor with end-effector of arbitrary shape," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*. IEEE, 2012, pp. 5428–5433. [Online]. Available: https://doi.org/10.1109/IROS.2012.6386064

[15] H. Liu, K. Nguyen, V. Perdereau, J. Bimbo, J. Back, M. Godden, L. D. Seneviratne, and K. Althoefer, "Finger contact sensing and the application in dexterous hand manipulation," *Auton. Robots*, vol. 39, no. 1, pp. 25–41, 2015. [Online]. Available: https://doi.org/10.1007/s10514-015-9425-4

[16] A. D. Luca, A. O. Albu-Schäffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the dlr-iii lightweight manipulator arm," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1623–1630, 2006.

[17] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[18] C. Stachniss, J. J. Leonard, and S. Thrun, "Simultaneous localization and mapping," *Springer Handbook of Robotics*, pp. 1153–1176, 2016.

[19] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial intelligence review*, vol. 43, pp. 55–81, 2015.

[20] B. Saund, S. Choudhury, S. S. Srinivasa, and D. Berenson, "The blindfolded robot: A bayesian approach to planning with contact feedback," in *International Symposium of Robotics Research*, 2019.

[21] A. Albini, F. Grella, P. Maiolino, and G. Cannata, "Exploiting distributed tactile sensors to drive a robot arm through obstacles," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4361–4368, 2021.

[22] A. M. Gruebele, M. A. Lin, D. Brouwer, S. Yuan, A. C. Zerbe, and M. R. Cutkosky, "A stretchable tactile sleeve for reaching into cluttered spaces," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5308–5315, 2021.

[23] G. S. Martins, R. P. Rocha, F. J. Pais, and P. Menezes, "Clusternav: Learning-based robust navigation operating in cluttered environments," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9624–9630.

[24] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," 2021. [Online]. Available: https://arxiv.org/abs/2107.04034

[25] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," 2022. [Online]. Available: https://arxiv.org/abs/2211.07638

[26] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," 2022. [Online]. Available: https://arxiv.org/abs/2205.02824

[27] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," 2022.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.