QuadPiPS: A Perception-informed Footstep Planner for Quadrupeds With Semantic Affordance Prediction

Max Asselmeier, Ye Zhao, and Patricio A. Vela

***Abstract*—This work proposes QuadPiPS, a perception-informed framework for quadrupedal foothold planning in the perception space. QuadPiPS employs a novel ego-centric local environment representation, known as the legged egocan, that is extended here to capture unique legged affordances through a joint geometric and semantic encoding that supports local motion planning and control for quadrupeds. QuadPiPS takes inspiration from the Augmented Leafs with Experience on Foliations (ALEF) planning framework to partition the foothold planning space into its discrete and continuous subspaces. To facilitate real-world deployment, QuadPiPS broadens the ALEF approach by synthesizing perception-informed, real-time, and kinodynamically-feasible reference trajectories through search and trajectory optimization techniques. To support deliberate and exhaustive searching, QuadPiPS over-segments the egocan floor via superpixels to provide a set of planar regions suitable for candidate footholds. Nonlinear trajectory optimization methods then compute swing trajectories to transition between selected footholds and provide long-horizon whole-body reference motions that are tracked under model predictive control and whole body control. Benchmarking with the ANYmal C quadruped across ten simulation environments and five baselines reveals that QuadPiPS excels in safety-critical settings with limited available footholds. Real-world validation on the Unitree Go2 quadruped equipped with a custom computational suite demonstrates that QuadPiPS enables terrain-aware locomotion on hardware.**

***Index Terms*—Legged Robots, Visual-Based Navigation, Motion and Path Planning, Computer Vision for Locomotion.**

I. Introduction

The recent proliferation of legged robot platforms within both academia [1]–[4] and industry [5]–[9] suggests that complex autonomous tasks such as real-world navigation and locomotion may be attainable in the near future. Formal competitions such as the DARPA Subterranean (SubT) challenge [10]–[12], the BARN [13] and DynaBARN challenges [14], and the ICRA Quadruped Challenge [15] have all demonstrated that for tasks such as these, well-defined hierarchical frameworks enable autonomous and reliable real-world operation.

Furthermore, navigation and planning performance are heavily coupled with the design of the perception system at hand. Poorly designed environment representations bound the capabilities of downstream planning and control, and these representations often act as the computational bottleneck for online frameworks. Fully autonomous platforms must minimize the latency in environment processing while also maximizing planner expressivity. This work augments the environment representation known as the egocan [16] arising from the Planning in the Perception Space (PiPS) [17] philosophy which bypasses sensor preprocessing and casts the act of planning as a sequence of egocentric decision-making problems. Egocan extensions include the geometric attribute of surface normals and the semantic environmental affordances [18] of steppability labels which are both vital to communicating movement opportunities for legged systems to planning and control layers. Moreover, only a single front-facing depth camera is required to construct this full 360° model of the local environment. This work marks the first exploration into how the egocan can be utilized for legged locomotion. It also serves as an investigation into how alternative representations beyond traditional elevation maps [19], [20] and voxel grids [21] may be better suited for an efficient perception-to-action pipeline for local motion planning on quadrupeds.

To perform foothold planning over the egocan, this work draws inspiration from the Augmented Leafs with Experience on Foliations (ALEF) framework [22]. However, existing ALEF implementations [23] perform offline kinematic planning over entirely known environments, and the time required to generate feasible paths ranges from seconds to minutes. In this work, an extension of the Simple Linear Iterative

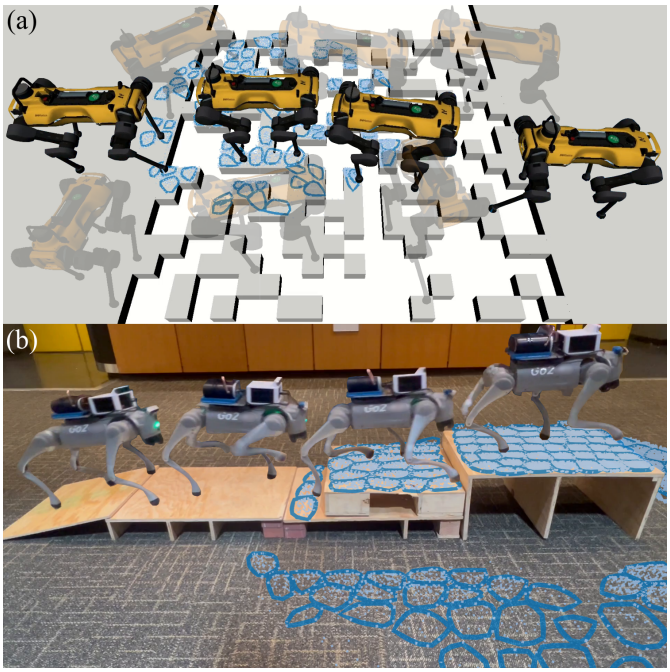

Fig. 1: QuadPiPS framework deployed in simulation on the ANYmal C and on hardware on the Unitree Go2. (a) ANYmal C planning over a set of sparsely distributed stepping stones. Transparent robots represent stance configurations visited during the graph search while solid robots represents the actual path taken by the robot. (b) Unitree Go2 equipped with a custom computational suite for onboard deployment planning over complex terrain including ramps and stairs. Superpixels are represented as blue points and boundaries.

M. Asselmeier, Y.Zhao, and P.A. Vela are with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta GA 30308, USA,{mass, yzhao301, pvela}@gatech.edu

The work of Max Asselmeier is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-2039655. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

Clustering (SLIC) [24] superpixels algorithm oversegments the egocan floor to facilitate exhaustive searching over candidate footholds through a contact mode family transition graph. Then, a nonlinear trajectory optimization formulation synthesizes kinodynamically-feasible and real-time whole body reference trajectories that transition the system between stances. Lastly, a Model Predictive Controller (MPC) and Whole Body Controller (WBC) track this reference trajectory. This entire framework for **Quad**rupedal Foothold **P**lanning **i**n the **P**erception **S**pace is referred to hereafter as **QuadPiPS**.

Analysis of the implementation consists of simulation-based benchmarking over ten diverse environments – featuring gaps, beams, and barriers – and five disparate foothold planning baselines – featuring model-based, model-free, and hybrid approaches – using the ANYmal C quadruped [4]. The experimental outcomes include that (a) the augmented egocan-based perception pipeline improves upon traditional elevation mapping-based representations in environments with numerous planar regions, and (b) the QuadPiPS framework as a whole improves upon heuristic foothold planners in environments with a limited number of available footholds. Validation of the QuadPiPS framework consists of fully onboard and sensor-informed real-world deployment over five challenging terrain configurations – featuring ramps, stairs, and stepping stones – using the Unitree Go2 quadruped [25] equipped with a custom fabricated computational suite, as seen Figure 1.

In summary, the contributions of this work are as follows.

1) Augmentation of the egocan representation to support legged environmental affordances including geometric surface normals and semantic steppability labels
2) Synthetic data generation through primitive shapes-based techniques to construct simulation scenes for predicting semantic environmental affordances
3) Oversegmentation of the egocan floor image using a novel implementation of the SLIC algorithm for sparse multi-channel images to facilitate foothold searching
4) Adaptation of the ALEF framework to support perception-informed, real-time, and dynamically-feasible motion planning for quadrupeds
5) Simulation benchmarking of QuadPiPS against state-of-the-art foothold planners across ten diverse environments using the ANYmal C platform
6) Real-world validation of QuadPiPS on five difficult terrain setups using the Unitree Go2 quadruped fitted with a custom onboard computational suite

II. Related Works

A. Legged Environment Representations

Within an autonomy stack, one of the earliest design decisions is how to represent the robot's local environment. In the literature for legged robots, the most popular model is that of the 2.5D height map or elevation map [19], [26]–[32]. An elevation map is a grid structure represented in a Cartesian frame, either an inertial frame or robot-aligned frame. This representation has roots in the early stages of perception-informed legged locomotion, namely the DARPA Learning Locomotion project [33], [34] where high-resolution height maps were provided to each team. Height maps are popular due to their simple structure, Cartesian frame representation, and open-sourced implementation [35]. However, constructing such a model requires point cloud processing which often necessitates GPU parallelization [19] or multi-threading to operate in real time.

Elevation maps can be used to build further representations such as discrete occupancy maps [36] through height thresholding or fast collision checking, continuous cost maps [37] which build cost functions based on neighboring height information, or signed distance fields [38]–[40] which calculate gradient information regarding distance between the environment and the robot. More recently, neural scenes [20] use learned networks to fill in gaps in environment representations due to occlusions or sensor artifacts. Attention-based map encodings [41], [42] learn attention weights to influence where footholds should be planned. Elevation maps are also often used to construct planar polygonal regions [19], [29], [40], [43]–[46] where further processing steps can be performed to extract planar regions of the environment that can support a footstep.

For environments where it is important to capture multiple height layers — including floors, obstacles, and ceilings — then 3D environment representations can be deployed such as the voxel grid [47]–[49]. In this case, all three dimensions are discretized, giving the user the freedom to model layers at the cost of memory and computation. For the SubT competition [10], which occurred in underground settings such as tunnels and caves, many teams opted for these models [11], [50]. Another flavor of voxel grids is the Octomap [51], which exploits environmental sparsity to efficiently build a volumetric grid.

All representations discussed here require point cloud processing to be constructed. This includes costly steps such as filtering, ray casting, and transformations. By propagating environment information through the egocan, QuadPiPS can construct a height map without any of this preprocessing.

B. Legged Affordances

The distinct ways in which legged robots can interact with the environment, referred to as affordances [18], are far more diverse than those of their wheeled or tracked counterparts. It then becomes important to capture these movement opportunities in the local environment representation to imbue an autonomy stack with the full capabilities of the particular robot platform. While affordances have existed as a concept outside of robotics for many years, they have only been adopted into autonomy frameworks as a way to ground reasoning in embodiments since the DARPA Robotics Challenge [52]–[54].

One of these affordances is what a majority of the literature refers to as *traversability* [37], [55]–[58]. Terrain traversability can be viewed as a continuous score assigned to a subsection of the local environment that reflects the ability to traverse the terrain if a foothold is placed there. Traversability is often calculated as a function of terrain properties such as height discontinuities [36], gradients [32], [59], and curvature [60]. Traversability has also been learned through neural networks [19], [30]. This metric is typically stored in a 2D cost map that is either searched [56], [57] or optimized [37] over for

planning. Some approaches do leverage the image space for terrain processing, but purely for the task of terrain class identification [61]. This motivates a deeper investigation into image space-based representations for legged platforms.

While continuous scoring of terrain is an important metric for footstep planning, the discrete decomposition of the local environment into *steppable* and *non-steppable* regions is paramount. The level of granularity can differ across approaches, with some making a binary classification between these two labels [29], [30] while others further decompose the environment into gait- or behavior-specific regions such as jumping [62]. Another common approach is to decompose the environment into a set of polygons [40], [43], [44], [63] that represent support regions for footholds. Such a representation fits nicely into optimization frameworks for footstep planning. While existing label schemes [62] may be similar to the one presented in this proposed work, all methods discussed here maintain a robot-centric 2D grid-based representation. The image space-based labeling in QuadPiPS fits well with computer vision-based segmentation tasks and aligns with the PiPS ideology.

C. Perception-Planning-Control Frameworks for Quadrupeds

For long-horizon planning and navigation, hierarchical frameworks are essential in decomposing these complex missions into simpler subtasks that can each be addressed through disparate planning and control modules. For the SubT Challenge [10], the CERBERUS team [11], equipped with the ANYmal platform, employed a combination of voxel grids [64] for global planning and height maps for local navigation planning [65]. The team searched over global and local graphs for torso paths. Footholds were resolved explicitly through heuristic placement and local optimization via MPC [37] or implicitly through learned low-level control policies [66]. The CoSTAR team [56] constructed elevation maps that they computed traversability [59] over for risk-aware geometric global planning [67] with the Boston Dynamics Spot. This global planner [68] performed a trajectory library-based search for a global torso path, and a kinodynamic MPC planner then tracked waypoints along the global path. Both teams relied heavily upon graph searches for autonomous planning, but they restricted their search space to the torso pose. QuadPiPS shows that under the proper low-dimensional contact mode family representations, real-time searching for footholds is not only viable, but also preferable in safety-critical environments.

In the Vision-Based Terrain-Aware Locomotion (ViTAL) framework [69], the vision-based foothold adaptation [70] and vision-based pose adaptation algorithms plan according to a shared set of foothold evaluation criteria. ViTAL uses a height map to predict optimal footholds through a convolutional neural network, and a pose optimizer finds a torso pose which maximizes the set of safe footholds available. Then, a whole body controller [71] generates torques that are tracked by a joint impedance controller [72] on the HyQ platform [73]. ViTAL successfully plans over stairs and rough terrain, but it does not deploy in environments where precise foothold planning is necessary. The MIT Mini-Cheetah navigated to a goal and avoided torso-level obstacles using a height map and artificial potential field methods [62] as well as search-based methods [74], both applied in the torso pose space. Grid map filtering provided labels similar to those presented in QuadPiPS which are used to accept or reject heuristically-calculated foothold positions. However, these labels are calculated through pure geometric reasoning such as height gradient thresholding. Heuristic footholds are finally tracked by a whole body controller [75]

More recently, systems opt to hierarchically decompose navigation into simpler subtasks which are then handled via learned modules. Both the ANYmal C [76] and Unitree Go2 [77] navigated through cluttered real-world terrains using a reconstructed elevation map, a learned local navigation policy which provides a velocity command for the system, and a learned locomotion policy capable of maneuvers including walking, crawling, and jumping. These learned frameworks demonstrate impressive robustness and generalization, but they are still tested in environments with numerous available footholds.

All frameworks discussed in this section rely on 2.5D elevation maps. Though, these maps possess computational and representational drawbacks including reliance on point clouds and sensitivity to confined spaces.

D. Planning in the Perception Space (PiPS)

This proposed work was designed according to the PiPS philosophy [17]. A perception space-based approach to planning eschews preprocessing steps for perception data and instead acts directly on the raw data representations that an external sensor provides. This means keeping the sensor data in an egocentric reference frame and recasting the task of local perception-informed planning as a robot-centric decision-making process.

To date, PiPS has provided strong benefits for fast depth space collision-checking [17], collision-free local robot navigation [78]–[80], and task and motion planning [81]. The PiPS approach has also proven effective for both model-based [32], [82] and model-free [83], [84] approaches to legged motion planning. However, torso-level planning and end-to-end efforts have dominated the PiPS literature. A thorough investigation into how the PiPS paradigm should be applied to foothold planning has yet to be performed.

While PiPS can facilitate certain local robot processes such as motion planning and collision checking, the perception space has its drawbacks as well. The rich nearby depth information is offset by sparse environment information at further distances, which limits the spatio-temporal horizons in which planning in the perception space is effective.

III. Overall QuadPiPS Framework

The QuadPiPS framework (see Figure 2) addresses perception-informed motion planning for quadrupedal locomotion over complex terrain. In this work, the only information known a priori is a goal pose in the world to attain.

The original egocan environment representation (Section IV) possesses depth information obtained from an incoming

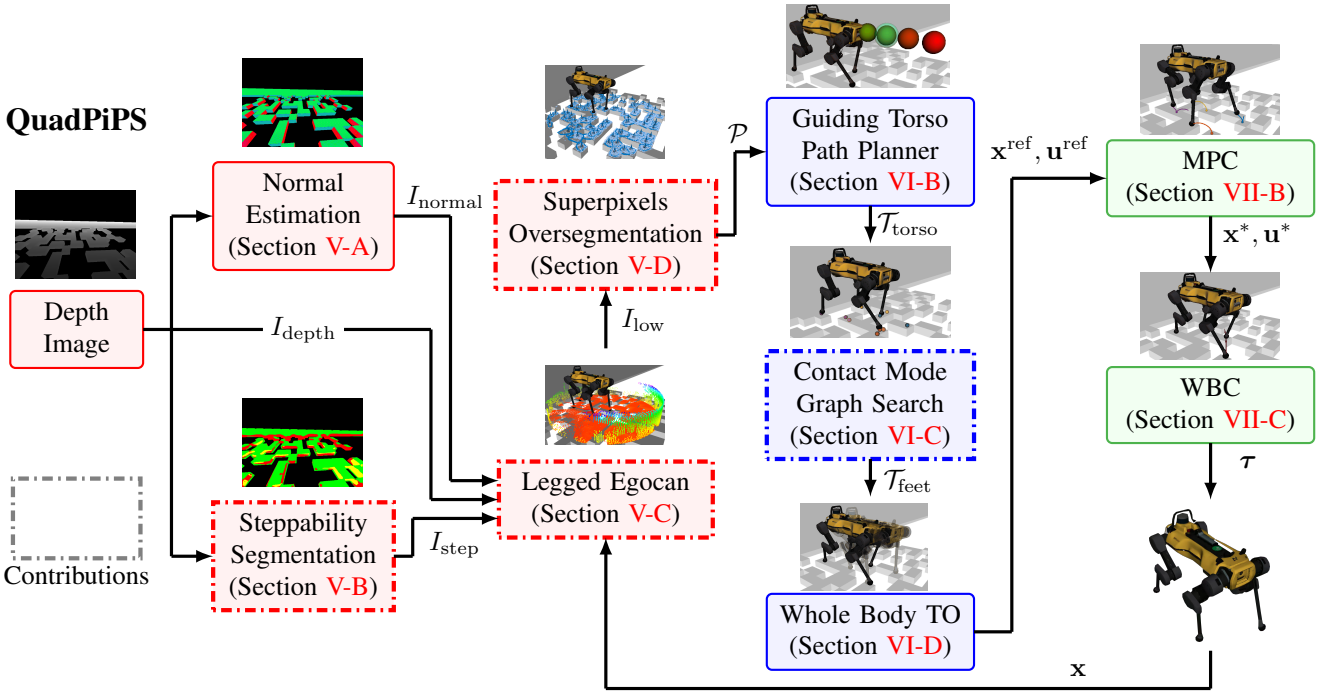

Fig. 2: Workflow for the QuadPiPS framework. The red modules represent the perception pipeline, blue modules represent the planning components, and green modules represent control. Dashed blocks represent novel components that are introduced in this proposed framework. Arrows represent downstream dependencies, meaning that the robot state $\mathbf{x}$ is incorporated into several modules downstream from the semantic egocan.

stereo image stream. QuadPiPS augments the egocan to also include geometric surface normals (Section V-A) at imaging frame rate (30 Hz on hardware) and semantic steppability labels (Section V-B) at 5 Hz. Surface normals are estimated from incoming depth image gradients. Steppability labels are predicted through a semantic segmentation model. New affordance information is inserted into the egocan (Section V-C), and old information is propagated forward in time at 150 Hz to enable a 360° view of the local environment. Near-ground and under-foot data is maintained through an egocan floor image which is oversegmented into superpixel planar regions (Section V-D) at 20 Hz which are convexified (Section V-E) and passed on for foothold planning.

To solve this motion planning problem (Section VI-A), the incoming set of planar regions is used to synthesize a planning graph online for footholds. A coarse torso search (Section VI-B) at 30 Hz informs the foothold graph (Section VI-C) which is built online according to geometric and kinematic constraints, allowing the structure and sparsity of the local environment to inform planning. This planning graph is searched over at 15 Hz with A^* [85] using solely a Euclidean distance heuristic to guide the search. Once this search returns a set of kinematically feasible stance configurations that takes the quadruped from start to goal, a sequence of trajectory optimization subproblems (Section VI-D) are run at 7 Hz to synthesize dynamic whole body trajectories that carry the robot between stance configurations.

Once this whole body reference trajectory (Section VII-A) has been generated, it is sent to an MPC+WBC framework for tracking. The MPC formulation (Section VII-B) runs at 50 Hz and mirrors the formulation used to generate the reference trajectory, while the WBC formulation (Section VII-C) can run at over 1 kHz and takes on a hierarchical QP structure, prioritizing dynamic feasibility constraints over less critical terms including reference trajectory tracking.

IV. Preliminaries

A. Egocan Representation

1) Egocylinder: The concept of the egocylinder will be briefly covered in this section. Readers interested in further details are referred to the existing literature [79], [86].

The concept of the egocylinder is rooted in the PiPS paradigm. LIDARs can provide full 360° views of the environment, but this comes at the cost having to process an unordered, dense point cloud. Traditional depth cameras can only provide a limited field of view of roughly $60° - 90°$ horizontally. This view greatly limits the horizon of planning that one can perform with a single depth image. Additionally, for a task such as perceptive locomotion where the environment is being revealed to the robot in real time, it is important to store prior views as the robot translates and rotates throughout the environment. To address this need, the egocylinder takes world points from a depth image sensor and projects them onto a virtual cylinder that surrounds the robot. Points on the cylinder are discretized into a panoramic image to leverage the favorable aspects of the image space representation such as parallelization. An example egocylinder image can be seen in Figure 3.

The egocylinder takes as input a depth image $I_{\text{depth}} \in \mathbb{R}^{w_{\text{cam}} \times h_{\text{cam}} \times 1}$ of width w_{cam} and height h_{cam}. Each pixel $\mathbf{r}_{\text{cam}} = \begin{bmatrix} u_{\text{cam}} & v_{\text{cam}} \end{bmatrix}^T \in \mathbb{R}^2$ stored in I_{depth}

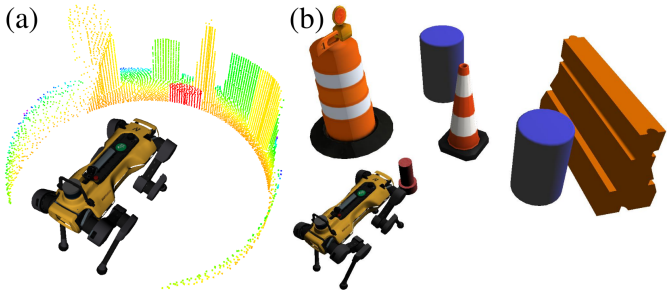

Fig. 3: (a) Example egocylinder and (b) its respective scene. Proximity is represented from near to far as red to blue. The missing parts of the egocylinder are due to depth sensing range limits.

can be parameterized in the camera frame as $\mathbf{p}_{\text{cam}} = \begin{bmatrix} x_{\text{cam}} & y_{\text{cam}} & z_{\text{cam}} \end{bmatrix}^T \in \mathbb{R}^3$ where z_{cam} is the depth value read off of I_{depth}. The point $\mathbf{p}_{\text{cam}}$ can be computed with the focal length f_{cam} and central pixel $\mathbf{r}_{\text{cam},0} = \begin{bmatrix} u_{\text{cam},0} & v_{\text{cam},0} \end{bmatrix}^T \in \mathbb{R}^2$.

New points from incoming sensor streams are projected onto the virtual cylinder that surrounds the ego-robot. This cylinder is propagated forward in time as the robot moves. This cylindrical surface is discretized into an image $I_{\text{cyl}} \in \mathbb{R}^{w_{\text{cyl}} \times h_{\text{cyl}} \times 1}$ of width w_{cyl} and height h_{cyl} which stores the egocylindrical range of each point. This range allows the egocylinder to retain information behind the robot. Therefore, incoming Cartesian camera frame information is transformed into egocylindrical coordinates via

$$\mathbf{p}_{\text{cyl}} = \begin{bmatrix} \rho_{\text{cyl}} \\ \theta_{\text{cyl}} \\ z_{\text{cyl}} \end{bmatrix} = T_{\text{c2e}}(\mathbf{p}_{\text{cam}}) = \begin{bmatrix} \sqrt{x_{\text{cam}}^2 + y_{\text{cam}}^2} \\ \text{Arg}(x_{\text{cam}} + \mathbf{j} z_{\text{cam}}) \\ y_{\text{cam}} \end{bmatrix} \quad (1)$$

where $\mathbf{p}_{\text{cyl}} \in \mathbb{R}^3$ is the egocylindrical coordinate, ρ_{cyl} represents the range, θ_{cyl} is the angle, and z_{cyl} is the height. The matrix $T_{\text{c2e}} \in SE(3)$ transforms from Cartesian to egocylindrical coordinates. Resulting egocylindrical coordinates are then mapped to image coordinates $\mathbf{r}_{\text{cyl}} = \begin{bmatrix} u_{\text{cyl}} & v_{\text{cyl}} \end{bmatrix}^T \in \mathbb{R}^2$ using the egocylinder projection matrix $K_{\text{cyl}} \in \mathbb{R}^{2\times 3}$:

$$\mathbf{r}_{\text{cyl}} = K_{\text{cyl}} \begin{bmatrix} \theta_{\text{cyl}} \\ z_{\text{cyl}} \\ 1 \end{bmatrix} \text{ with } K_{\text{cyl}} = \begin{bmatrix} f_{\text{cyl}} & 0 & u_{\text{cyl},0} \\ 0 & f_{\text{cyl}} & v_{\text{cyl},0} \end{bmatrix} \quad (2)$$

where $f_{\text{cyl}} = \cot\left(\frac{2\pi}{w_{\text{cyl}}}\right)$, $u_{\text{cyl},0} = \frac{w_{\text{cyl}}}{2}$, and $v_{\text{cyl},0} = \frac{h_{\text{cyl}}}{2}$.

Existing egocylindrical coordinates are mapped back to Cartesian coordinates, propagated forward in time under a Euclidean transform for the induced pose $T_{\text{move}} \in SE(3)$ between the prior and current timesteps, and mapped back to egocylindrical coordinates as

$$\mathbf{p}'_{\text{cyl}} = T_{\text{c2e}} \circ T_{\text{move}} \circ T_{\text{e2c}}(\mathbf{p}_{\text{cyl}}) \text{ with } T_{\text{e2c}} = \begin{bmatrix} \rho_{\text{cyl}} \cos(\theta_{\text{cyl}}) \\ z_{\text{cyl}} \\ \rho_{\text{cyl}} \sin(\theta_{\text{cyl}}) \end{bmatrix} \quad (3)$$

where $T_{\text{e2c}} \in SE(3)$ transforms from egocylindrical coordinates to Cartesian coordinates.

2) Egocan: During egocylindrical propagation, points that do not map onto the panoramic image are lost. This means that points that pass over or under the cylinder are not retained. Note that in Figure 3, a portion of the short red rod in front of the robot is *not* captured on the egocylinder as it passes underneath the egocylinder field of view. To address this need, the egocan extends the egocylinder by adding upper and lower *egocaps* on the top and bottom of the egocylindrical image to create a closed surface. The egocan was introduced for aerial vehicles [16], but has yet to see use in locomotion. For this work, the lower cap, visualized in Figure 4, is used to capture prior scenes as they travel underfoot, which are later used for footstep planning. Functionally, the egocan floor surface acts much like a height map. However, no expensive point cloud processing is required to build it. Furthermore, the egocan floor information is maintained as an image which allows QuadPiPS to leverage the benefits of the data structure.

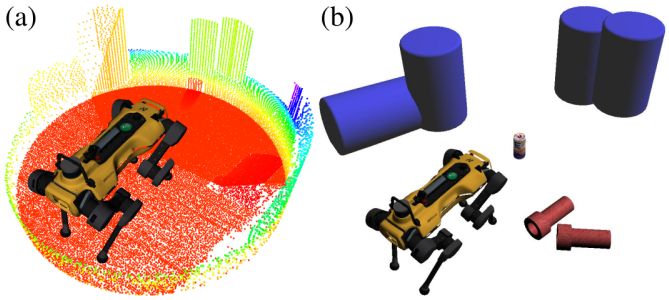

Fig. 4: (a) Example egocan and (b) its respective scene. This representation captures nearby ground elevation information including the can in front of the robot which is otherwise missed by the egocylinder.

Egocap surfaces map to the images $I_{\text{low}}, I_{\text{up}} \in \mathbb{R}^{w_{\text{cap}} \times h_{\text{cap}} \times 1}$ with width w_{cap} and height h_{cap} where $w_{\text{cap}} = w_{\text{low}} = w_{\text{up}} = h_{\text{cap}} = h_{\text{low}} = h_{\text{up}}$. A new or existing point $\mathbf{p}_{\text{cyl}}$ that does not map onto I_{cyl} is mapped onto I_{low} if $z_{\text{cyl}} > 0$ and I_{up} otherwise. Throughout this work, I_{low} may be referred to as the egocan floor image. The upper image I_{up} is not used in this work and will no longer be addressed for clarity. The height information z_{cyl} will be placed at the lower image coordinate $\mathbf{r}_{\text{low}} = \begin{bmatrix} u_{\text{low}} & v_{\text{low}} \end{bmatrix}^T \in \mathbb{R}^2$ where

$$\mathbf{r}_{\text{low}} = K_{\text{low}} \begin{bmatrix} \frac{x_{\text{cam}}}{|y_{\text{cam}}|} \\ \frac{z_{\text{cam}}}{|y_{\text{cam}}|} \\ 1 \end{bmatrix} \text{ with } K_{\text{low}} = \begin{bmatrix} f_{\text{low}} & 0 & u_{\text{low},0} \\ 0 & f_{\text{low}} & v_{\text{low},0} \end{bmatrix} \quad (4)$$

and the lower cap focal length $f_{\text{low}} = \frac{v_{\text{fov}} w_{\text{low}}}{4}$ where v_{fov} is the vertical field of view of the egocylinder which is a user-specified parameter between $45°$ and $90°$. The lower cap center coordinate is expressed as $\mathbf{r}_{\text{low},0} = \begin{bmatrix} u_{\text{low},0} & v_{\text{low},0} \end{bmatrix}^T \in \mathbb{R}^2$ where $u_{\text{low},0} = \frac{w_{\text{low}}}{2}$ and $v_{\text{low},0} = \frac{h_{\text{low}}}{2}$.

V. Legged Egocan

A. Image Space Normal Estimation

For the task of perceptive locomotion over complex terrain, additional environmental affordances are required beyond just depth. The QuadPiPS framework adopts the depth image gradients-based approach [87] for surface normal estimation in the image space. The approach is briefly covered here, and more information can be found in the original work. The subscript $(\cdot)_{\text{cam}}$ for the camera frame will be temporarily dropped in this section for clarity.

This normal estimation technique relies on building a local planar approximation at the point $\mathbf{p}(u, v)$ using its directional derivatives $\mathbf{p}_u(u, v)$ and $\mathbf{p}_v(u, v)$. These directional derivatives can be computed as

$$\begin{aligned}\mathbf{p}_u(u,v) &= \left(\frac{\partial x(u,v)}{\partial u}, \frac{\partial y(u,v)}{\partial u}, \frac{\partial z(u,v)}{\partial u}\right) \\ \mathbf{p}_v(u,v) &= \left(\frac{\partial x(u,v)}{\partial v}, \frac{\partial y(u,v)}{\partial v}, \frac{\partial z(u,v)}{\partial v}\right).\end{aligned} \tag{5}$$

The partial derivative terms can then be calculated as

$$\begin{aligned}\frac{\partial x(u,v)}{\partial u} &= \frac{z(u,v)}{f} + \frac{(u-u_0)}{f}\frac{\partial z(u,v)}{\delta u} \\ \frac{\partial y(u,v)}{\partial u} &= \frac{(y-y_0)}{f}\frac{\partial z(u,v)}{\delta u} \\ \frac{\partial x(u,v)}{\partial v} &= \frac{(u-u_0)}{f}\frac{\partial z(u,v)}{\delta v} \\ \frac{\partial y(u,v)}{\partial v} &= \frac{z(u,v)}{f} + \frac{(y-y_0)}{f}\frac{\partial z(u,v)}{\delta v},\end{aligned} \tag{6}$$

with

$$\begin{aligned}\frac{\partial z(u,v)}{\delta u} &\approx z(u+1,v) - z(u,v) \\ \frac{\partial z(u,v)}{\delta v} &\approx z(v+1) - z(u,v).\end{aligned} \tag{7}$$

Lastly, the directional cross product can be calculated as

$$\mathbf{n}(u,v) = \mathbf{p}_v(u,v) \times \mathbf{p}_u(u,v) \tag{8}$$

and subsequently normalized:

$$\hat{\mathbf{n}}(u,v) = \frac{\mathbf{n}(u,v)}{\|\mathbf{n}(u,v)\|_2}. \tag{9}$$

These pixel-wise normals comprise $I_{\text{normal}} \in \mathbb{R}^{w_{\text{cam}} \times h_{\text{cam}} \times 3}$. Example images and normals are shown in Figure 5.

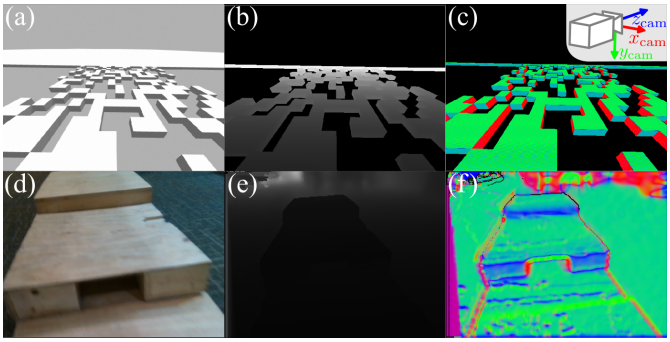

Fig. 5: Example scenes with their estimated image space normals. The top row is an example scene in simulation and the bottom row is an example scene on hardware. The left images (a) and (d) are color images of the two scenes, the middle images (b) and (e) are depth images, and the right images (c) and (f) are the estimated normals. The RGB color scheme corresponds to the xyz axes of the camera frame where the absolute value of the estimated surface normal vector components are used to calculate colors. The region of purple pixels in (f) are due to artifacts from a hole-filling filter.

B. Steppability Segmentation

Depth and normal information make up the geometric affordances present in the legged egocan representation. However, it is also important to be able to semantically reason about local terrain to encode preferences or behaviors that cannot be ascertained through geometry alone. For foothold planning, semantic labels for preferring to step on, over, or around objects can help guide the planner away from regions that are geometrically permissible, but not preferable.

In this section, the process in which steppability labels are predicted for the local environment is detailed. First, the simulation scene creation process used to generate synthetic steppability data is discussed. Here, QuadPiPS adapts a primitive shapes-based technique that was previously used for manipulation tasks [88], [89]. Primitive shapes-based approaches hypothesize that the geometry of graspable objects can be decomposed into a set of primitive shapes where each primitive shape class has a particular family of effective grasps. For data collection, the ground truth labels of these objects can then be ascertained through the color of the primitives within the simulation scene. QuadPiPS applies this ideology to generate class labels for steppability. The synthetic scenes used for training data are assembled according to a key set of design parameters that are detailed below.

1) Primitive Shape Classes: Nine primitive shape classes are used to build scenes: *Cuboid*, *Ramp*, *Cylinder*, *Sphere*, *Semisphere*, *Pole*, *Pipe*, *Tube*, and *Floor*. The Cuboid and Ramp classes are parameterized by length l_{prim}, width w_{prim}, and height h_{prim}, the Cylinder class is parameterized by radius r_{prim} and height h_{prim}, the Sphere and Semisphere classes are parameterized by radius r_{prim}, and the Pipe, Pole, and Tube classes are parameterized by length l_{prim} and radius r_{prim}. The floor class is non-parametric in that all of its instances are 4 m $\times$ 4 m with a set of 3D surface equations to capture non-flat terrain. Each shape class is shown in Figure 6.

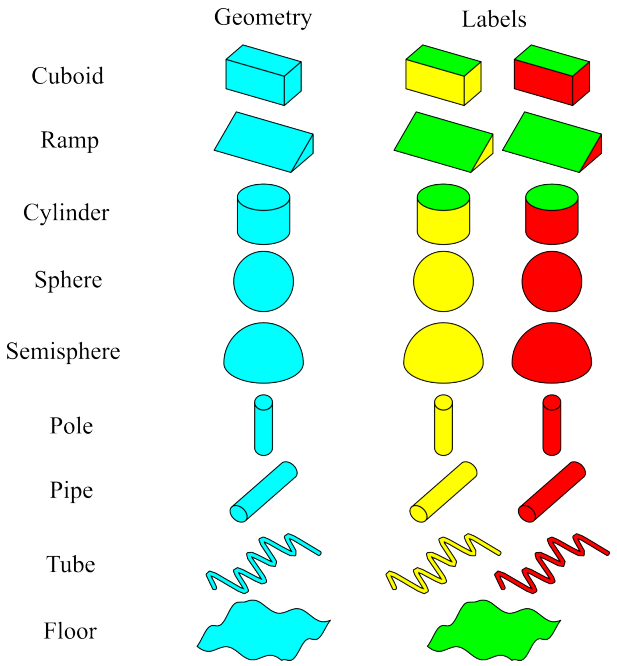

Fig. 6: Example visualizations of the primitive shape classes used to construct the simulation scenes that comprise the synthetic data for training the steppability policy. Green corresponds to steppable, yellow corresponds to passable, and red corresponds to non-passable.

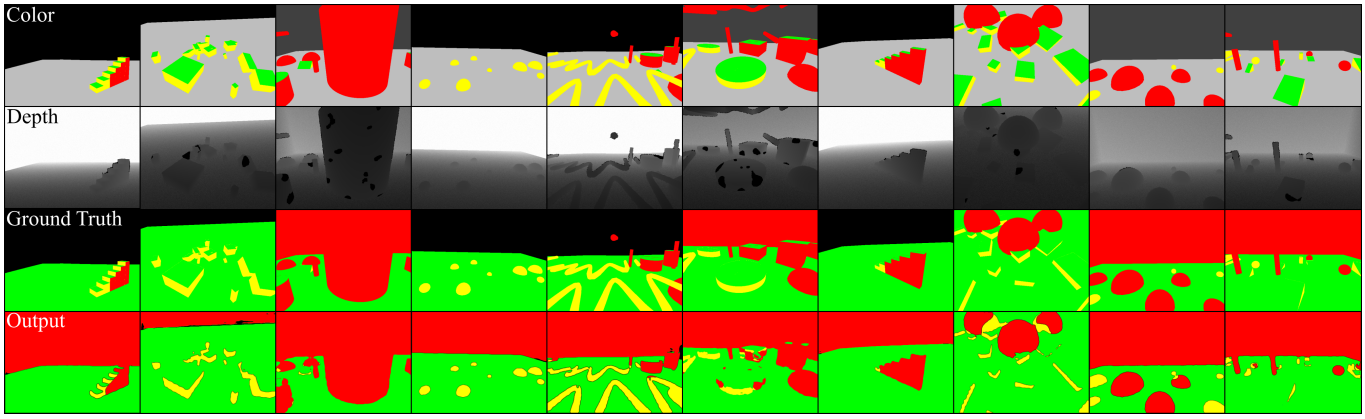

Fig. 7: Example simulation scenes along with steppability model predictions. Top row: color images. Second row: depth images. These images also display the noisy artifacts injected into the training data including surface and boundary depth loss along with Gaussian blur. Third row: ground truth steppability labels of the corresponding column's input depth image. Bottom row: model outputs for the corresponding column's input depth image.

2) Primitive Shape Steppability Policies: Each primitive shape class is assigned a mesh-based steppability policy that defines which faces of the shapes should be assigned which labels. The three labels used are:

- **Steppable (Green):** can support a stable foothold
- **Passable (Yellow):** can *not* support a stable foothold, but can be stepped over by a foot swing trajectory
- **Non-passable (Red):** can *not* support a stable foothold and can *not* be stepped over by a foot swing trajectory

For Cuboids, Ramps, and Cylinders, the top face is labeled as steppable due to its planar geometry. If the height discontinuity surrounding the primitive exceeds a maximum swing height for the robot leg, defined as $h_{\text{swing}}^{\max} = 0.15$ m, then all vertical faces are labeled as non-passable. Otherwise, the vertical faces are labeled as passable. For Spheres, Semispheres, Poles, Pipes, and Tubes, the entire primitive is labeled as non-passable if the z-coordinate of the top of the primitive exceeds $h_{\text{swing}}^{\max}$. Otherwise, the entire primitive is labeled as passable. Lastly, floors are labeled as completely steppable.

3) Primitive Shape Pose: The six-dimensional pose of each primitive shape in the scene frame, $\mathbf{q}_{\text{scene}} \in \mathbb{R}^6$, can be set according to desired scene attributes. For instance, scenes can be set to feature clusters of primitive shapes localized within a particular region of the scene, the primitives can be scattered throughout the scene, or the poses can be overwritten to accept manually defined entries if the user wants to create more contrived scenes with staircases or stepping stones. The z-dimension of all shapes, z_{scene}, is restricted so that they are placed on support surfaces outside of Pipes and Tubes, which are allowed to be suspended in air. The orientations of Cuboids, Ramps, and Cylinders are restricted to ensure that the face labeled as steppable is the top face in the scene.

4) Camera pose: The pose of the camera for each simulation scene can also be parameterized to emulate expected real-world circumstances for onboard sensing. Given the desired application of quadrupedal locomotion, the camera is set at a nominal height and pitch to approximate the pose of the depth camera attached to the quadrupedal hardware platform. To capture multiple frames of a single scene, the camera is set to follow a prescribed trajectory that approximates how the quadruped's torso would move through a real-world environment. Gaussian noise is also applied to all six pose dimensions to capture the jitter that the camera would experience during locomotion in the real world.

5) Scene Environment: Lastly, the overall synthetic environment that the primitive shapes are placed within can also be controlled. A random binary variable is sampled which chooses between indoor environments with walls and a ceiling and outdoor environments with an infinite horizon.

6) Domain Alignment: The process of bi-directional domain alignment is performed in which (a) simulation data is corrupted to approximate defects observed in real-world sensing, and (b) real-world sensor data is post-processed to mitigate these flaws and artifacts.

For simulation data, a sequence of corrupting filters is applied to the depth data. Gaussian noise and an oil painting filter are applied to corrupt the depth data within dilated object boundaries. Then, randomly connected groups of pixels are voided on the boundaries and surfaces of objects to approximate depth loss. On hardware, a sequence of post-processing filters provided by the RealSense API including decimation and hole-filling [90] is applied to the depth data.

From each scene, a depth image I_{depth} is extracted along with a ground truth steppability mask $I_{\text{step}}^{*} \in \mathbb{R}^{w_{\text{cam}} \times h_{\text{cam}} \times 1}$.

7) Model Training: A DeepLabV3+ [91] model from the Detectron2 deep learning library [92] is used for steppability prediction. The dataset contains 600 scenes with 5 frames each. In total, dataset generation took roughly 12 hours. The generated data was put into 80/10/10% splits for training, validation, and test subsets, respectively. Model training took 65 minutes on a Dell Precision 5820 Tower with an Intel Xeon W-2223 CPU (single-thread passmark of $2,199$; multi-thread score of $8,586$) and a NVIDIA T1000 GPU. Example scene data is shown in Figure 7. Training loss and intersection-over-union (IoU) plots can be seen in Figure 8.

The model output is represented as the predicted steppability mask $I_{\text{step}} \in \mathbb{R}^{w_{\text{cam}} \times h_{\text{cam}} \times 1}$ in which each pixel contains a label $l(u_{\text{cam}}, v_{\text{cam}})$.

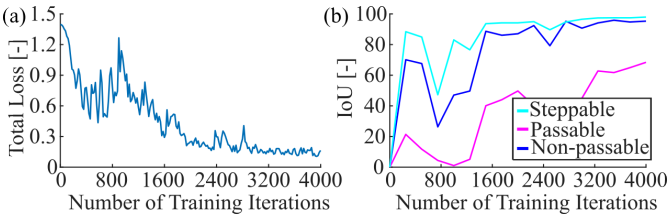

Fig. 8: Results of training. (a) Total training loss over the training iterations. (b) Intersection-over-union (IoU) of all classes over the training iterations.

C. Legged Egocan Representation

The legged egocan takes as input a multi-channel affordance image $I_{\text{afford}} = \begin{bmatrix} I_{\text{depth}} & I_{\text{normal}} & I_{\text{step}} \end{bmatrix} \in \mathbb{R}^{w_{\text{cam}} \times h_{\text{cam}} \times 5}$. The egocan propagation step is kept the same, the only difference being that additional normal and steppability information is stored along with depth information. To reflect this, the floor image also takes on the dimensionality $I_{\text{low}} \in \mathbb{R}^{w_{\text{cap}} \times h_{\text{cap}} \times 5}$.

D. Superpixels Oversegmentation

Once the egocan has been populated, it is still necessary to identify what parts of the terrain can support footholds. It is common to decompose a height map into a set of planar convex regions that can support footholds [40]. While such regions work well for linear optimization constraints, the process for searching over these regions is unclear. If the ego-robot is presented with a large, flat floor, one large convex region may be generated. This is not conducive to a search framework. For this reason, QuadPiPS adopts an oversegmentation approach via superpixels.

Superpixel algorithms [24], [93] are intended to cluster pixels into perceptually homogeneous regions which are evenly distributed throughout the image space. These homogeneous regions, or superpixels, are meant to modify or replace the uniform gridding of the pixel matrix within the image. In the case of this work, superpixels act as an insightful primitive that allows the foothold graph search to not only select which terrain features to plan footholds on, but also decide where within the feature to step.

1) Health Check: As input, this superpixel algorithm takes the egocan floor image I_{low}. Prior to passing I_{low} in for oversegmentation, a health check is performed on I_{low}. Due to the nature of the egocan propagation, the floor image is sparse. Normally, superpixel algorithms are deployed on full RGB or RGB-D [94], [95] images. However, in this use case, a majority of the pixels lack data. First, pixels with no data or invalid data are set to zero. Pixels with unstable normals, defined as $\hat{\mathbf{n}}_{\text{cam}}(u_{\text{low}}, v_{\text{low}}) \cdot -\hat{\mathbf{e}}_{y_{\text{cam}}} \leq 0.5$, are also set to zero to avoid building planar regions with them. The term $\hat{\mathbf{e}}_{y_{\text{cam}}}$ represents the basis vector for the y-dimension of the camera frame. Pixels with labels $l(u_{\text{low}}, v_{\text{low}})$ that are not assigned steppable are also removed.

2) Image Preprocessing: When the egocan is initialized, I_{low} is empty. However, this yields no planar regions. QuadPiPS assumes that planning begins on flat, obstacle-free terrain. Therefore, regions of the floor image that have not yet been populated by sensor data are initialized to default heights and normals that represent standard ground floor terrain.

3) Superpixels Approach: The oversegmentation algorithm can be understood as a modified version of the Simple Linear Iterative Clustering (SLIC) [24] algorithm that is designed to handle sparse multi-channel images. The initial set of superpixel cluster centers $\mathcal{S}$ are defined by uniformly sampling pixels in I_{low} at regular grid steps $s_{\text{low}} = \sqrt{\frac{n_{\text{low}}}{k_{\text{des}}}}$. The value $n_{\text{low}} = w_{\text{low}} \times h_{\text{low}}$ is the number of pixels in the floor image and k_{des} is the user-defined desired number of superpixels. Each cluster $\zeta \in \mathcal{S}$ contains a pixel $\mathbf{r}_\zeta = \begin{bmatrix} u_\zeta & v_\zeta \end{bmatrix}^T \in \mathbb{R}^2$, a point $\mathbf{p}_{\text{cam}}(u_\zeta, v_\zeta) \in \mathbb{R}^3$ which is abbreviated as $\mathbf{p}_\zeta$, and a normal $\hat{\mathbf{n}}_{\text{cam}}(u_\zeta, v_\zeta) \in \mathbb{R}^3$ which is abbreviated as $\hat{\mathbf{n}}_\zeta$. The initial clusters undergo a refinement step in which they are moved to the centroid of the $\frac{s_{\text{low}}}{4} \times \frac{s_{\text{low}}}{4}$ neighborhood surrounding the initial cluster center. This helps in generating smoother superpixel regions.

Then, the proposed algorithm iterates through the set of clusters and calculates the distance between the current cluster ζ and the current pixel $\mathbf{r}_{\text{low}}$. Only pixels in a $2s_{\text{low}} \times 2s_{\text{low}}$ neighborhood are evaluated. The distance metric scores deviation from the local plane formed by the cluster ζ as well as cluster compactness. For ζ and $\mathbf{r}_{\text{low}}$, the complete distance calculation is

$$\begin{aligned} d_{\text{total}}(\zeta, \mathbf{r}_{\text{low}}) = w_n \frac{d_n(\zeta, \mathbf{r}_{\text{low}})}{d_n^{\max}} + w_p \frac{d_p(\zeta, \mathbf{r}_{\text{low}})}{d_p^{\max}} \\ + w_w \frac{d_w(\zeta, \mathbf{r}_{\text{low}})}{d_w^{\max}} + w_c \frac{d_c(\zeta, \mathbf{r}_{\text{low}})}{d_c^{\max}}. \end{aligned} \tag{10}$$

The term

$$d_n(\zeta, \mathbf{r}_{\text{low}}) = 1 - \hat{\mathbf{n}}_\zeta \cdot \hat{\mathbf{n}}_{\text{cam}}(u_{\text{low}}, v_{\text{low}}) \tag{11}$$

measures deviation from the cluster normal $\hat{\mathbf{n}}_\zeta$ and the current pixel's normal $\hat{\mathbf{n}}_{\text{cam}}(u_{\text{low}}, v_{\text{low}})$. The term

$$d_p(\zeta, \mathbf{r}_{\text{low}}) = |(\mathbf{p}_\zeta - \mathbf{p}_{\text{cam}}(u_{\text{low}}, v_{\text{low}})) \cdot \hat{\mathbf{n}}_\zeta| \tag{12}$$

measures distance between the current pixel's camera frame position $\mathbf{p}_{\text{cam}}(u_{\text{low}}, v_{\text{low}})$ and the cluster plane $(\mathbf{p}_\zeta, \hat{\mathbf{n}}_\zeta)$. The term

$$d_w(\zeta, \mathbf{r}_{\text{low}}) = \|\mathbf{p}_\zeta - \mathbf{p}_{\text{cam}}(u_{\text{low}}, v_{\text{low}})\|_2, \tag{13}$$

measures Euclidean distance between the current point and the cluster center, and the final term

$$d_c(\zeta, \mathbf{r}_{\text{low}}) = \sqrt{(u_\zeta - u_{\text{lower}})^2 + (v_\zeta - v_{\text{lower}})^2} \tag{14}$$

measures Euclidean pixel distance between the current pixel and the cluster center. The values $w_{(\cdot)}$ are positive scalar values for weighting importance. The notation $d_{(\cdot)}^{\max}$ represents the maximum distance for the respective terms. These are applied to scale the distance terms.

Once pixel-wise distances are calculated and clusters are assigned, cluster centroids are updated using the newly assigned pixels. Cluster normals are also refined via RANSAC [96]. This assignment and update process is repeated for a user-specified number of iterations n_{iter}. A diagram for the workflow is also shown in Figure 9.

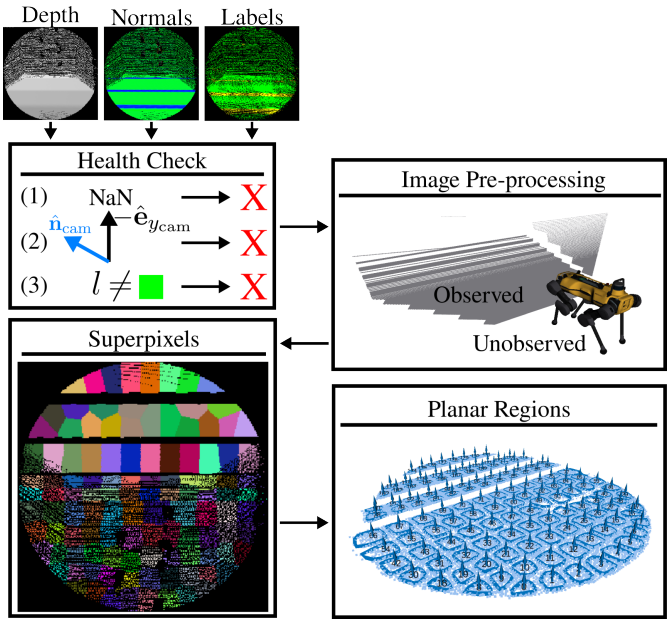

Fig. 9: Information flow for the proposed superpixels oversegmentation and planar region extraction method. The input is the multi-channel floor image I_{low}. First, the image is cleaned to remove all invalid pixels as well as pixels corresponding to non-upright normals and non-steppable labels. Next, regions of the image that have not yet been populated from sensor data are initialized to standard ground plane data. Then, the superpixels algorithm is run to generate a set of superpixel clusters, and those clusters are then passed to a convex hullification step to obtain closed planar regions.

E. Convex Planar Region Generation

Downstream foothold planning expects a set of convex planar regions as a characterization of the local environment. Therefore, these superpixel regions must be convexified.

First, all points in a superpixel cluster ζ are projected onto the plane described by $(\mathbf{p}_\zeta, \hat{\mathbf{n}}_\zeta)$ from the region's centroid. Then, this set of 2D points is passed to the Graham Scan algorithm [97] to compute the convex hull. The set of points defining the hull are then passed on to foothold planning as a set of planar regions $\mathcal{P}$. Each region $\varphi \in \mathcal{P}$ is defined by a pose $\mathbf{q}_\varphi \in \mathbb{R}^6$ in an inertial frame and a list of 2D boundary points $\{\mathbf{b}_{\varphi,0}, \mathbf{b}_{\varphi,1}, \ldots, \mathbf{b}_{\varphi,n_b}\} \in \mathbb{R}^{2\times n_b}$ that lie in the plane defined by $\mathbf{q}_\varphi$. The set size n_b can vary between regions.

VI. Quadrupedal Motion Planning

A. Problem Statement

Consider a quadrupedal robot with a configuration space $\mathcal{Q} \subset \mathbb{R}^{6+n_j}$. The robot configuration $\mathbf{q} \in \mathcal{Q}$ is comprised of a torso pose $\mathbf{q}_{\text{torso}} \in \mathbb{R}^6$ represented in an inertial frame and a sequence of n_j joints $\mathbf{q}_j \in \mathbb{R}^{n_j}$. The pose $\mathbf{q}_{\text{torso}}$ can be further decomposed into an orientation $\boldsymbol{\theta}_{\text{torso}} = \begin{bmatrix}\alpha_{\text{torso}} & \beta_{\text{torso}} & \gamma_{\text{torso}}\end{bmatrix}^T \in \mathbb{R}^3$ and position $\mathbf{p}_{\text{torso}} = \begin{bmatrix}x_{\text{torso}} & y_{\text{torso}} & z_{\text{torso}}\end{bmatrix}^T \in \mathbb{R}^3$. The proposed planning framework enables the robot to travel from a starting configuration $\mathbf{q}_{\text{start}}$ to a goal region $\mathcal{Q}_{\text{goal}} = \{\mathbf{q} \in \mathcal{Q} \,|\, \|\mathbf{q}_{\text{torso}} - \mathbf{q}_{\text{goal}}\|_2 < \epsilon_{\text{torso}}\}$ where ϵ_{torso} is a user-defined distance threshold. In this work, the environment is entirely unknown a priori and partially observable. Therefore, the robot must use its onboard sensing to reveal the environment online. A pictorial representation of the planning framework is provided in Figure 10.

B. Torso Search

First, QuadPiPS performs a coarse initial search over the torso pose $\mathbf{q}_{\text{torso}}$ to construct a guiding torso path through the local environment. The graph itself will be represented as $\mathcal{G}_{\text{torso}} = (\mathcal{N}_{\text{torso}}, \mathcal{E}_{\text{torso}})$. For each planar region $\varphi \in \mathcal{P}$, a set of nodes $\{\eta_{\varphi,0}, \ldots \eta_{\varphi,n_{\text{offset}}}\}$ are created at the region pose $\mathbf{q}_\varphi$ plus an offset. The pose of each node is offset in the direction of the region's normal vector by a nominal torso height $h^{\text{nom}}_{\text{torso}}$ and in the region yaw by $\Delta\gamma_{\text{torso}}$. An edge is added between nodes with a Euclidean distance under $d^{\max}_{\text{torso}}$ and a yaw difference under $\Delta\gamma^{\max}_{\text{torso}}$.

The node closest to the global goal is marked as the goal node. The A* search algorithm [85] is employed with Euclidean distance plus yaw difference as the edge weight and cost-to-go heuristic. The sequence of torso nodes is passed onto the foothold search as a guiding torso path $\mathcal{T}_{\text{torso}} = \{\mathbf{q}_{\text{torso},0}, \ldots, \mathbf{q}_{\text{torso},n_{\text{torso}}}\}$ where the furthest pose $\mathbf{q}_{\text{torso},n_{\text{torso}}}$ is taken as the local waypoint $\mathbf{q}_{\text{LG}}$ for footstep planning.

C. Foothold Search

QuadPiPS adapts its core hierarchical philosophy from the Augmented Leafs with Experience on Foliations (ALEF) framework [22], [23]. To perform perception-informed and kinodynamically-feasible quadrupedal motion planning in real time, QuadPiPS decomposes the task of foothold planning into its discrete and continuous planning spaces. The discrete space consists of the set of potential footholds, and the continuous space consists of the whole body configurations that the robot can assume to perform said footholds. In this section, the discrete space is discussed.

Foothold planning is treated as a search problem over this lower-dimensional discrete planning space. This search employs the graph $\mathcal{G}_{\text{foot}} = (\mathcal{N}_{\text{foot}}, \mathcal{E}_{\text{foot}})$ in which each node $\eta_{\text{foot}} \in \mathcal{N}_{\text{foot}}$ represents a contact mode family. A contact mode family is a partial stance in which a proper subset of the quadruped's feet are in contact with a unique combination of planar regions in the environment. For a single contact mode family, each stance foot must be in contact with a different region. Each edge $\xi_{\text{foot}} \in \mathcal{E}_{\text{foot}}$ represents a transition between two partial stances, meaning that the edge itself is a full stance in which all feet are in contact.

1) Contact Mode Family Transition Graph Construction: To initialize the contact mode family transition graph, a starting mode family η_{start} is constructed from the current robot state and inserted into a priority queue. This proposed framework employs lazy graph construction: during the search, the highest priority node η_{high} is popped from the queue and its neighboring nodes and their respective edges are only added to the graph once η_{high} is expanded. Nodes and edges are added according to set of constraints detailed below.

Contact Sequence: First, a contact sequence constraint is applied. Whichever contact phase the current node is in, the

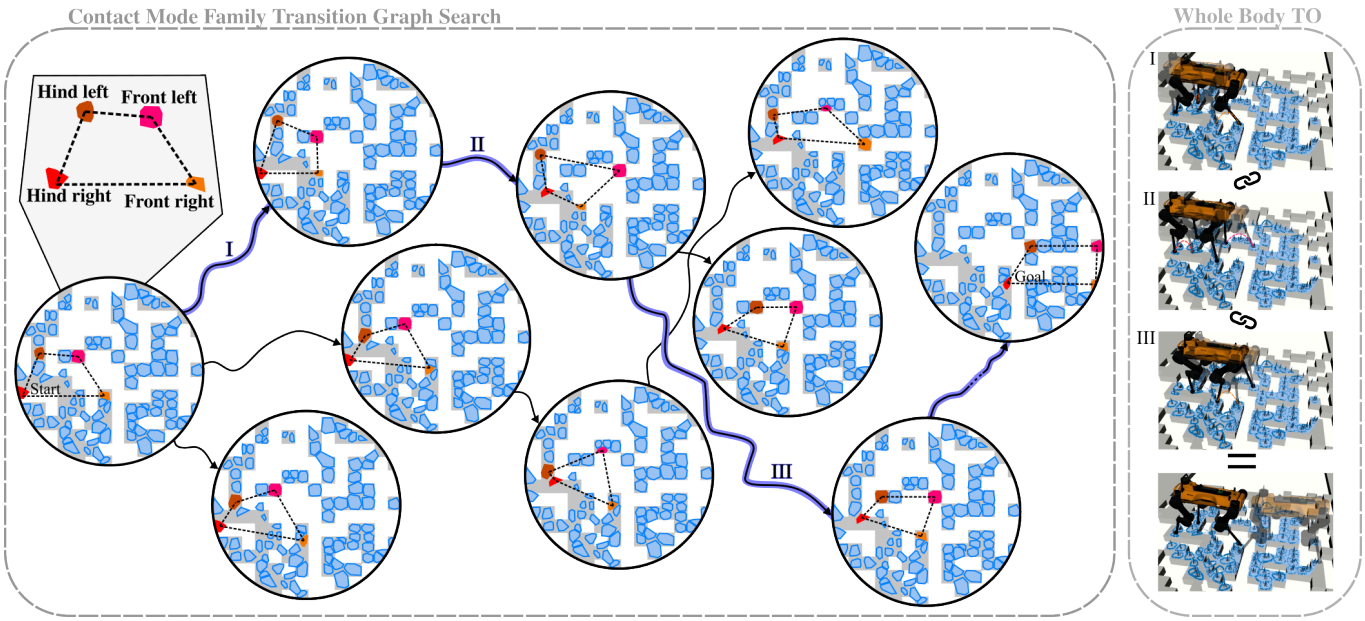

Fig. 10: Overall diagram of the proposed planning framework. A graph search (Section VI-C) is performed over contact mode family transitions defined through a series of geometric and kinematic constraints including a user-defined gait, kinematic reachability volumes, and stance stability. Then, the suggested contact sequence is passed to a long-horizon trajectory optimization (Section VI-D) problem to synthesize the whole body reference trajectory.

candidate nodes must satisfy the following phase of a user-defined gait. While this constraint is not strictly necessary and relaxing it would enable acyclic planning, it does help to reduce the number of potential transitions.

Reachability: Once the next set of stance feet has been decided, a set of reachable regions $\{\mathcal{R}_{\text{FL}}, \mathcal{R}_{\text{FR}}, \mathcal{R}_{\text{HL}}, \mathcal{R}_{\text{HR}}\}$ are compared against the current set of perceived planar regions $\mathcal{P}$ to determine which reachable planar regions $\mathcal{P}_{\text{reach}} = \{\mathcal{P}_{\text{FL}}, \mathcal{P}_{\text{FR}}, \mathcal{P}_{\text{HL}}, \mathcal{P}_{\text{HR}}\}$ can be transitioned to for the current swing feet. The reachable regions are defined as superquadrics [98], an expressive family of 3D volumes. The set of points within a superquadric centered at (x_0, y_0, z_0) are defined as

$$\mathcal{R} = \left\{(x,y,z) \in \mathbb{R}^3 \,\middle|\, \left|\frac{x-x_0}{a}\right|^d + \left|\frac{y-y_0}{b}\right|^e + \left|\frac{z-z_0}{c}\right|^f \leq 1\right\}, \tag{15}$$

where scalars a, b, c and d, e, f define the dimensions and curvature of the regions, respectively. The reachable volumes for the Go2 and ANYmal are visualized in Figure 11.

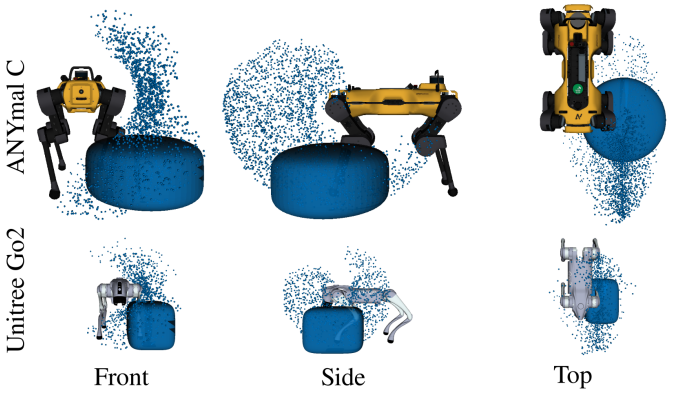

Fig. 11: Visualizations of reachable volumes for ANYmal C (top row) and Unitree Go2 (bottom row). Only the reachable volumes for the front left foot are visualized here. Four volumes in total are generated, one for each foot.

For each planar region $\varphi \in \mathcal{P}$, the pose $\mathbf{q}_\varphi$ is evaluated in Equation 15 to determine if it can be reached for a given foot. Then, this graph expansion step iterates through $\mathcal{P}_{\text{reach}}$. Each combination of reachable regions defines a candidate node η_{cand} that must be evaluated for addition to the graph.

Stance stability: The stance formed by the edge between the current node η_{curr} and the candidate node η_{cand} is checked for stability. A nominal torso pose $\mathbf{q}_{\text{nom}} = \begin{bmatrix}\boldsymbol{\theta}_{\text{nom}} & \mathbf{p}_{\text{nom}}\end{bmatrix}^T \in \mathbb{R}^6$ and footholds $\{\mathbf{p}_{\text{FL}}, \mathbf{p}_{\text{FR}}, \mathbf{p}_{\text{HL}}, \mathbf{p}_{\text{HR}}\}$ are estimated from this candidate edge, and the widths $w_{\text{front}}, w_{\text{hind}}$ and lengths $l_{\text{left}}, l_{\text{right}}$ of the stance feet positions are evaluated against a set of ideal stance dimensions taken at stand up. The stance positions are calculated as

$$\begin{aligned} w_{\text{front}} &= \|R^T(\boldsymbol{\theta}_{\text{nom}}) \cdot (\mathbf{p}_{\text{FR}} - \mathbf{p}_{\text{FL}}) \cdot \hat{\mathbf{e}}_{y_{\text{torso}}}\|_2 \\ w_{\text{hind}} &= \|R^T(\boldsymbol{\theta}_{\text{nom}}) \cdot (\mathbf{p}_{\text{HR}} - \mathbf{p}_{\text{HL}}) \cdot \hat{\mathbf{e}}_{y_{\text{torso}}}\|_2 \\ l_{\text{left}} &= \|R^T(\boldsymbol{\theta}_{\text{nom}}) \cdot (\mathbf{p}_{\text{FL}} - \mathbf{p}_{\text{HL}}) \cdot \hat{\mathbf{e}}_{x_{\text{torso}}}\|_2 \\ l_{\text{right}} &= \|R^T(\boldsymbol{\theta}_{\text{nom}}) \cdot (\mathbf{p}_{\text{FR}} - \mathbf{p}_{\text{HR}}) \cdot \hat{\mathbf{e}}_{x_{\text{torso}}}\|_2 \end{aligned}, \tag{16}$$

where the function $R(\boldsymbol{\theta}_{\text{nom}})$ provides an SO(3) rotation matrix for the torso frame orientation $\boldsymbol{\theta}_{\text{nom}}$ in the inertial frame and $(\hat{\mathbf{e}}_{x_{\text{torso}}}, \hat{\mathbf{e}}_{y_{\text{torso}}})$ provide the x- and y-dimension basis vectors in the torso frame. Stability is then represented as

$$\begin{aligned} |w_{\text{front}} - w_{\text{ideal}}| < \epsilon_{\text{width}} \text{ and } |w_{\text{hind}} - w_{\text{ideal}}| < \epsilon_{\text{width}} \\ |l_{\text{left}} - l_{\text{ideal}}| < \epsilon_{\text{length}} \text{ and } |l_{\text{right}} - l_{\text{ideal}}| < \epsilon_{\text{length}} \end{aligned}, \tag{17}$$

where $\epsilon_{\text{width}}, \epsilon_{\text{length}}$ are user-defined error thresholds.

Swing distance: Lastly, a constraint is applied to the distance between the liftoff and touchdown positions of the swing feet, $\mathbf{p}_{\text{liftoff}}, \mathbf{p}_{\text{touchdown}} \in \mathbb{R}^3$, for the candidate node.

$$\|\mathbf{p}_{\text{liftoff}} - \mathbf{p}_{\text{touchdown}}\|_2 \leq d^{\max}_{\text{swing}}, \tag{18}$$

where $d^{\max}_{\text{swing}}$ is a user-defined maximum swing distance.

If all constraints are met, the candidate node η_{cand} and edge between η_{curr} and η_{cand} are added if they do not already exist.

2) Contact Mode Family Transition Graph Search: To plan a discrete sequence of footholds, QuadPiPS performs a search over the contact mode family transition graph $\mathcal{G}_{\text{foot}}$. The graph search itself follows a standard A^* [85] implementation.

3) Edge weight: For a transition from η_i to η_{i+1}, the edge $\xi_{i,i+1} = (\eta_i, \eta_{i+1})$ is given the weight $d_{\text{torso}}(\eta_i, \eta_{i+1})$ in which $d_{\text{torso}}(\eta_i, \eta_{i+1})$ computes the Euclidean distance between nominal torso positions for η_i and η_{i+1}.

4) Cost-to-go: Each node η is added to the priority queue according to the search heuristic $g(\eta) = d_{\text{torso}}(\eta, \mathbf{q}_{\text{LG}})$.

This graph search returns a discrete sequence of footholds $\mathcal{T}_{\text{feet}}$ that are passed to a whole-body TO program.

D. Swing Trajectory Optimization

1) Trajectory Optimization Subproblem Formulation: The proposed formulation employs the state $\mathbf{x} = \begin{bmatrix}\mathbf{q}_{\text{torso}} & \mathbf{h} & \mathbf{q}_j\end{bmatrix}^T \in \mathbb{R}^{24}$ where the term $\mathbf{h} = \begin{bmatrix}\mathbf{k} & \mathbf{l}\end{bmatrix}^T \in \mathbb{R}^6$ is the centroidal momentum vector comprised of the angular and linear $\mathbf{k} \in \mathbb{R}^3$, $\mathbf{l} \in \mathbb{R}^3$ momentum, respectively. The input consists of $\mathbf{u} = \begin{bmatrix}\mathbf{f} & \mathbf{v}_j\end{bmatrix}^T \in \mathbb{R}^{24}$ where $\mathbf{f} \in \mathbb{R}^{12}$ are the contact forces and $\mathbf{v}_j \in \mathbb{R}^{12}$ are the joint velocities. The optimization subproblem for synthesizing a swing trajectory between two stance configurations is then:

$$\min_{\mathbf{x},\mathbf{u}} \quad \sum_{k=0}^{n_{\text{knot}}-1} \left(\|\mathbf{x}_k - \mathbf{x}_k^{\text{des}}\|_Q^2 + \|\mathbf{u}_k - \mathbf{u}_k^{\text{des}}\|_R^2 \right) + \|\mathbf{x}_{n_{\text{knot}}} - \mathbf{x}_{n_{\text{knot}}}^{\text{des}}\|_{Q_f}^2$$

s.t.

$$\dot{\mathbf{x}}_k = f_{\text{cent}}(\mathbf{x}_k, \mathbf{u}_k) \tag{19a}$$
$$f_{\text{planar},l}(\mathbf{q}_k, \varphi_l) = \mathbf{0} \qquad \forall l \in \mathcal{C}_i \tag{19b}$$
$$\mathbf{f}_{l,k} \in \mathcal{F}_l(\mu, \mathbf{q}_k) \qquad \forall l \in \mathcal{C}_i \tag{19c}$$
$$\mathbf{f}_{l,k} = \mathbf{0} \qquad \forall l \notin \mathcal{C}_i \tag{19d}$$
$$\mathbf{q}_{j,k} \in \mathcal{Q}_{\text{feas}} \tag{19e}$$
$$\mathbf{v}_{j,k} \in \mathcal{V}_{\text{feas}} \tag{19f}$$

where f_{cent} is the centroidal dynamics model [99]. The set $\mathcal{F}_l(\mu, \mathbf{q}_k)$ represents the friction cone which depends on the friction coefficient μ and robot pose $\mathbf{q}_k$. The planar region constraint function $f_{\text{planar},l}(\mathbf{q}_k, \varphi_l)$ is represented as a set of half-space constraints defined by the boundary of the planar region φ_l for swing foot l. The set $\mathcal{C}_i$ represents the set of stance feet for the contact mode family, or node, η_i. The sets $\mathcal{Q}_{\text{feas}}$ and $\mathcal{V}_{\text{feas}}$ define joint position and velocity limits. The cost matrices Q and R are defined through user-defined hyperparameters, and Q_f is computed from a linear-quadratic regular solution at a nominal state and input.

2) Implementation Considerations: The above TO subproblem is solved with the Optimal Control for Switched Systems (OCS2) library [100]. Each swing trajectory occurs over a time horizon of $t_f = 0.5$ seconds using $n_{\text{knot}} = 50$ knot points. The target state $\mathbf{x}_{n_{\text{knot}}}^{\text{des}}$ is obtained through random sampling and projection using the pseudo-inverse of the Jacobian matrix [22]. Then, cubic splines between the takeoff and touchdown configurations define the flight phase of $\mathbf{x}^{\text{des}}$. The desired input $\mathbf{u}^{\text{des}}$ is set to zero to encourage a minimum-energy solution.

The solutions to the sequence of swing trajectory subproblems are appended to form a long-horizon reference trajectory. This reference trajectory is then run through a refinement step to smoothen any irregularities between subproblem solutions. After refinement, the reference trajectories $\mathbf{x}^{\text{ref}}$ and $\mathbf{u}^{\text{ref}}$ are sent to an MPC module for real-time tracking.

VII. Motion Tracking Control

A. Reference Trajectory Publication Monitoring

The proposed motion planner publishes the reference trajectory for tracking and subsequently monitors the robot state for events that signal a re-plan. One such event is that the robot completes the previously sent reference trajectory. Another such event is that the quadruped begins to tip or becomes unstable, signaled by an abnormally large roll or pitch or abnormally large joint velocities or torques. If so, the quadruped is commanded to abort the prior reference trajectory and maintain stability before re-planning.

B. MPC Trajectory Tracking

The Model Predictive Controller (MPC) follows the same formulation given in Equation 19. The MPC employs a timestep Δt of 15 milliseconds over a time horizon of $t_f = 1.0$ seconds. The solver is also warm-started for each new solution using the prior solution and subsequently run for only one SQP iteration. At this level, the reference trajectory $\mathbf{x}^{\text{ref}}$ encourages the MPC to step on the selected regions from the graph search, but the MPC has the authority to assign swing feet to different regions in order to maintain stability. The MPC also synthesizes new swing trajectories depending on the current foot positions to afford the solver more flexibility to deviate from and return to the reference trajectory when necessary.

C. WBC

The optimal state $\mathbf{x}^*$ and input $\mathbf{u}^*$ from the MPC are passed on to a whole body controller (WBC) to synthesize the joint torque commands $\boldsymbol{\tau}$. This WBC takes the form of a hierarchical quadratic program (QP) adapted from [101].

VIII. Experimental Results

A. Experimental Setup - Software

QuadPiPS is integrated into the `quad_auto` codebase which is an extension of the existing open-sourced `legged_control` [102] codebase. All code for testing and deployment is provided in the project repository[1].

B. Simulation Benchmarking

Simulation benchmarking contextualizes QuadPiPS performance amongst several other methods for footstep planning. All testing is performed on a Dell Precision 3660 with an Intel i9-12900K CPU (single-thread passmark score of $4,336$; multi-thread score of $41,322$) and an NVIDIA T1000 GPU.

[1] https://quadpips.github.io/

1) Baselines: Benchmarking compares five baselines:

Elevation Mapping-based Model Predictive Control (EM-MPC) [40]: This baseline performs elevation mapping [19], [35] to obtain a local 2.5D grid representation that is passed on for filtering, classification, segmentation, and precomputation. Inpainting and median filters are applied to mitigate occlusions and noise. Binary steppability classification is performed to ignore regions of the elevation map that can not support a foothold. Then, planar region segmentation is performed via connected component labelling and contour extraction. Lastly, an SDF is computed over the map for collision detection. The MPC formulation follows the same design as Equation 19 so that the perception pipelines can be compared with the same planning and control.

Elevation mapping-based Reinforcement Learning (EM-RL) [103]: This reinforcement learning (RL) baseline employs the same elevation mapping pipeline, but passes a downsampled map along with a twist command and proprioceptive state information into a multi-layer perceptron which outputs a leg phase offset and residual joint position targets.

Elevation mapping-based Deep Tracking Control (EM-DTC) [104] This baseline employs the same elevation mapping pipeline, but employs both a model-based foothold planner as well as a model-free tracking controller. A terrain-aware trajectory generation scheme [105] is employed to provide desired footholds, base poses, twists, and accelerations, and joint positions. Then, a learned RL policy tracks the desired state and outputs target joint positions.

Superpixels-based Model Predictive Control (S-MPC): This baseline employs the proposed perception pipeline in this work. This oversegmented set of planar regions is passed into the MPC formulation defined in Section VII-B. Footholds are generated through a heuristic calculation and local adjustment based on nearby planar regions. **Quadrupedal footstep Planning in the Perception Space (QuadPiPS):** The entire proposed framework.

2) Environments: Benchmarking spans 10 different environments designed to evaluate distinct terrain attributes: *Ramp*, *Stairs*, *Rubble*, *Pegboard*, *Balance beam*, *Ramped balance beam*, *Ramped stepping stones*, *Sparse stepping stones*, *Obstructed balance beam*, and *Winding balance beam*.

3) Perception Timing: Average timing is recorded across 10 runs in the Ramp simulation environment. A RealSense D435i camera streams depth images at 60 Hz. A module-by-module breakdown on timing is given in Figure 12.

While the steppability segmentation is not able to run in real-time given GPU hardware constraints, the overall update rate of the perception pipeline is not constrained by this bottleneck. When a new segmentation mask is available, that information is included in the egocan update. Otherwise, new points are added with an unknown steppability label and prior points with labels are propagated forward. The perception pipeline can then run at 20 Hz.

The elevation mapping update rate is comparable to the egocan update rate. However, the planar region decomposition pipeline runs at roughly four times the speed of the superpixel oversegmentation. The incoming point cloud used for elevation mapping is also downsampled to reduce the time spent preprocessing. This timing difference is the tradeoff for performing oversegmentation. However, this level of granularity pays off in performance, as will be seen in Section VIII-B4.

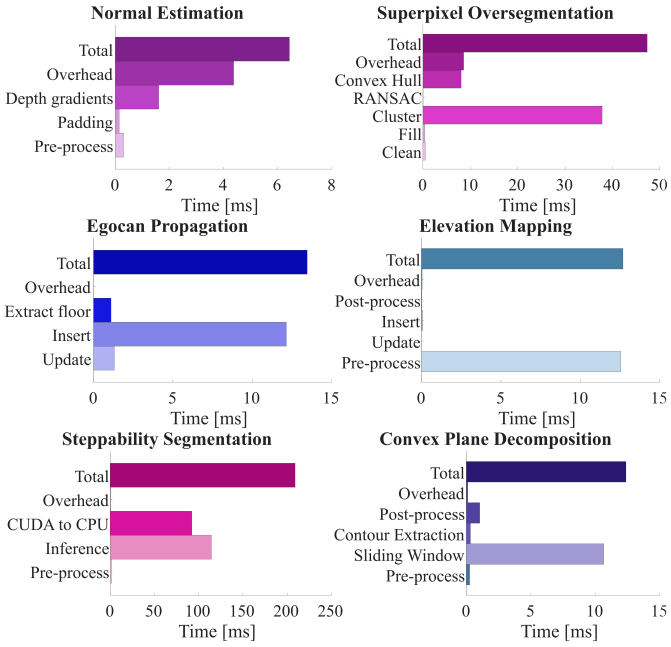

Fig. 12: Average timing for each module in the proposed superpixels-based perception pipeline along with the elevation map-based perception pipeline. The steppability model is run in a separate update thread to prevent the large inference time from acting as a bottleneck on the entire pipeline.

4) Benchmarking Results: Each baseline is run in each environment over 10 trials. For each trial, the robot is assigned the same start and goal position. All baselines outside of QuadPiPS take a twist command pointing from the current robot position to the goal position as input. For QuadPiPS, the global goal position is provided to the foothold planner. All trials are run on the ANYmal C platform due to the limited availability of learned baselines across quadrupedal platforms; pre-trained models for the EM-RL and EM-DTC were only available for the ANYmal C platform, and the proposed work is comprised of model-based planners that can be adapted to any quadrupedal platform with minimal tuning. For simulation trials, ground truth odometry is provided to maximize performance and restrict independent variables to perception, planning, and control. If the quadruped falls over or steps off of the evaluated terrain elements, the trial is marked as a failure. Otherwise, if the baseline reaches the goal, the trial is marked as a success. Visualizations of the environments along with sample superpixels are provided in Figure 13 and baseline results are depicted in Figure 14.

Ramp: Only one EM-MPC trial failed on this environment. During this trial, the whole body controller hit the maximum number of recalculations during its QP solution and returned a poorly conditioned set of joint torques. This caused the quadruped to collapse. While somewhat of an outlier, this trial does highlight a key element to the model-based and model-free control tradeoff. Model-based optimization problems have larger variance in solve times due to features such as initial conditions and time-varying constraints. Model-

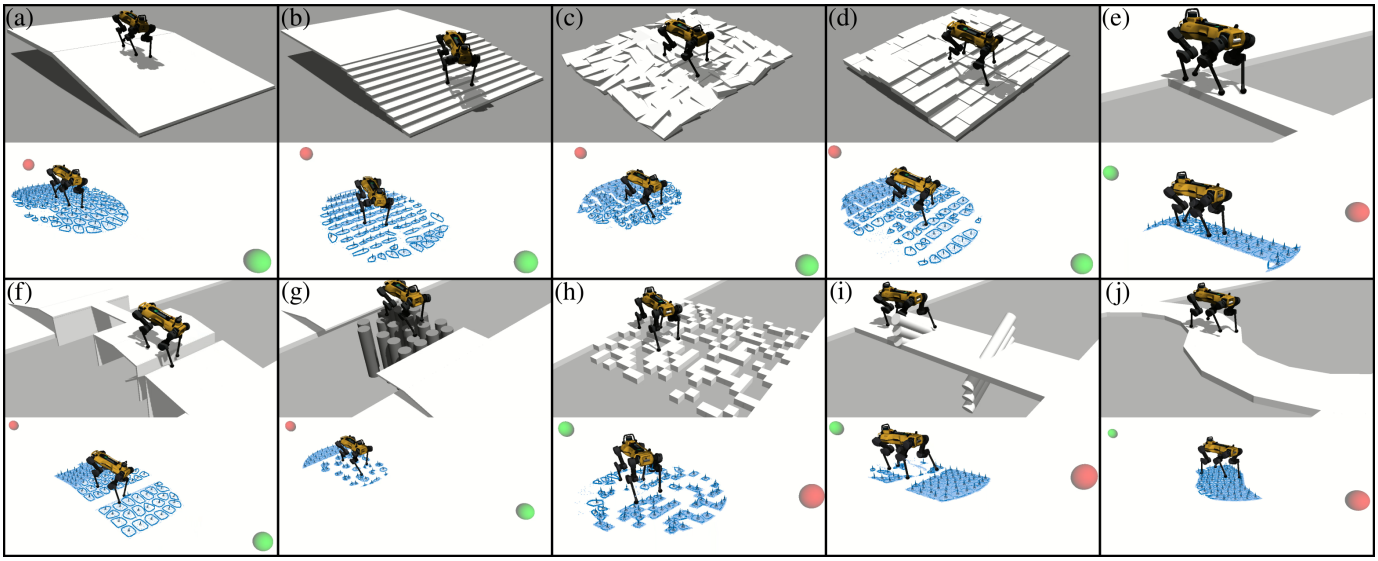

Fig. 13: Visualization of environments with corresponding superpixels. (a) Ramp, (b) Stairs, (c) Rubble, (d) Pegboard, (e) Balance beam, (f) Ramped balance beam, (g) Ramped stepping stones, (h) Sparse stepping stones, (i) Obstructed balance beam, and (j) Winding balance beam. Top images are in Gazebo and bottom images are RViz. In RViz, green spheres represent the start torso pose and red spheres represent the goal torso pose. Superpixels are depicted by blue boundaries, normals, and points.

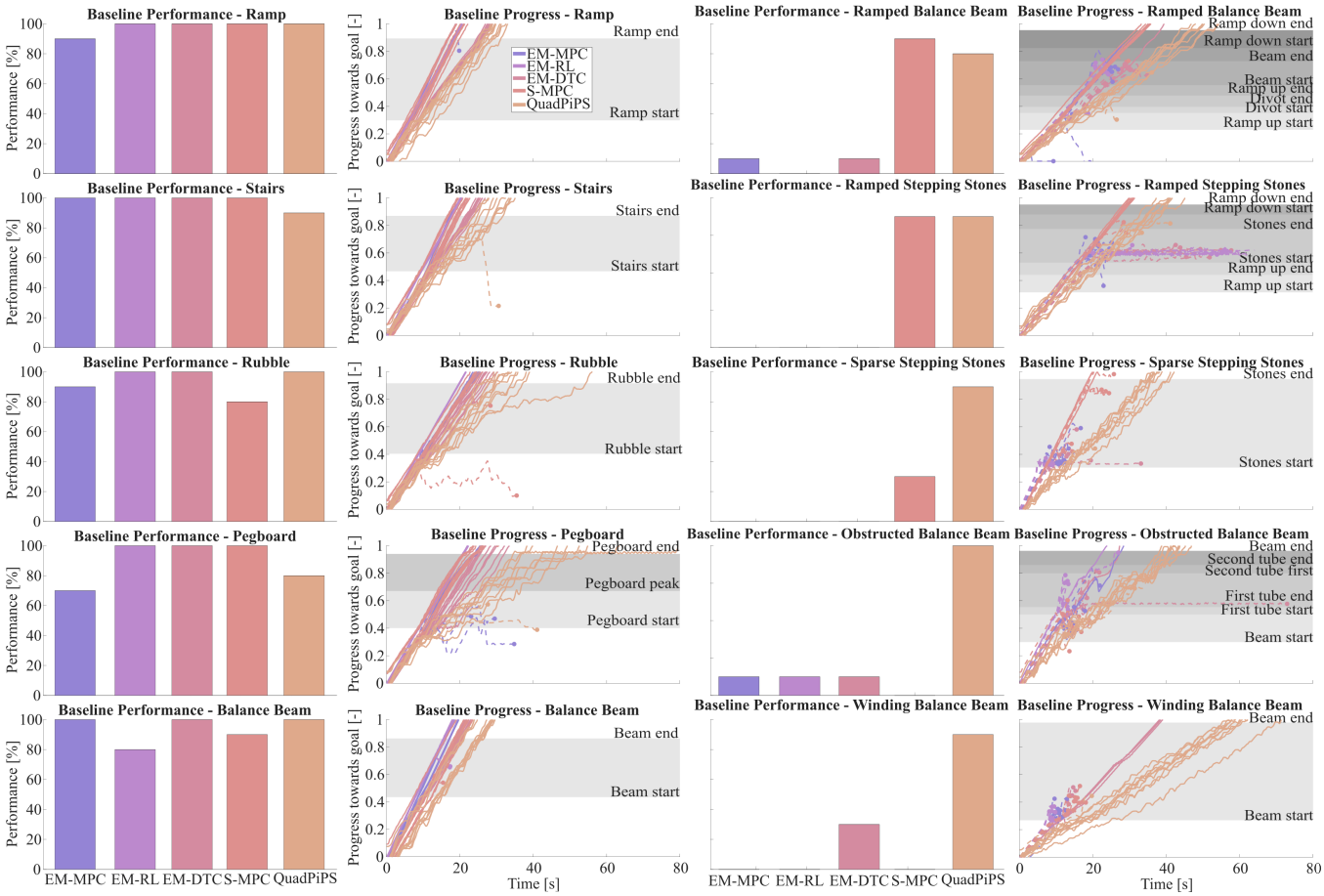

Fig. 14: Simulation benchmarking results for the ten environments. For each environment, the left plot shows the success rates for all baselines. The right plot shows task progress percentage over time for each attempted trial. Solid lines are successes and dashed lines are failures. Environmental feature locations are also displayed to give context on where trials failed.

free controllers represented as neural networks have more consistent inference times, which can be seen in the similar progress rates of the trials for EM-RL and EM-DTC.

Stairs: One QuadPiPS trial failed on this environment. The MPC module received a reference trajectory at a slight

time delay due to latency. Tracking was delayed and the robot lurched forward in attempt to catch up to the reference trajectory, causing the robot to fall. This trial highlights the fragility of model-based planners. Even small perturbations from assumed conditions can cause failures.

The back of the egocan floor image tended to be sparser due to attrition of points during propagation. Planar regions behind the quadruped did span across steps because of this, causing some stumbles that were small enough to recover from.

Rubble: In this setting the EM-MPC baseline sustained one failure whereas the S-MPC baseline sustained two failures. These failures arose from unexpected contact with the environment. In the case of EM-MPC, this was due to the smoothed elevation map reporting incorrect terrain heights. The elevation map segmented the entire local representation into a single planar region during most runs. In the case of S-MPC, these arose from inaccurate swing trajectory tracking causing contact with the side of a stone as opposed to the top. This is another example of the type of small perturbation from expected conditions, including assumed contact sequences, that model-predictive controllers can struggle to reject.

Pegboard: Three failures are observed for the EM-MPC baseline and two are observed for QuadPiPS. EM-MPC failures stemmed from the large step up required to initially mount the pegboard. As can be seen from Figure 14, the failures are clustered towards the start. For QuadPiPS, observed failures were largely due to instability during tracking. Sequencing multiple TO subproblems together limits the momentum of the system between steps. For environments such as this, maintaining forward momentum is crucial to stability.

The learned controllers were the smoothest over this setting. An occasional foothold would get stuck in the negative steps on the pegboard, but the learned controllers were able to lift the foot out of these and continue walking.

Balance beam: In this setting, the EM-RL baseline exhibited two failures and the S-MPC baseline exhibited one. The EM-RL stepped off of the balance beam and fell. This can largely be attributed to the lack of explicit foothold planning. While crossing the beam, the EM-RL baseline would also crouch down and place its feet very close to each other. For the S-MPC baseline, the quadruped approached the beam at an offset and the MPC struggled to transition all four feet onto the beam, leading to a slip. In some trials, QuadPiPS executed lateral steps to align the robot with the beam before stepping onto the beam which proved to be useful. The EM-DTC exhibited the impressive ability to recover from one foot stepping off the beam on a handful of trials.

Ramped balance beam: In this setting, the EM-RL baseline reported no successful trials whereas the EM-MPC and EM-DTC trials each report one successful trial. The S-MPC baseline reported nine successes whereas the QuadPiPS baseline reported eight successes. For the elevation mapping baselines, most failures were concentrated on the balance beam. The two failure modes were mounting the balance beam at an offset and failing to transition all four feet onto the beam, or getting onto the beam but losing stability and tipping over. A select few trials failed on the ramp up to the beam. Some baselines stepped in the divot in the ramp and buckled, leading to a fall.

The superpixels-based perception pipeline presented a significant benefit in this environment. The region oversegmentation provided a natural decomposition over the balance beam that facilitated S-MPC and QuadPiPS with adjusting their desired footholds inwards onto the beam. Figure 13(f) shows that superpixels were not generated for the divot.

Ramped stepping stones: All elevation mapping baselines reported zero successful trials in this environment. Moreover, these baselines made little to no progress over the stepping stones themselves. This came largely from the elevation map representation. The continuous nature of the elevation map led to the height information of smaller stones being merged together, giving the false impression of one large surface that can support footholds. This occurred even when infilling and smoothing filters were removed. With this terrain representation, all baselines planned footholds over unsafe terrain and fell off the stepping stones. The superpixels representation generated clean, disparate regions which enabled stable foothold planning across the terrain. Subsequently, the superpixels baselines were the only planners capable of making progress across the stepping stones, with S-MPC and QuadPiPS both recording nine successful trials. Failures for these baselines arose from inaccurate foothold tracking which caused the robot to slip off the desired stepping stones.

Sparse stepping stones: In this setting, the elevation mapping baselines all reported no successful trials. The S-MPC baseline reported three successful trials, and the QuadPiPS baseline reported eight. The elevation map representation exhibited similar issues to what was discussed for the Ramped Stepping Stones environment in which nearby stepping stones were merged into incorrectly large planar regions. The S-MPC was able to complete some trials, but the nominal foothold positions generated by the MPC constrained the planner to suboptimal locations. All failures observed for S-MPC were from slipping off of stones due to an inability to accurately track the heuristically generated footholds. The search-based foothold planning in the QuadPiPS baseline proved to be enormously beneficial in finding stable stance configurations in this environment. QuadPiPS took longer to find a path to the goal, and these paths were often more circuitous compared to other baselines. However, for an environment such as this where terrain is sparse and precise foothold planning is critical, this tradeoff is heavily preferred.

Obstructed balance beam: In this environment, each elevation map baseline reported one successful trial. These baselines walked directly into the barriers and, by chance, were able to stay upright while circumventing them. For the failed trials, these baselines would often attempt to step onto the barriers which would push the robot towards the edge of the beam, leading to the robot falling off the beam.

For this environment, the legged affordances provided by the steppability model proved essential. Geometric information of depth and surface normals permits planning a foothold on the angled barriers, but this could easily tip the robot over. Therefore, having the steppability model to mark this terrain as passable ensured that the QuadPiPS framework did not step on the barriers. While the S-MPC also does this, the heuristics used to generate nominal footholds still place the footholds in

the center of the beam. The MPC then attempts more difficult swing trajectories over taller parts of the barriers.

Winding balance beam: For this environment, the EM-DTC baseline recorded three successful trials, and the Quad-PiPS framework recorded nine. MPC modules do not have much authority or flexibility to deviate significantly from their desired trajectory. If this desired trajectory is infeasible or poorly conditioned, this can generate internal conflict within the solver. For this environment, the reference trajectory encourages the quadruped to keep its torso along the line between the start and the goal. However, this does not align with the available local terrain. This mismatch caused the baselines outside of QuadPiPS to step off of the beam. For QuadPiPS, all levels of planning are informed by the local terrain. Therefore, the guiding torso path encourages the quadruped to follow the curvature of the beam.

C. Simulation Demonstrations

Demonstrations are also provided for a parallel beams environment, as seen in Figure 15, to showcase the ability of QuadPiPS to find challenging stance configurations.

Parallel beams: This environment possesses two beams that are each a width of 0.2 m and are separated by a distance of 0.85 m, roughly the length of the ANYmal. Here, the ANYmal must cross by side-stepping across the beams. For this environment, additional depth cameras are attached to the left and right sides of the robot. This is only done because the front-facing camera can not perceive the environment as the quadruped is side-stepping across the beams.

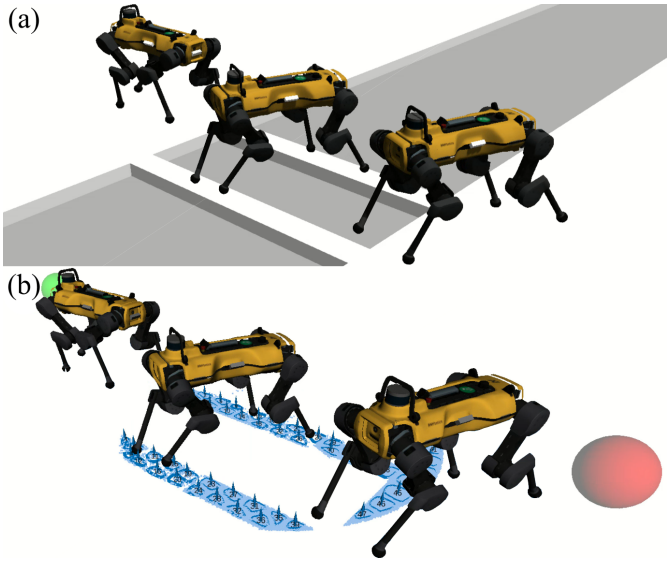

Fig. 15: Visualization of the QuadPiPS framework traversing parallel beams. (a) Gazebo depiction, and (b) RViz visualization. In RViz, the green sphere represents the start torso pose and the red sphere represents the goal torso pose. Superpixels are depicted by blue boundaries, normals. and points.

The distance between the parallel beams is equal to the ideal stance length l_{ideal} in the stance stability constraint in the contact mode family transition graph construction. Therefore, when searching over the beams for candidate stances, the graph search is encouraged to face sideways and strafe across the beams instead of facing forward and widening its stance to step along the beams. Heuristic footstep planners simply walk forward across the beams, but the stance required to bridge the two beams is too wide for the robot. In this setting, large changes to the torso orientation are necessary which heuristic footstep planners or MPC modules struggle to perform unless they are explicitly instructed to do so.

D. Experimental Setup - Hardware

Hardware experiments are run with the Contact-Aided Kalman Filter [106] along with a RealSense T265 Localization Camera. The torso orientation θ_{torso} is observed directly from the onboard IMU. Position estimates are fused with x_{torso} and y_{torso} from the T265 and z_{torso} from the Kalman filter.

1) Networking: There are two PCs onboard the quadruped that collectively run the autonomy stack. First, an NVIDIA Jetson Orin NX with an ARM Cortex-A78AE CPU (single-thread passmark score of 938; multi-thread passmark score of $3,788$) runs the steppability module. Depth images are streamed in from a RealSense D435i camera featuring depth images with 640×480 dimensions at 30 Hz on hardware. Second, a Minisforum UM890 Pro Mini PC with an AMD Ryzen 9 8945HS CPU (single-thread passmark score of $3,822$; multi-thread passmark score of $29,446$) runs the remaining modules. A third low-level PC on the quadruped handles internal communication, sensor readings, and API requests. This machine sends out joint state readings and receives joint torque commands over ROS2 [107] + DDS [108]. Lastly, the user performs remote visualization and monitoring and sends high-level waypoint commands on a Lenovo ThinkPad X1 Carbon laptop with an Intel Core Ultra 7 155U CPU (single-thread passmark score of $3,315$); multi-thread passmark score of $16,160$) over ROS2+DDS as well. Communication between the Jetson, MiniPC, and the internal PC is done over ethernet for minimum latency. Communication between the MiniPC and the laptop is done over WiFi on a private network to ensure real-time performance.

2) Computational Suite: A custom fabricated onboard mount carries all of the previously mentioned computing devices along with the necessary sensors onboard the Go2. This setup is visualized in Figure 16.

(a) Leash
(b) Mini PC
(c) Mini PC cage
(d) Battery
(e) Riser
(f) Rail Plate
(g) Jetson Orin NX
(h) T265 Tracking Camera
(i) D435 Depth Camera

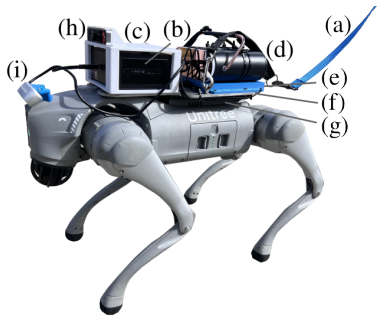

Fig. 16: Custom mounting setup onboard the Unitree Go2 quadruped.

E. Hardware Demonstrations

The proposed work is deployed over real-world terrain setups. Visualizations are provided in Figure 17 and full trials

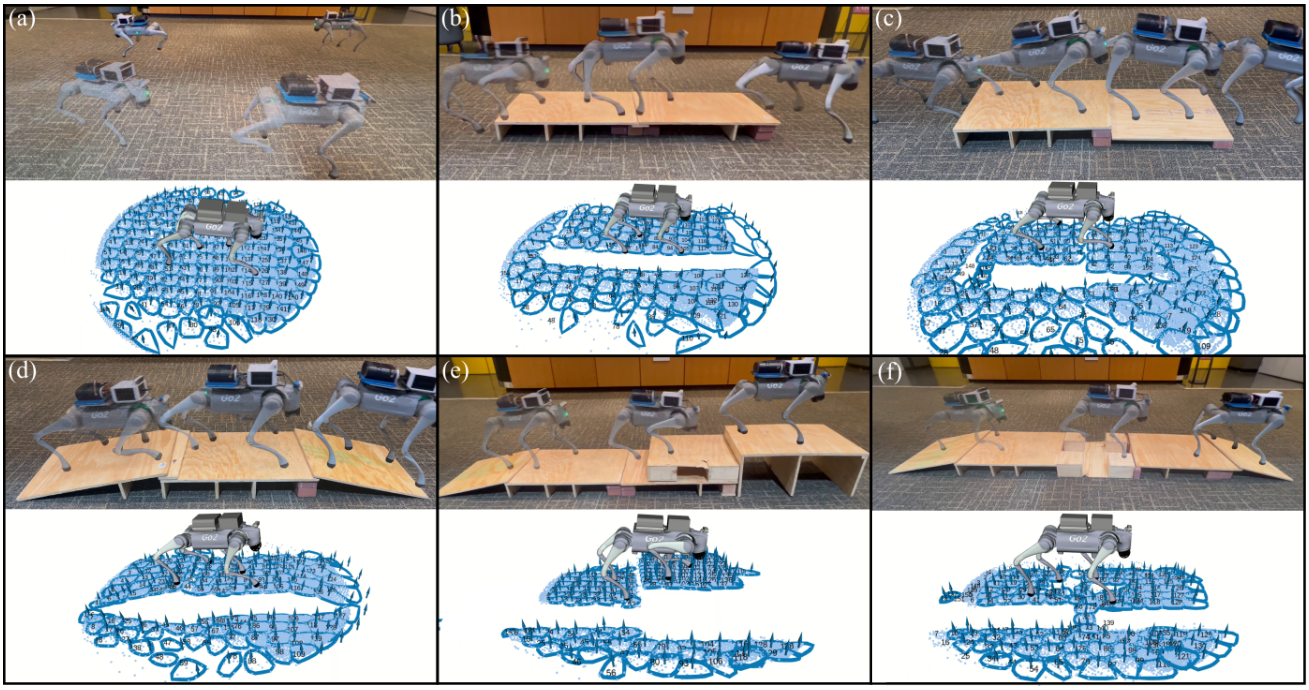

Fig. 17: Visualizations of QuadPiPS hardware deployment in both indoor and outdoor settings. (a) Flat, (b) Single Platform, (c) Double platform, (d) Ramped single platform, (e) Triple platform, (f) Ramped stepping stones platform. Top images are the real world and bottom images are RViz. In RViz, superpixel regions are depicted by blue points, normals, and boundaries.

are shown in the accompanying video[2]. A human operator is present alongside the robot during demonstrations, but they are not providing any support for the robot during deployment.

1) Perception Timing: First, timing data is taken on hardware and reported in this section as well as in Figure 18. For the superpixels-based perception pipeline, timing on hardware is comparable to timing in simulation. The oversegmentation process runs at 20 Hz. For the elevation mapping-based perception pipeline, the mapping process itself is remarkably fast, taking under one millisecond. However, the planar region decomposition process scales more aggressively, limiting the overall pipeline to 25 Hz. Overall, the superpixels pipeline scales nicely onto hardware given its image-based framework.

2) Deployment Results: QuadPiPS plans over five different terrain configurations as well as one flat ground setting.

Flat: Here, QuadPiPS plans a circular trajectory over flat ground to demonstrate baseline performance. During turns, the visibility of the egocan floor is reduced due to the limited field of view of the camera. This manifests in partial loss of data for planar regions, as can be observed in Figure 17.

Single platform: Here, QuadPiPS plans over a raised platform that is 12 cm tall. For reference, the quadruped itself has 23 cm of clearance underneath the torso, and its nominal swing height is 10 cm tall. To be able to step onto the platform with all four legs, the quadruped pitches its torso upwards to give its legs more clearance. For stepping down off the platform, a camera pointed directly forwards misses a significant portion of critical terrain information immediately in front of the robot. Therefore, the camera was pitched 45° downwards in order to capture this crucial data. The quadruped stumbles after stepping down off of the platform, but the MPC is able to stabilize the robot and continuing walking.

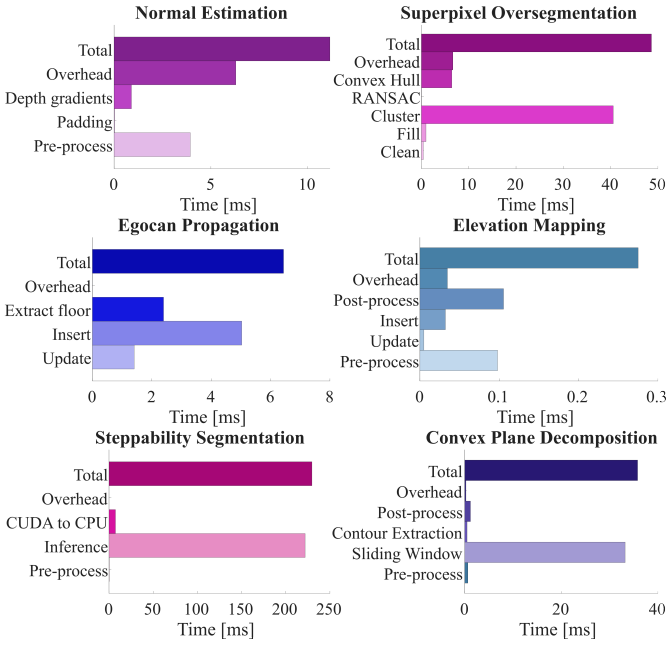

Fig. 18: Average timing for each module in the proposed superpixels-based perception pipeline along with the elevation map-based perception pipeline. The steppability model is run in a separate update thread to prevent the large inference time from acting as a bottleneck on the entire pipeline.

Double platform: Here, QuadPiPS plans over one raised platform that is 12 cm in height and second platform that is 5 cm in height. As in the prior configuration, the planner performs torso pitch adjustment as it steps onto the terrain.

[2]https://quadpips.github.io/

This adjustment is slightly delayed, partially causing tracking mistakes on later steps. For instance, the front left leg plans a foothold on the edge of the first platform. However, during tracking, this foot slips off of the first platform and onto the second. This kind of perturbation is not captured in the QuadPiPS framework. The planned contact sequence is assumed to be followed exactly, so these missteps can lead to tracking failures. In this case, the perturbation was small enough for the MPC to ultimately recover. Although, these common tracking mishaps do motivate the use of more robust tracking controllers such as learned policies.

Ramped single platform: In this trial, QuadPiPS plans over a ramped version of the previously used platforms, here at a 10° incline and decline. While traversing up the ramp, the terrain boards shift and slide around underneath the weight of the quadruped. However, the superpixels-based perception pipeline reacts to these environmental changes and the low-level control rejects these disturbances. The terrain configuration itself is also not necessarily neat. Given the real-time perception, the terrain does not have to be conveniently organized. The board placed underneath the inclined ramp is partially exposed and the quadruped steps on this part of the board. This is all modeled by the perception and subsequently handled by the planning framework. The MPC is delayed in adjusting the robot's height as well as its pitch. This delay yields a low torso height as the quadruped ascends the terrain and a high torso height as it descends the terrain.

Triple platform: For this trial, QuadPiPS plans up one ramped platform with the previously states dimensions, a second platform that is 13 cm tall (for a total height of 25 cm), and a third platform that is 10 cm tall (for a total height of 35 cm). The entire terrain configuration is as tall as the robot itself. While ascending the stairs, QuadPiPS plans steps up against the lip of each platform. The terrain-aware swing trajectory augmentation active within OCS2 is used in both the reference trajectory generation and tracking to lift the swing feet over these sudden height increases. Though, performing additional inflation to ensure steps are not planned on the border of planar regions might be useful in environments such as these. As mentioned before, the terrain boards used in this configuration are chipped and warped, leading to significant deflection and displacement when stepped on. Despite this, the low-level control robustly tracks the footholds up the stairs.

Ramped stepping stones platform: Lastly, QuadPiPS plans over a ramped platform with the previously stated dimensions which follows into a set of stepping stones. Each stepping stone is 20×20 cm. The stones are separated center-to-center by 40 cm. All four feet land on both the back and front sets of stones except for the hind right leg which swings over the back stone. This is an environment in which precise foothold planning is crucial. While nothing prevents QuadPiPS from planning footholds on the ground-level terrain around the stepping stones, quickly stepping down onto the ground and back up onto the platform after the stepping stones poses a challenge for stability. QuadPiPS opts to simply plan over the stepping stones to maintain a consistent torso height over the entire terrain configuration. While descending the ramp after the stones, the front right foot slips at the junction between the platform and the ramp. However, this is a small enough perturbation for the tracking control to reject.

IX. Discussion

This work marks the first investigation into how the Planning in the Perception Space (PiPS) philosophy can be applied to quadrupedal foothold planning. The computational benefits from PiPS such as minimal sensor preprocessing and dense local depth information align well with the task of foothold planning. Foothold planning should only be done in close proximity to the ego-robot, so the task itself does not concern the sparser data further away from the robot. For similar reasons, minimizing latency for foothold planning is crucial.

Simulation environments (a)-(e) represent settings where precise foothold planning is not strictly necessary. If a desired foothold was not accurately tracked, there is neighboring terrain where the robot can step. The last five environment (f)-(j) all require more care for foothold placement, and this is exactly where the superpixels baselines outperform the elevation mapping baselines. Beyond this, QuadPiPS outperformed S-MPC in the last three environments with each environment highlighting a core benefit of QuadPiPS. For sparse stepping stones, heuristic foothold placement is insufficient. When available footholds are limited and do not subscribe to a particular pattern in the case of randomly placed terrain, a more exhaustive method such as searching or sampling is much more beneficial. For the obstructed balance beam, semantic steppability labels encode preferences for movement opportunities. This prevents QuadPiPS from planning footholds on the barriers which would disrupt the stability of the system. Lastly, the winding balance beam exemplifies how QuadPiPS excels through its telescopic framework. A guiding torso path informed by the local environment provides a waypoint for the foothold search, but the foothold sequence is not required to track the torso path. This is useful in environments where the torso path might lead the robot to portions of the environment where foothold planning is more difficult. This freedom can be viewed as robustness to incorrect or suboptimal commands.

It should also be said that superpixels are not necessary for all environments. Across the first five environments, EM-DTC was the only baseline to succeed on every trial. For environments such as the rubble or pegboard, precise foothold planning is not necessary. If the robot can scramble across the terrain and stay upright, it will traverse the terrain successfully. Furthermore, the non-uniform gridding from the over-segmentation can yield asymmetric footholds and inefficient locomotion through unnecessary adjustment from step to step.

With regards to limitations for the QuadPiPS framework, the number of superpixels as well as the poses and boundaries of each superpixel are variable between timesteps. One of the advantages of clustering-based algorithms is that when run over time, they can be warm-started with solutions from prior timesteps. Propagating cluster information over time and warm-starting the superpixels algorithm can mitigate the jitter observed between timesteps and expedite the overall runtime. Superpixels can also hang over edges and overlap with one another on occasion. Further image processing to score the

density or convexity of regions could generate regions that are better suited for footholds.

This work explored how to use the egocan floor for foothold planning, but the egocan ceiling can also inform planning about overhead environment information and allow the quadruped to crouch underneath obstacles. QuadPiPS could then plan through environments such as caves or tunnels.

The current contact mode family transition graph provides stance configurations that are kinematically feasible, but not necessarily dynamically feasible. Additional graph constraints such as stance stability are then used to encourage, but not enforce, dynamic feasibility. Re-introducing the experience heuristic from ALEF would provide further coordination between the search and optimization levels of QuadPiPS and ensure the dynamic feasibility of foothold sequences.

Lastly, the authors found that the robustness of the MPC and WBC restricted the breadth of hardware testing available for QuadPiPS. Replacing the low-level controllers with a learned policy such as DTC that can interface with the superpixels representation would allow for more robust tracking and more challenging terrain traversal including outdoor settings.

X. Conclusion

This work presents QuadPiPS, a perception-informed framework for foothold planning in the perception space. The local environment representation known as the egocan is extended beyond its use for mobile robots to support legged environmental affordances — surface normals and steppability labels — and facilitate foothold searching through a superpixels-based oversegmentation. To construct simulation scenes with accessible ground truth steppability information, QuadPiPS adapts a primitive shapes-based synthetic data generation scheme from the field of grasp prediction. The oversegmented egocan floor is integrated into an extension of the Augmented Leafs with Experience on Foliations (ALEF) framework, which is modified in this work to support (a) perception-informed, (b) real-time, and (c) kinodynamically-feasible motion planning for quadrupeds. Simulation benchmarking across diverse environments and state-of-the-art planners shows that QuadPiPS excels in safety-critical environments where limited footholds are available, and real-world validation on challenging terrain setups indicates that the QuadPiPS framework is readily deployable on legged hardware platforms.

References

[1] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018.

[2] M. Chignoli, D. Kim, E. Stanger-Jones, and S. Kim, "The MIT Humanoid Robot: Design, Motion Planning, and Control For Acrobatic Behaviors," in *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, Jul. 2021.

[3] T. Zhu, M. S. Ahn, and D. Hong, "ARTEMIS: An Open-Source, Full-Sized Humanoid Robot for Dynamic Locomotion."

[4] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Bloesch, H. Kolvenbach, M. Bjelonic, L. Isler, and K. Meyer, "ANYmal - toward legged robots for harsh environments," *Advanced Robotics*, Sep. 2017.

[5] Boston Dynamics, "Atlas Humanoid Robot." [Online]. Available: https://bostondynamics.com/products/atlas/

[6] ——, "Spot." [Online]. Available: https://bostondynamics.com/products/spot/

[7] Unitree Robotics, "Humanoid robot G1." [Online]. Available: https://www.unitree.com/g1

[8] F. Robotics, D. Aldarondo, A. Pervan, D. Corbalan, D. Petrillo, B. Dai, A. Iyer, N. Mortensen, E. Pearson, S. P. Arunachalam, E. Reznick, D. Weis, J. Davison, S. Patterson, T. Carella, M. Suguitan, D. Ye, O. Ferro, N. Suriyarachchi, S. Ling, E. Su, D. Giebisch, P. Traver, S. Fonseca, M. Mor, R. Singh, S. Guven, K. Liu, Y. K. Orru, A. R. A. Batcha, S. Ravindranath, S. Arora, H. Ponte, D. Hernandez, U. Chaudhary, Z. Walker, M. Kelberman, I. Veloz, C. S. Lucia, K. Casale, H. Han, M. Gromis, M. Mignatti, J. Reisman, K. Guerin, D. Narvaez, C. Anderson, A. Moschella, R. Cochran, and J. Merel, "Fauna Sprout: A lightweight, approachable, developer-ready humanoid robot." arXiv, Jan. 2026.

[9] "Hello, Electric Atlas - IEEE Spectrum." [Online]. Available: https://spectrum.ieee.org/atlas-humanoid-robot

[10] DARPA, "Subterranean Challenge Final Event." [Online]. Available: https://www.darpa.mil/research/challenges/subterranean

[11] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, and K. Alexis, "CERBERUS in the DARPA Subterranean Challenge," *Science Robotics*, vol. 7, no. 66, May 2022.

[12] T. H. Chung, V. Orekhov, and A. Maio, "Into the Robotic Depths: Analysis and Insights from the DARPA Subterranean Challenge," *Annual Review of Control, Robotics, and Autonomous Systems*, May 2023.

[13] D. Perille, A. Truong, X. Xiao, and P. Stone, "Benchmarking Metric Ground Navigation," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Abu Dhabi, UAE, Nov. 2020.

[14] A. Nair, F. Jiang, K. Hou, Z. Xu, S. Li, X. Xiao, and P. Stone, "DynaBARN: Benchmarking Metric Ground Navigation in Dynamic Environments," in *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Nov. 2022.

[15] A. Jacoff, J. Jeon, O. Huke, D. Kanoulas, S. Ha, D. Kim, and H. Moon, "Taking the First Step Toward Autonomous Quadruped Robots: The Quadruped Robot Challenge at ICRA 2023 in London [Competitions]," *IEEE Robotics & Automation Magazine*.

[16] J. S. Smith and P. Vela, "AeriaLPiPS: A Local Planner for Aerial Vehicles with Geometric Collision Checking," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023.

[17] ——, "PiPS: Planning in perception space," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 6204–6209.

[18] J. J. Gibson, *The Ecological Approach to Visual Perception*. New York: Psychology Press, Dec. 2014.

[19] T. Miki, L. Wellhausen, R. Grandia, F. Jenelten, T. Homberger, and M. Hutter, "Elevation Mapping for Locomotion and Navigation using GPU," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022.

[20] D. Hoeller, N. Rudin, C. Choy, A. Anandkumar, and M. Hutter, "Neural Scene Representation for Locomotion on Structured Terrain," *IEEE Robotics and Automation Letters*, Oct. 2022.

[21] T. Miki, J. Lee, L. Wellhausen, and M. Hutter, "Learning to walk in confined spaces using 3D representation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024.

[22] Z. Kingston and L. E. Kavraki, "Scaling Multimodal Planning: Using Experience and Informing Discrete Search," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 128–146, Feb. 2023.

[23] M. Asselmeier, J. Ivanova, Z. Zhou, P. A. Vela, and Y. Zhao, "Hierarchical Experience-informed Navigation for Multi-modal Quadrupedal Rebar Grid Traversal," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[24] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[25] Unitree Robotics, "Robot Dog Go2." [Online]. Available: https://www.unitree.com/go2

[26] S. Thrun, "Learning Occupancy Grid Maps with Forward Sensor Models," *Autonomous Robots*, vol. 15, no. 2, pp. 111–127, Sep. 2003.

[27] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "Robot-Centric Elevation Mapping with Uncertainty Estimates," in *Mobile Service Robotics*. Poznan, Poland: WORLD SCIENTIFIC, Aug. 2014, pp. 433–440.

[28] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic Terrain Mapping for Mobile Robots With Uncertain Localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3019–3026, Oct. 2018.

[29] P. Karkowski and M. Bennewitz, "Prediction Maps for Real-Time 3D Footstep Planning in Dynamic Environments," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019.

[30] L. Wellhausen and M. 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)*, Sep. 2021.

[31] S. Behnke, M. Schwarz, T. Rodehutskors, D. Droeschel, M. Schreiber, A. Topelidou-Kyniazopoulou, D. Schwarz, C. Lenz, S. Schuller, J. Razlaw, I. Ivanov, N. Araslanov, and M. Beul, "Team NimbRo Rescue at DARPA Robotics Challenge Finals," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. Seoul, South Korea: IEEE, Nov. 2015, pp. 554–554.

[32] I. D. Miller, F. Cladera, A. Cowley, S. S. Shivakumar, E. S. Lee, L. Jarin-Lipschitz, A. Bhat, N. Rodrigues, A. Zhou, A. Cohen, A. Kulkarni, J. Laney, C. J. Taylor, and V. Kumar, "Mine Tunnel Exploration Using Multiple Quadrupedal Robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2840–2847, Apr. 2020.

[33] P. D. Neuhaus, J. E. Pratt, and M. J. Johnson, "Comprehensive summary of the Institute for Human and Machine Cognition's experience with LittleDog," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 216–235, Feb. 2011.

[34] J. Pippine, D. Hackett, and A. Watson, "An overview of the Defense Advanced Research Projects Agency's Learning Locomotion program," *The International Journal of Robotics Research*, Feb. 2011.

[35] "ANYbotics/elevation_mapping," Aug. 2024, original-date: 2014-05-08T13:26:47Z. [Online]. Available: https://github.com/ANYbotics/elevation_mapping

[36] K. Stepanas, J. Williams, E. Hernández, F. Ruetz, and T. Hines, "OHM: GPU Based Occupancy Map Generation," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 078–11 085, Oct. 2022.

[37] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive Locomotion in Rough Terrain – Online Foothold Optimization," *IEEE Robotics and Automation Letters*, Oct. 2020.

[38] M. Zucker, N. Ratliff, M. Stolle, J. Chestnutt, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "Optimization and learning for rough terrain legged locomotion," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 175–191, Feb. 2011.

[39] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, Aug. 2013.

[40] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive Locomotion through Nonlinear Model Predictive Control," in *arXiv*, Aug. 2022.

[41] J. He, C. Zhang, F. Jenelten, R. Grandia, M. Bächer, and M. Hutter, "Attention-based map encoding for learning generalized legged locomotion," *Science Robotics*, Aug. 2025.

[42] C. Zhang, V. Klemm, F. Yang, and M. Hutter, "AME-2: Agile and Generalized Legged Locomotion via Attention-Based Neural Map Encoding," Mar. 2026.

[43] S. Bertrand, I. Lee, B. Mishra, D. Calvert, J. Pratt, and R. Griffin, "Detecting Usable Planar Regions for Legged Robot Locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, Oct. 2020.

[44] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, "Footstep Planning for Autonomous Walking Over Rough Terrain," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, Oct. 2019, pp. 9–16.

[45] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-Layered Safety for Legged Robots via Control Barrier Functions and Model Predictive Control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Jun. 2021.

[46] B. Acosta and M. Posa, "Bipedal Walking on Constrained Footholds with MPC Footstep Control." arXiv, Sep. 2023.

[47] H. Oleynikova, Z. Taylor, M. Fehr, J. Nieto, and R. Siegwart, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 1366–1373.

[48] A.-a. Agha-mohammadi, E. Heiden, K. Hausman, and G. S. Sukhatme, "Confidence-rich grid mapping," *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1352–1374, Oct. 2019.

[49] J. Frey, D. Hoeller, S. Khattak, and M. Hutter, "Locomotion Policy Guided Traversability Learning using Volumetric Representations of Complex Environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 5722–5729.

[50] T. Hines, K. Stepanas, F. Talbot, I. Sa, J. Lewis, E. Hernandez, N. Kottege, and N. Hudson, "Virtual Surfaces and Attitude Aware Planning and Behaviours for Negative Obstacle Navigation," *IEEE Robotics and Automation Letters*, Apr. 2021.

[51] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, Apr. 2013.

[52] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. P. D'Arpino, R. Deits, M. DiCicco, D. Fourie, T. Koolen, P. Marion, M. Posa, A. Valenzuela, K.-T. Yu, J. Shah, K. Iagnemma, R. Tedrake, and S. Teller, "An Architecture for Online Affordance-based Perception and Whole-body Planning," *Journal of Field Robotics*, 2015.

[53] M. Geisert, T. Yates, A. Orgen, P. Fernbach, and I. Havoutis, "Contact Planning for the ANYmal Quadruped Robot Using an Acyclic Reachability-Based Planner," in *Towards Autonomous Robotic Systems*, K. Althoefer, J. Konstantinova, and K. Zhang, Eds. Springer International Publishing, 2019.

[54] K. Fang, Y. Chen, X. Zhu, F. Niroui, L. Sun, and J. Wang, "SAGA: Open-World Mobile Manipulation via Structured Affordance Grounding." arXiv, Dec. 2025.

[55] Y.-C. Lin and D. Berenson, "Humanoid Navigation Planning in Large Unstructured Environments Using Traversability - Based Segmentation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 7375–7382.

[56] A. Agha, K. Otsu, B. Morrell, D. D. Fan, R. Thakker, A. Santamaria-Navarro, S.-K. Kim, A. Bouman, X. Lei, J. Edlund, M. F. Ginting, K. Ebadi, M. Anderson, T. Pailevanian, E. Terry, M. Wolf, A. Tagliabue, T. S. Vaquero, M. Palieri, S. Tepsuporn, Y. Chang, A. Kalantari, F. Chavez, B. Lopez, N. Funabiki, G. Miles, T. Touma, A. Buscicchio, J. Tordesillas, N. Alatur, J. Nash, W. Walsh, S. Jung, H. Lee, C. Kanellakis, J. Mayo, S. Harper, M. Kaufmann, A. Dixit, G. J. Correa, C. Lee, J. Gao, G. Merewether, J. Maldonado-Contreras, G. Salhotra, M. S. Da Silva, B. Ramtoula, S. Fakoorian, A. Hatteland, T. Kim, T. Bartlett, A. Stephens, L. Kim, C. Bergh, E. Heiden, T. Lew, A. Cauligi, T. Heywood, A. Kramer, H. A. Leopold, H. Melikyan, H. C. Choi, S. Daftry, O. Toupet, I. Wee, A. Thakur, M. Feras, G. Beltrame, G. Nikolakopoulos, D. Shim, L. Carlone, and J. Burdick, "NeBula: TEAM CoSTAR's Robotic Autonomy Solution that Won Phase II of DARPA Subterranean Challenge," *Field Robotics*, Jul. 2022.

[57] N. Kottege, J. Williams, B. Tidd, F. Talbot, R. Steindl, M. Cox, D. Frousheger, T. Hines, A. Pitt, B. Tam, B. Wood, L. Hanson, K. Lo Surdo, T. Molnar, M. Wildie, K. Stepanas, G. Catt, L. Tychsen-Smith, D. Penfold, L. Overs, M. Ramezani, K. Khosoussi, F. Kendoul, G. Wagner, D. Palmer, J. Manderson, C. Medek, M. O'Brien, S. Chen, and R. C. Arkin, "Heterogeneous Robot Teams With Unified Perception and Autonomy: How Team CSIRO Data61 Tied for the Top Score at the DARPA Subterranean Challenge," *IEEE Transactions on Field Robotics*, vol. 2, pp. 100–130, 2025.

[58] Z. Yoon, L. Y. Zhu, J. Lu, L. Gan, and Y. Zhao, "STATE-NAV: Stability-Aware Traversability Estimation for Bipedal Navigation on Rough Terrain," *IEEE Robotics and Automation Letters*, Feb. 2026.

[59] A. Dixit, D. D. Fan, K. Otsu, S. Dey, A.-A. Agha-Mohammadi, and J. W. Burdick, "STEP: Stochastic Traversability Evaluation and Planning for Risk-Aware Navigation; Results From the DARPA Subterranean Challenge," *IEEE Transactions on Field Robotics*, 2025.

[60] H. Biggie, E. R. Rush, D. G. Riley, S. Ahmad, M. T. Ohradzansky, K. Harlow, M. J. Miles, D. Torres, S. McGuire, E. W. Frew, C. Heckman, and J. S. Humbert, "Flexible Supervised Autonomy for Exploration in Subterranean Environments," *Field Robotics*, vol. 3, no. 1, pp. 125–189, Jan. 2023.

[61] L. Wellhausen, A. Dosovitskiy, R. Ranftl, K. Walas, C. Cadena, and M. Hutter, "Where Should I Walk? Predicting Terrain Properties From Images Via Self-Supervised Learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1509–1516, Apr. 2019.

[62] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim, "Vision Aided Dynamic Exploration of Unstructured Terrain with a Small-Scale Quadruped Robot," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020.

[63] R. Buchanan, L. Wellhausen, M. Bjelonic, T. Bandyopadhyay, N. Kottege, and M. Hutter, "Perceptive whole-body planning for multilegged robots in confined spaces," *Journal of Field Robotics*, 2021.

[64] M. Kulkarni, M. Dharmadhikari, M. Tranzatto, S. Zimmermann, V. Reijgwart, P. De Petris, H. Nguyen, N. Khedekar, C. Papachristos, L. Ott, R. Siegwart, M. Hutter, and K. Alexis, "Autonomous Teamed Exploration of Subterranean Environments using Legged and Aerial

Robots," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022.
[65] L. Wellhausen and M. Hutter, "ArtPlanner: Robust Legged Robot Navigation in the Field," *Field Robotics*, 2023.
[66] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning Quadrupedal Locomotion over Challenging Terrain," *Science Robotics*, Oct. 2020.
[67] A. Bouman, M. F. Ginting, N. Alatur, M. Palieri, D. D. Fan, T. Touma, T. Pailevanian, S.-K. Kim, K. Otsu, J. Burdick, and A.-a. Agha-Mohammadi, "Autonomous Spot: Long-Range Autonomous Exploration of Extreme Environments with Legged Locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 2518–2525.
[68] S.-K. Kim, A. Bouman, G. Salhotra, D. D. Fan, K. Otsu, J. Burdick, and A.-a. Agha-mohammadi, "PLGRIM: Hierarchical Value Learning for Large-scale Exploration in Unknown Environments," *Proceedings of the International Conference on Automated Planning and Scheduling*, May 2021.
[69] S. Fahmi, V. Barasuol, D. Esteban, O. Villarreal, and C. Semini, "ViTAL: Vision-Based Terrain-Aware Locomotion for Legged Robots," *IEEE Transactions on Robotics*, Apr. 2023.
[70] O. Villarreal, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini, "Fast and Continuous Foothold Adaptation for Dynamic Locomotion through CNNs," *IEEE Robotics and Automation Letters*, Apr. 2019.
[71] S. Fahmi, C. Mastalli, M. Focchi, and C. Semini, "Passive Whole-Body Control for Quadruped Robots: Experimental Validation Over Challenging Terrain," *IEEE Robotics and Automation Letters*, Jul. 2019.
[72] T. Boaventura, J. Buchli, C. Semini, and D. G. Caldwell, "Model-Based Hydraulic Impedance Control for Dynamic Robots," *IEEE Transactions on Robotics*, Dec. 2015.
[73] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ – a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, Sep. 2011.
[74] T. Dudzik, M. Chignoli, G. Bledt, B. Lim, A. Miller, D. Kim, and S. Kim, "Robust Autonomous Navigation of a Small-Scale Quadruped Robot in Real-World Environments," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020.
[75] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control," in *arXiv*, Sep. 2019.
[76] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "ANYmal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, Mar. 2024.
[77] Y. Chen, J. Ma, Z. Luo, Y. Han, Y. Dong, B. Xu, and P. Lu, "Learning Autonomous and Safe Quadruped Traversal of Complex Terrains Using Multi-Layer Elevation Maps," *IEEE Robotics and Automation Letters*, Oct. 2025.
[78] R. Xu, S. Feng, and P. A. Vela, "Potential Gap: A Gap-Informed Reactive Policy for Safe Hierarchical Navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8325–8332, 2021.
[79] J. S. Smith, S. Feng, F. Lyu, and P. A. Vela, "Real-Time Egocentric Navigation Using 3D Sensing," in *Machine Vision and Navigation*, O. Sergiyenko, W. Flores-Fuentes, and P. Mercorelli, Eds., 2020.
[80] J. S. Smith, R. Xu, and P. Vela, "egoTEB: Egocentric, Perception Space Navigation Using Timed-Elastic-Bands," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2703–2709.
[81] D. Driess, J.-S. Ha, and M. Toussaint, "Deep Visual Reasoning: Learning to Predict Action Sequences for Task and Motion Planning from an Initial Scene Image," in *Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation, Jul. 2020.
[82] S. Feng, Z. Zhou, J. Smith, M. Asselmeier, Y. Zhao, and P. A. Vela, "GPF-BG: A hierarchical vision-based planning framework for safe quadrupedal navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
[83] M. Sorokin, J. Tan, C. K. Liu, and S. Ha, "Learning to Navigate Sidewalks in Outdoor Environments," *IEEE Robotics and Automation Letters*, Apr. 2022.
[84] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged Locomotion in Challenging Terrains using Egocentric Vision," in *Proceedings of The 6th Conference on Robot Learning*, Mar. 2023.
[85] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
[86] L. Matthies, R. Brockers, Y. Kuwata, and S. Weiss, "Stereo vision-based obstacle avoidance for micro air vehicles using disparity space," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3242–3249.
[87] Y. Nakagawa, H. Uchiyama, H. Nagahara, and R.-I. Taniguchi, "Estimating Surface Normals with Depth Image Gradients for Fast and Accurate Registration," in *2015 International Conference on 3D Vision*, Oct. 2015, pp. 640–647.
[88] M. Nieuwenhuisen, J. Stueckler, A. Berner, R. Klein, and S. Behnke, "Shape-Primitive Based Object Recognition and Grasping," in *ROBOTIK 2012; 7th German Conference on Robotics*, May 2012.
[89] Y. Lin, C. Tang, F.-J. Chu, and P. A. Vela, "Using synthetic data and deep networks to recognize primitive shapes for object grasping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 494–10 501.
[90] A. Grunnet-Jepsen and D. Tong, "Depth Post-Processing for Intel® RealSense™ D400 Depth Cameras."
[91] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," in *ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Springer International Publishing, 2018.
[92] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick, "Detectron2," 2019. [Online]. Available: https://github.com/facebookresearch/detectron2
[93] P. Neubert and P. Protzel, "Compact Watershed and Preemptive SLIC: On Improving Trade-offs of Superpixel Segmentation Algorithms," in *2014 22nd International Conference on Pattern Recognition*, Aug. 2014, pp. 996–1001.
[94] D. Weikersdorfer, D. Gossow, and M. Beetz, "Depth-adaptive superpixels," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, Nov. 2012, pp. 2087–2090.
[95] D. Weikersdorfer, A. Schick, and D. Cremers, "Depth-adaptive supervoxels for RGB-D video segmentation," in *2013 IEEE International Conference on Image Processing*, Sep. 2013, pp. 2708–2712.
[96] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, Jun. 1981.
[97] R. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Information Processing Letters*, vol. 1, no. 4, pp. 132–133, Jun. 1972.
[98] O. Melon, R. Orsolino, D. Surovik, M. Geisert, I. Havoutis, and M. Fallon, "Receding-Horizon Perceptive Trajectory Optimization for Dynamic Legged Locomotion with Learned Initialization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 9805–9811.
[99] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, Oct. 2013.
[100] "OCS2: An open source library for Optimal Control of Switched Systems." [Online]. Available: https://github.com/leggedrobotics/ocs2
[101] C. Dario Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov. 2016.
[102] Qiayuan Liao and others, "{legged_control}: NMPC, WBC, state estimation, and sim2real framework for legged robots based on OCS2 and ros-controls," [Online]. Available: \url{https://github.com/qiayuanl/legged_control}.
[103] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, Jan. 2022.
[104] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "DTC: Deep Tracking Control," *Science Robotics*, vol. 9, no. 86, Jan. 2024.
[105] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, "TAMOLS: Terrain-Aware Motion Optimization for Legged Systems," *IEEE Transactions on Robotics*, Dec. 2022.
[106] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended Kalman filtering for robot state estimation," *The International Journal of Robotics Research*, 2020.
[107] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, May 2022.
[108] G. Pardo-Castellote, "OMG Data-Distribution Service: architectural overview," in *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, May 2003, pp. 200–206.