Rough Terrain Navigation for Legged Robots using Reachability Planning and Template Learning

Rough Terrain Navigation for Legged Robots using Reachability Planning and Template Learning

Lorenz Wellhausen and Marco Hutter

***Abstract*— Navigation planning for legged robots has distinct challenges compared to wheeled and tracked systems due to the ability to lift legs off the ground and step over obstacles. While most navigation planners assume a fixed traversability value for a single terrain patch, we overcome this limitation by proposing a reachability-based navigation planner for legged robots. We approximate the robot morphology by a set of reachability and body volumes, assuming that the reachability volumes need to always be in contact with the environment, while the body should be contact-free. We train a convolutional neural network to predict foothold scores which are used to restrict geometries which are considered suitable to step on. Using this representation, we propose a navigation planner based on probabilistic roadmaps. Through validation of only low-cost graph edges during graph expansion and an adaptive sampling scheme based on roadmap node density, we achieve real-time performance with fast update rates even in cluttered and narrow environments. We thoroughly validate the proposed navigation planner in simulation and demonstrate its performance in real-world experiments on the quadruped ANYmal.**

I. Introduction

Navigation planning for legged robots has distinct challenges which are not present for other types of robots. While flying robots attempt to avoid any contact with the environment, ground robots by definition require contact with the ground to locomote. Compared to other types of ground robots, which have a constant contact patch with the ground, legged robots can overcome obstacles by lifting their legs. Most traditional navigation planning approaches assume a single traversability value for any given terrain patch, which they check against the footprint of the robot [1], [2]. These approaches are limiting for legged robots due to their ability to change their footprint and choose contact locations with the environment deliberately. Therefore, we have chosen to apply a different, simplified robot representation when planning for legged systems based on limb reachability abstractions [3]. We represent a robot as one collision volume for its torso, and one reachability volume for each of its limbs. When checking the feasability of a given robot pose, we expect the torso volume to be collision-free, while we enforce collision for the reachability volume, to ensure that the robot is able to make environment-contact with its legs.

While this approach relies purely on geometric terrain information, it also enables the inclusion of semantically derived quantities into the planning process by maintaining separate maps for torso and reachability collision checking. As we have shown in previous work [5], [6], we can learn to predict non-geometric obstacles like slippery or unknown terrain from visual semantic information. We can regard this information as foothold feasibility classification and only allow safe regions to support footholds by removing unsafe terrain from the reachability collision map. We leverage this by training a convolutional neural network (CNN) to predict foothold scores from the planner map and geometrically identify unsteppable regions.

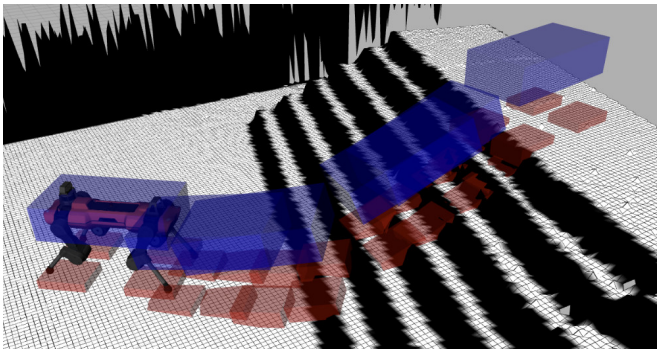


Fig. 1: Path generated by our navigation planner for the quadruped ANYmal [4]. Blue boxes represent torso collision abstractions while red boxes indicate reachability volumes.

Since we are interested in a navigation planner which can work in possibly unknown environments, it will use information from an onboard mapping pipeline which is continuously updated as the robot moves. We therefore require a fast update rate for our planner to keep up with map updates. While frequent drastic map changes would favor a single-query planner, like RRT* [7], we target environments with few dynamic obstacles such that most map changes occur through newly observed terrain while previously observed map regions remain largely unchanged. Therefore, we base our planner on PRM* [7], building a planning graph online, while maintaining a persistent planning graph between planning queries.

PRM* requires a nearest-neighbor lookup when inserting a new sample into the planning graph, which grows in computational demand as the graph gets larger. Uniformly sampling the map for new nodes risks bloating the graph in wide-open "easy" map regions, risking failure to find a solution in more difficult, narrow spaces. We therefore employ a sampling distribution which uses the inverse density of graph nodes as sampling probability to guide graph expansion towards

This work was supported by the Swiss National Science Foundation (SNF) through the NCCR Robotics and project 188596, and the EU Horizon 2020 research and innovation programme under grant agreements No 780883, No 852044 and No 101016970. It has been conducted as part of ANYmal Research, a community to advance legged robotics

[1]Robotic Systems Lab, ETH Zürich, Switzerland

underexplored regions.

The main contribution of this work is a real-time navigation planning framework for legged robots. An open-source implementation of our planning pipeline is available online[1]. We show that our navigation planner is able to generate low-cost feasible paths with fast query times and runs in the loop with locomotion and mapping pipelines on our quadruped ANYmal [4].

The main limitations of our work are two-fold. First, our assumption that reachability collision abstractions must be in contact prevents us from planning highly dynamic maneuvers with long flight-phases, e.g. jumping over a large gap. Second, due to computational constraints we restrict the optimized cost function to shortest-path, without considering foothold safety margins, e.g. the size of the steppable area inside of the foot reachability. While this is partially compensated for by our foothold score learning method, this means that our planner can produce risky paths, especially if the reachability volumes are set too big, which requires a highly capable locomotion controller to successfully track them.

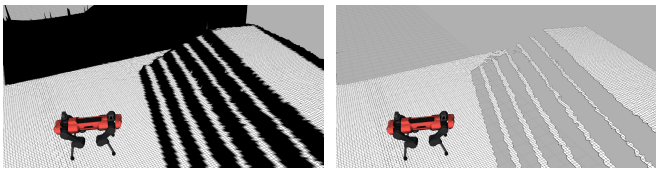
II. Related Work

Navigation planning for mobile robots is a vast field of research with a manifold of different approaches, both in the sensors used to perceive the environment as well as the environment representation used for planning.

Traditional navigation approaches for mobile robots use a geometric environment representation as their basis for planning [1], [2], [8], [9]. They use various different terrain representations for planning, most commonly 2.5D height maps [1], [8], point clouds [2] or truncated signed-distance field (TSDF)s [9]. While only point clouds and TSDFs are full 3D representations of the environment, current methods do not support processing these at resolutions necessary for planning with legged robots, which are highly mobile and therefore require mapping resolutions of a few centimeters. Additionally, locomotion planning and control methods which can leverage full 3D environments are not yet real-time capable [10]. We therefore work with 2.5D height maps as environment representation, which is also used by current real-time capable locomotion controllers [11]. Most planning approaches compute a single geometric traversability value per terrain patch [1], [2] as measure how easily the terrain can be traversed, irrespective of robot orientation, which works well for wheeled and tracked robots, which have a continuously moving contact area while locomoting. Legged robots, however, have much higher mobility and can step over obstacles, which makes the notion of a single traversability value impractical.

While we have argued in previous work [5], [6] that purely geometric approaches are not sufficient for navigation in natural outdoor environments, approaches relying on semantic information exhibit the same issues as traditional geometric approaches. They, either implicitly through semantic segmentation of the environment [12], [13], [14], [15], or explicitly [16], [17], [18] predict a traversability label. However, we can instead reinterpret traversability labels as foothold feasibility labels and use them to enhance geometric planning.

[1] www.github.com/leggedrobotics/art_planner



(a) Torso Collision World (b) Reachability Collision World

Fig. 2: Collision worlds for torso (a) and reachability (b) volumes. The torso collision world (a) considers all geometry perceived by the mapping algorithm. Regions considered unsuitable for footholds are colored in black. This geometry is removed (b) when checking reachability volumes for collision.

While full kino-dynamic planning over long horizons would be the most general and accurate planning method, applying these methods in real-time is not tractable on current computational hardware [10], [19], [20]. Dynamic planning using a reduced robot model has recently shown promising results [21] but has not been evaluated in deployment scenarios. Other work on navigation planning specifically for legged robots either only considers cases of obstacle avoidance on flat terrain [22], [23] or does additional contact planning, which pushes computational complexity past the real-time mark [24], [25], [26].

Learning traversability [8] and motion cost [27] is an interesting approach which can encapsulate locomotion capabilities of legged robots but due to the sequential querying of neural networks during sampling-based planning they are computationally inefficient. Concurrent work on a parallel sampling planner [28] has produced promising results but struggles with complex maneuvers in tight spaces due to simplifications made to achieve faster sampling speeds. Methods which learn a sampling distribution [29] to accelerate sampling-based planners or completely learned planners [30] show promise for fast planning but require an already existing planner to generate training data.

We adopt a reachability-based robot representation [3] which sufficiently approximates the locomotion capabilities of legged robots, while allowing for computationally efficient state validity checking.

III. Method

Our approach is based on a sampling-based planner, using a reachability assumption for state validity checking and a custom sampling scheme to bias graph expansion towards difficult regions. A learned foothold score is used to restrict geometry which is considered suitable to step on.

A. State Validity Checking

To check for validity of sampled robot poses, we use a reachability-based approach [3]. It is based on the notion that the ground support surface needs to be reachable for the robot's legs, while the torso remains collision-free. While this

assumption can be broken by highly dynamic gaits with full flight-phases we have found that in practice it is sufficient to encapsulate the motions produced by current state-of-the-art controllers [31], [11] which have no or very short flight-phases.

To check for this condition, we decompose the robot body into volumes representing torso collisions and leg reachability. Reachability volumes can be obtained by sampling random joint configurations for the robot, computing the end-effector position with forward kinematics and then using alpha-shape [32] computation to obtain an accurate polygon. While this yields the most accurate representation for reachability, most locomotion controllers do not use the full range-of-motion available to them. Therefore, we use a simple box to represent reachability, as done in other work [20], which represents the actual capabilities of current locomotion controllers better and improves collision checking speed significantly. The abstraction used for the ANYmal [4] robot is shown in Figure 1. Body collision models are typically available as part of a robot model or can be abstracted by simple primitive shapes.

A downside of this reachability-based approach is that in its basic form it does not have a notion of dynamic feasibility.

It would consider poses valid which stand vertically on a wall or upside down on the ceiling. While other work exists, which extends the reachability-based model [3] with dynamic feasability [19], this method is not currently real-time capable. To overcome this problem, we segment the environment into two separate collision worlds, shown in Figure 2, one for the torso and one for reachability. The torso collision world contains all geometry as provided by the mapping pipeline. For the reachability collision world we remove geometry which is considered unsuitable for supporting a foothold, by thresholding a learned foothold score, further detailed in Section III-B.

A valid pose is therefore a pose where all reachability volumes are in contact with geometry in the reduced reachability world, while the torso volume is collision free in the full torso collision world. To check validity of transitions between two poses we linearly interpolate between both and check state validity at intermediate states with a fixed resolution.

B. Learning Foothold Scores

Information about valid footholds can come from different sources, like semantic information [5], [6] or geometric analysis of the environment [1]. In this work we use a geometric template-learning approach which predicts foothold scores $\in [0, 1]$ from height maps and can both be computed rapidly and trained with very little data. An overview of our method is shown in Figure 3.

Previous methods define hand-tuned filters to score footholds, like terrain roughness, step height and inclination [1] which are applied to every grid cell of the map and then combined into a score using a weighted sum. Because these filters are position-invariant, applying them can be regarded as convolving templates over the height map. Instead of defining them by hand, we learn both these templates and weights for the weighted sum using a CNN and gradient-based optimization. This allows for fast and intuitive definition of invalid footholds, because they can directly be selected on a few training height maps, instead of iteratively tuning heuristic weights and thresholds.

We use a single convolutional layer with 12 kernels of size 3×3 without activation function or bias, followed by a 1×1 convolution which acts as a weighted sum. The 3×3 convolutions use different dilation sizes $\in [1, 2, 3]$ to increase receptive field size while keeping the number of parameters low. We enforce the weights of the 3×3 convolution filters to be zero-sum to ensure invariance to the absolute terrain height. In the same fashion, we enforce the weights of the weighted sum to be positive. Because of these constraints, uneven terrain which is unsuitable for footholds will generate a strong response and possibly large output values while perfectly flat terrain will have zero response. To obtain an output between zero for invalid footholds and one for good footholds, we use the exponential of the negated weighted sum output as our final foothold score.

As we refrain from using biases in the convolutional layers, this network has only $12 \cdot 3 \cdot 3 + 12 = 120$ learnable parameters, which can be trained rapidly and with very little manually labelled training data. The small number of parameters also makes the network less prone to overfitting to training data. As Figure 3 shows, the model learns generic filter kernels akin to corner and edge detectors, such that it generalizes well.

C. Planner

Algorithm 1: Graph Expansion Method

```
Input: Graph 𝒢
do                                  // Sample valid pose.
    p_3D ← samplePose();
while ¬valid(p_3D);
𝒩_goal ← addPoseToGraph(p_3D, 𝒢);
do                                  // Find start node.
    𝒩_start ← selectRandomNode(𝒢);
while ¬connected(𝒩_start, 𝒩_goal, 𝒢);
do
    ℰ ←constructSolution(𝒩_start, 𝒩_goal, 𝒢);
    success ← True;
    foreach e ∈ ℰ do  // Validate solution.
        if ¬valid(e) then
            removeEdge(e, 𝒢);
            success ← False;
        end
    end
while ¬success and connected(𝒩_start, 𝒩_goal, 𝒢);
```

We base our planner on the LazyPRM* [33] algorithm, customizing it for our application of continuous replanning in a mostly unchanged map. LazyPRM* is a variant of PRM* which builds a planning graph without checking connectivity

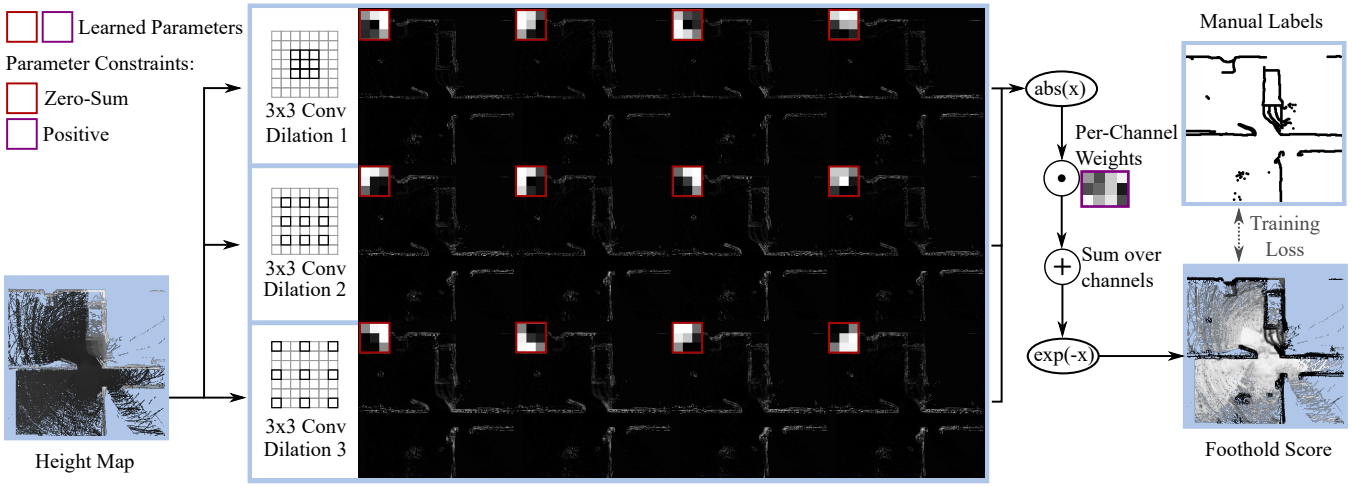


Fig. 3: We train a very shallow convolutional network to predict foothold scores from height maps, using only 20 manually labelled training examples. The center shows the actual learned convolutional weight parameters of different dilations and their resulting activation on an example input height map. The absolute value of each filter channel is multiplied with a learned weight. Finally, the negative exponential of the sum over all channels comprises the foothold score, which is trained using manually labelled examples.

when adding new nodes. Validity of a graph edge is only checked when it is part of the optimal solution at query time. This helps increase planning speeds in applications where validity checking is expensive to compute. While state validity checking using our reachability abstraction is fast, checking motions between states is not, since we do discrete motion validation at a fine resolution which makes LazyPRM* the more appropriate choice for our task. While continuous collision checking would be faster and applicable to the torso collision shape, this is not possible for the reachability shapes, which need to be in contact at each step, not only once along the motion path. The downside of this approach is that the planning graph can accumulate large numbers of possibly invalid but unchecked graph edges in regions which have not been part of the optimal solution for planner queries. If they later become relevant for a planner query, this large backlog of invalid edges needs to be worked off, which can take a long time if many invalid edges have accumulated. We combat this issue by adjusting our planner as detailed in Algorithm 1.

In our implementation we check some, but not all, graph edges for validity when expanding the graph. Our aim is to identify the newly added graph edges which are most relevant for optimal path planning and only check validity for these. We achieve this by submitting an A* query from a randomly selected graph node to the newly added node. This means that only the edges offering lowest cost connectivity to the new node are checked. This prevents excessive accumulation of unchecked invalid edges while maintaining the LazyPRM* concept of only validating optimal connections between nodes.

D. Sampling

We employ a custom sampling scheme to increase the likelihood of drawing valid samples and bias graph expansion towards unexplored regions.

1) 2D-assisted 3D sampling: Sampling the full 6-degrees of freedom (DoF) pose space uniformly during planning is inefficient because only a narrow range of height values and pitch and roll angles can fulfil the reachability requirements. Therefore, we leverage the knowledge that all reachability volumes need to be in contact with the ground by first sampling a 2D pose which is then augmented to a 3D pose using map information. Note that in order to simplify the following equations we consider the pose of the center between reachability volumes, which is the torso pose vertically offset by the nominal stance height.

First, we sample a 2D pose $\mathbf{p}_{2D}$ using the sampling distribution described in Section III-D.2:

$$\mathbf{p}_{2D} = \left[x, y, \phi\right]^T \tag{1}$$

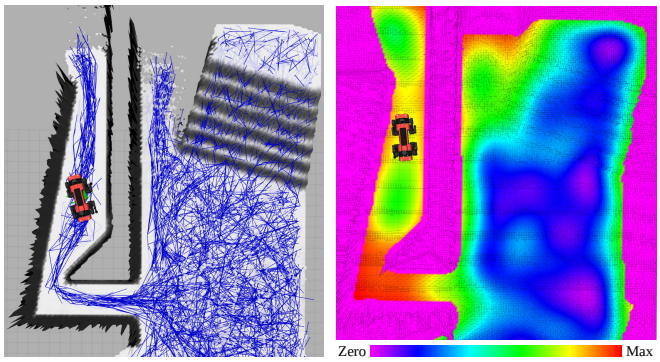
Knowing that the reachability volumes need to be in contact with the ground, we obtain the height value z from the terrain map at $\mathbf{p}_{2D}$. Assuming that the robot needs to be aligned with the terrain, we then obtain the roll and pitch angle for the 3D pose using the terrain normal $\mathbf{n}_{p_{2D}}$ of a smoothed version of the map at position p_{2D}. We first align $\mathbf{n}_{p_{2D}}$ with the sampled yaw angle using its rotation matrix $\mathbf{R}(\phi)$

$$\mathbf{n}^{\phi} = \mathbf{R}^{-1}(\phi) \cdot \mathbf{n}_{p_{2D}} = \left[x_n^{\phi}, y_n^{\phi}, z_n^{\phi}\right]^T \tag{2}$$

and then extract roll and pitch angles from $\mathbf{n}^{\phi}$

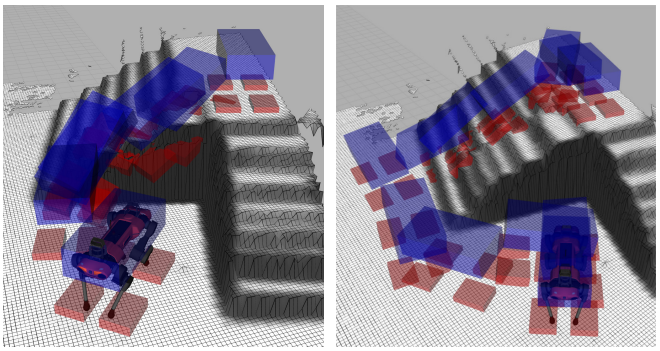
$$\mathbf{p}_{3D}^{*} = \begin{bmatrix} x \\ y \\ z \\ -\text{atan2}(y_n^{\phi}, z_n^{\phi}) \\ \text{atan2}(x_n^{\phi}, z_n^{\phi}) \\ \phi \end{bmatrix} \tag{3}$$

to obtain a 3D anchor pose $\mathbf{p}_{3D}^{*}$ which would guarantee a valid pose in a fully planar world.



(a) Planner Graph (b) Sample Probability

Fig. 4: ANYmal exploring a narrow environment in simulation. The planner graph (a) has low node density in narrow corners and high density in wide-open spaces. Accordingly, the sampling probability (b) is highest in narrow spaces and low in open areas, while unobserved space is completely ignored.



(a) Without learned foothold scores. (b) With learned foothold scores.

Fig. 5: Comparison when planning with and without learned foothold scores. When invalid foothold geometry (colored dark grey) is considered for foothold placement (a) the robot "climbs up" the side-wall of the stairs. When foothold scores are taken into account (b), a feasible path is planned.

Since height, roll and pitch of $\mathbf{p}^*_{3D}$ are not randomly sampled, we need to introduce additional perturbations to maintain probabilistic completeness of the planner and to enable valid pose discovery in rough terrain. We perturb the height by applying a uniformly sampled offset $w_{pos} \in [-\sigma_n, \sigma_n]$ along the terrain normal $\mathbf{n}_{p_{2D}}$, where σ_n is the standard deviation of terrain height inside the robot footprint. This means we apply a small offset in flat terrain and a larger offset in rough terrain. Finally, we apply random perturbations $w_{roll} \in [-\omega_{roll}, \omega_{roll}]$ and $w_{pitch} \in [-\omega_{pitch}, \omega_{pitch}]$, where ω_{roll} and ω_{pitch} are tuning parameters. This gives us our final 3D sample:

$$\mathbf{p}_{3D} = \mathbf{p}^*_{3D} + \begin{bmatrix} \mathbf{n}_{p_{2D}} \cdot w_{pos} \\ w_{roll} \\ w_{pitch} \\ 0 \end{bmatrix} \tag{4}$$

2) Adaptive Sampling Distribution: Since both the A* algorithm used to solve planner queries, as well as the nearest-neighbor search for node insertion scale non-linearly with the number of graph nodes, reducing this is a major concern for maintaining fast planning speeds. Uniform sampling tends to accumulate nodes in wide-open spaces, where most drawn samples are valid, while node density in narrow spaces is low (see Figure 4). This leads to issues when transitioning between open and narrow spaces. Therefore, we compute the sample probability as the inverse density of graph nodes.

We overlay a grid over the planning space and count the number of graph nodes per grid cell, which is then average-filtered with a kernel the size of the robot footprint to obtain the smoothed sample density $\mathcal{D}(x, y)$. The sample probability density $p(x, y)$ is computed as the maximum sample density over the sampling space, minus the local sample density, with a small bias ϵ to prevent numerical issues with constant density and to maintain probabilistic completeness.

$$p(x, y) \approx \max_{x,y}(\mathcal{D}(x, y)) - \mathcal{D}(x, y) + \epsilon \tag{5}$$

The yaw angle ϕ is sampled uniformly from $[0, 2\pi[$.

IV. Experimental Results

We validate individual elements of our approach in simulation and demonstrate deployment of the entire method on the quadruped ANYmal. It is equipped with four Intel Realsense D435 depth cameras which are used to build a robocentric 2.5D elevation map of the environment [34]. We use a blind learning-based locomotion controller [31], which is able to overcome obstacles up to 17cm in height and moves at an average speed of $0.68\text{m}\,\text{s}^{-1}$. We plan on a $12\text{m}\times12\text{m}$ grid with 4cm resolution using a shortest-path objective and a simple P-controller as path follower. The robot pose is obtained from onboard localization.

A. Simulation

Simulation experiments were run on a Desktop computer with an Intel i7-8700K CPU, a Nvidia RTX 2080 GPU and 32 GB of RAM. Figure 6 shows paths planned in simulation over various challenging terrain. We are able to plan in very narrow spaces, over uneven terrain and over features which have been unapproachable by existing methods [1], like negative obstacles and winding stairs, even with incomplete map information caused by online mapping. Additionally, we show the effect the learned foothold score has on the resulting paths and finally, how our extensions to LazyPRM* improve worst-case performance over the original algorithm and other commonly used planning algorithms.

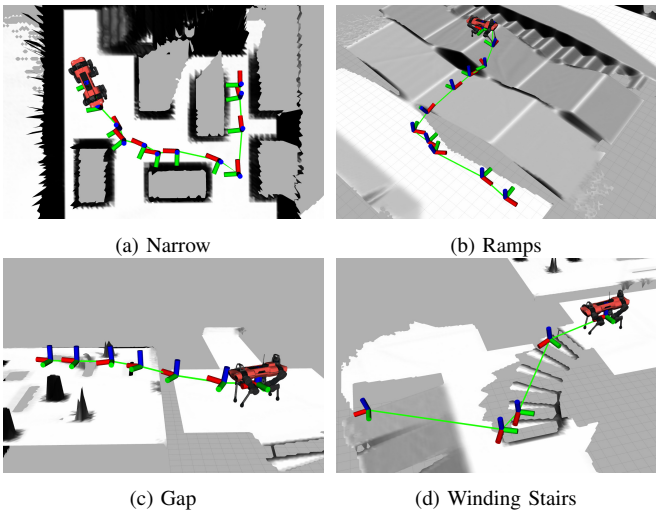
1) Foothold Score Learning: We manually labelled 26 height maps of size 200×200 obtained from log data of real robot deployments on a diverse set of terrains, and trained the foothold score network using a binary cross-entropy loss. 20 maps were used for training and 6 for validation. The network fully converged in 32 seconds using the Adam optimizer [35]. Inference times are 0.35 milliseconds on GPU

and 3 milliseconds on CPU for a map of the aforementioned size. We use a foothold score threshold of 0.5 to classify terrain as steppable or not.

Figure 5 shows how using our foothold score to restrict available collision geometries for the reachability volumes results in a more realistic, feasible path, when planning up the side of a set of stairs. In the basic configuration without using foothold scores, the side-wall of the stairs is considered a valid object to step onto and the planner consequently plans a path which climbs up the side of the stairs, which is not practically achievable by the robot. When we use the foothold score to remove invalid foothold geometry, colored dark grey in figure 5, the result is a feasible path, which avoids the side-wall and high steps on the right side of the staircase.

2) Planning Time: We determine a real-time planning threshold T_{thres} based on the maximal locomotion speed of our target platform, $0.68\mathrm{m/s}$, and the size of the robocentric planning map, $12 \times 12\mathrm{m}$, or $6\mathrm{m}$ in each direction around the robot, $T_{thres} = \frac{6\mathrm{m}}{0.68\mathrm{m\,s^{-1}}} \approx 8.8\mathrm{s}$. We compare the worst-case planning times of each method against this threshold, instead of the average time, because any outlier which significantly exceeds the target update interval even once during continuous operation will cause either mission delays or lead to unsafe robot paths and is therefore not real-world ready.

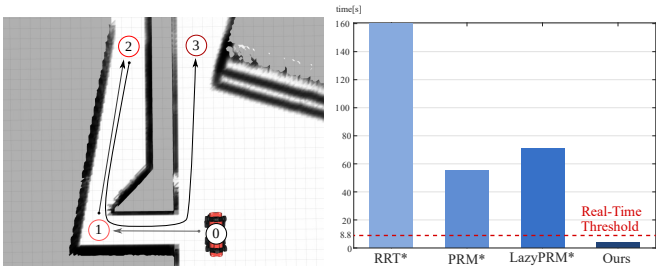
To evaluate planning time performance mimicking a real deployment scenario, we sequentially plan to three goal points, while continuously replanning, in an environment where a narrow corridor with an extremely tight corner just the size of the robot branches off of an open area, shown in Figure 7(a). This environment showcases two key factors which can cause planning time outliers for planning algorithms, narrow passages and planning into previously observed but not planned to regions. The experiment is repeated for four algorithms, RRT*, PRM*, LazyPRM* and our extension to LazyPRM*, all of which are probabilistically complete and asymptotically optimal. All methods use our foothold score predictor for collision checking and sample using our proposed 2D-assisted 3D sampling. We run each algorithm for at least one second, to allow for a low-cost path to be found, and until a solution is found. We therefore want the planner to ideally find a solution in one second, to allow for continuous and fast replanning and to never exceed T_{thres}. Results of worst-case performance per algorithm are shown in Figure 7(b). RRT* needs over a minute to plan into the tight turn of goal 1 and therefore fails to accomplish this. Similarly, PRM* takes 52 seconds to find an initial solution but allows for fast replanning once a graph has been built. While LazyPRM* finds an initial solution in 8.0 seconds due to lazy validating of graph edges, this leads to a single very long planning time of 68 seconds when planning from goal 2 to goal 3. Many unchecked edges between the left and right passages have accumulated which need be validated during a single planner query. Finally, our method finds an initial solution in 3.1 seconds, faster than LazyPRM*, due to the density-based sampling scheme, which leads to more graph nodes in difficult regions, as shown in Figure 8. At the same time, unchecked edges do not accumulate due to our graph expansion scheme, and therefore, planning between goal 2 and goal 3 does not exceed the real-time threshold. In summary, while density-based sampling helps increase planning speed through narrow passages, our graph expansion scheme is the crucial component to avoid excessive planning times.



(a) Narrow (b) Ramps

(c) Gap (d) Winding Stairs

Fig. 6: Example paths planned over various terrain in simulation. Our method can plan in (a) very narrow environments, (b) uneven terrain, (c) over gaps and (d) over winding stairs, even with incomplete maps.



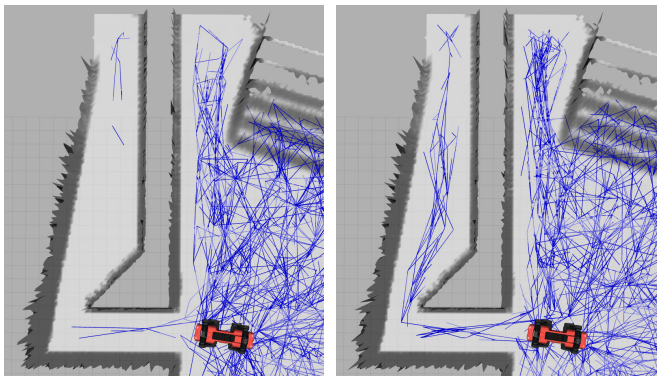
(a) Environment and goal points. (b) Worst-case planning time per algorithm.

Fig. 7: We sequentially navigate the robot to three goal points (a) while continuously replanning. Other planning algorithms have excessive planning time outliers (b), which make continuous replanning impossible in practical applications.

B. Real Robot

The ANYmal-C quadruped used for our experiment is equipped with an Intel i7-8850H processor with 16GB of RAM which in addition to our planner was running camera drivers, elevation mapping and ICP-based SLAM.

Our method has been tested over several months during multiple fully autonomous mission tests for the DARPA Subterranean Challenge in various challenging underground environments. In this context, goal poses were provided to our planner by a high-level exploration planner [36], which finds map frontiers to maximize information gain. Hereafter we present results for one specific test conducted in an abandoned industrial facility in Rümlang, Switzerland. An

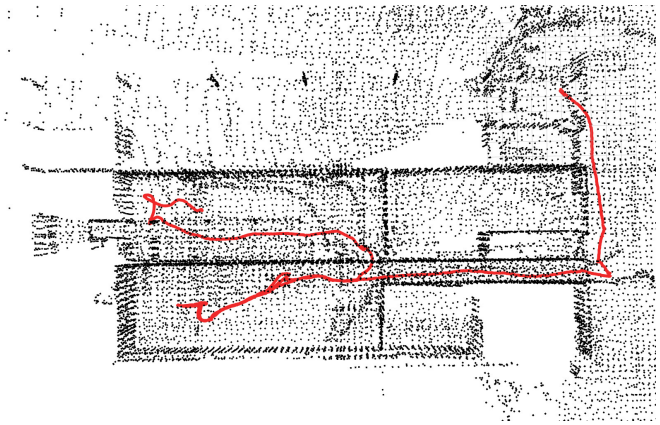


(a) Uniform Sampling (b) Density-based Sampling (Ours)

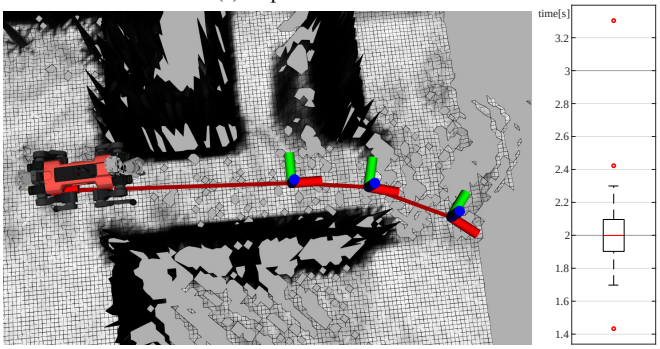
Fig. 8: Valid edges of the planning graph after sampling for 5 seconds using (a) uniform sampling and (b) our density-based sampling. The graph does not connect through the narrow corner on the left with uniform sampling while our method does.

overview of the environment is shown in Figure 9(a), which consists of narrow corridors and trip hazards on the ground.

Our planner was able to safely navigate over steps up to 17cm in height and corridors 90cm in width, even in the presence of incomplete maps corrupted with artifacts, as shown in Figure 9(b). For real deployment, we changed the planning scheme to query a new path every 2 seconds instead of always sampling for a fixed time before returning, even if a solution was found earlier. This helps to achieve consistent path update intervals, which consistently stay below the real-time threshold of 8.8s, as shown in Figure 9(c), and improves path tracking performance.



(a) Explored Environment



(b) Planning through a narrow corridor. (c) Planner Update Intervals

Fig. 9: Fully autonomous deployment of our planner in an abandoned industrial facility with narrow corridors and trip hazards (a). The planner is able to plan through difficult environments with incomplete maps (b) and maintains consistent update intervals (c).

V. Conclusion

In this work we presented an approach capable of real-time path planning over challenging terrain for legged robots. It uses a reachability-based robot abstraction enhanced by a learned foothold score prediction network to evaluate whether a robot can negotiate a given terrain without requiring an explicit traversability measure. We expand the LazyPRM* algorithm with a custom graph expansion scheme which alleviates issues with accumulating unvalidated graph edges, which can lead to spikes in planning time. Our sampling scheme based on node graph density favors sampling in underexplored map regions, thereby accelerating planning in narrow environments. Finally, we demonstrated our approach on the ANYmal quadruped both in simulation and in reality, showcasing real-time in-the-loop performance. Future work will focus on further improving the sampling scheme with a learned sampling distribution, introducing a more sophisticated cost measure to replace the shortest path objective and combining it with our previous work on semantic foothold score prediction [5], [6].

References

[1] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *IROS*. IEEE, 2016, pp. 1184–1189.

[2] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart, "Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments," *Journal of Field Robotics*, vol. 34, no. 5, pp. 940–984, 2017.

[3] S. Tonneau, N. Mansard, C. Park, D. Manocha, F. Multon, and J. Pettré, "A reachability-based planner for sequences of acyclic contacts in cluttered environments," in *International Symposium on Robotics Research (ISSR 2015)*, 2015.

[4] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *IROS*. IEEE, 2016, pp. 38–44.

[5] 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, 2019.

[6] L. Wellhausen, R. Ranftl, and M. Hutter, "Safe robot navigation via multi-modal anomaly detection," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1326–1333, 2020.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[8] R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, and A. Giusti, "Image classification for ground traversability estimation in robotics," in *Int. Conf. on Advanced Concepts for Intelligent Vision Systems*. Springer, 2017, pp. 325–336.

[9] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *IROS*. IEEE, 2017, pp. 1366–1373.

[10] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and

N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, 2018.

[11] 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*, vol. 5, no. 4, pp. 5370–5376, 2020.

[12] B. Rothrock, R. Kennedy, C. Cunningham, J. Papon, M. Heverly, and M. Ono, "Spoc: Deep learning-based terrain classification for mars rover missions," in *AIAA SPACE 2016*, 2016, p. 5539.

[13] D. M. Bradley, J. K. Chang, D. Silver, M. Powers, H. Herman, P. Rander, and A. Stentz, "Scene understanding for a high-mobility walking robot," in *IROS*. IEEE, 2015, pp. 1144–1151.

[14] K. Otsu, M. Ono, T. J. Fuchs, I. Baldwin, and T. Kubota, "Autonomous terrain classification with co-and self-training approach," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 814–819, 2016.

[15] A. Valada, J. Vertens, A. Dhall, and W. Burgard, "Adapnet: Adaptive semantic segmentation in adverse environmental conditions," in *ICRA*. IEEE, 2017, pp. 4644–4651.

[16] D. Kim, J. Sun, S. M. Oh, J. M. Rehg, and A. F. Bobick, "Traversability classification using unsupervised on-line visual learning for outdoor robot navigation," in *ICRA*. IEEE, 2006, pp. 518–525.

[17] D. Barnes, W. Maddern, and I. Posner, "Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy," in *ICRA*. IEEE, 2017, pp. 203–210.

[18] N. Hirose, A. Sadeghian, M. Vázquez, P. Goebel, and S. Savarese, "Gonet: A semi-supervised deep learning approach for traversability estimation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3044–3051.

[19] P. Fernbach, S. Tonneau, A. Del Prete, and M. Taïx, "A kinodynamic steering-method for legged multi-contact locomotion," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3701–3707.

[20] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[21] J. Norby and A. M. Johnson, "Fast global motion planning for dynamic legged robots," 2020.

[22] Y. Zhao, X. Chai, F. Gao, and C. Qi, "Obstacle avoidance and motion planning scheme for a hexapod robot octopus-iii," *Robotics and Autonomous Systems*, vol. 103, pp. 199–212, 2018.

[23] M. Y. Harper, J. V. Nicholson, E. G. Collins, J. Pusey, and J. E. Clark, "Energy efficient navigation for running legged robots," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6770–6776.

[24] D. Belter, J. Wietrzykowski, and P. Skrzypczyński, "Employing natural terrain semantics in motion planning for a multi-legged robot," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 3-4, pp. 723–743, 2019.

[25] Y.-C. Lin and D. Berenson, "Humanoid navigation in uneven terrain using learned estimates of traversability," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 9–16.

[26] W. Reid, R. Fitch, A. H. Göktoğan, and S. Sukkarieh, "Sampling-based hierarchical motion planning for a reconfigurable wheel-on-leg planetary analogue exploration rover," *Journal of Field Robotics*, vol. 37, no. 5, pp. 786–811, 2020.

[27] J. Guzzi, R. O. Chavez-Garcia, M. Nava, L. M. Gambardella, and A. Giusti, "Path planning with local motion estimations," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2586–2593, 2020.

[28] B. Yang, L. Wellhausen, T. Miki, M. Liu, and M. Hutter, "Real-time optimal navigation planning using learned motion costs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[29] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.

[30] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2118–2124.

[31] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020.

[32] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on information theory*, vol. 29, no. 4, pp. 551–559, 1983.

[33] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2951–2957.

[34] P. Fankhauser and M. Hutter, "A universal grid map library: Implementation and use case for rough terrain navigation," in *Robot Operating System (ROS)*. Springer, 2016, pp. 99–120.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference for Learning Representations*, 2015.

[36] T. Dang, F. Mascarich, S. Khattak, C. Papachristos, and K. Alexis, "Graph-based path planning for autonomous robotic exploration in subterranean environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3105–3112.