ArtPlanner: Robust Legged Robot Navigation in the Field

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 had previously proposed a reachability-based navigation planner for legged robots, which approximates the robot morphology by a set of reachability and body volumes. It uses a learned convolutional neural network to restrict valid footholds to feasible regions and incrementally builds a probabilistic roadmap graph for fast planning queries. We extend this work using an additional neural network trained in simulation, which predicts locomotion cost and risk over a given terrain. This produces safe and low-cost paths and increases planning speed by leveraging batched processing of the cost prediction network on GPU to rapidly validate navigation graph edges through locomotion risk thresholding. We deployed this navigation planner in team CERBERUS' winning entry to the DARPA Subterranean Challenge finals, where the proposed planner powered the local navigation of four ANYmal quadrupeds during all three competition runs without a single navigation failure.**

***Paper Type* – Original Work.**

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, in previous work [3], we have chosen to apply a different, simplified robot representation when planning for legged systems, based on limb reachability abstractions [4]. 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.

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

Authors are with the Robotic Systems Lab, ETH Zürich, Switzerland

Fig. 1: Team CERBERUS won the DARPA Subterranean Challenge with four ANYmal quadrupeds deployed during the final run. The navigation planner presented in this work guided all four robots safely during the hour-long mission.

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. While visual semantic information can be used [5], [6], we trained a convolutional neural network (CNN) to predict foothold scores from the planner map and only allow safe regions to support footholds by removing unsafe terrain from the reachability collision map [3]. This generally produces feasible paths, but the shortest path cost used leads to paths close to walls and edges, which can be risky due to imperfect path following and uncertainty in the environment. Therefore, we extend our method by instead optimizing a learned motion cost [7], [8] which is computed by a CNN which predicts traversal risk and cost from a height map.

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. In our previous work [3], we built a PRM*[9] planning graph incrementally by maintaining a planning graph between planning queries, arguing that most map updates in a static environment will not invalidate the planning graph. This approach necessitates graph maintenance between map updates which can typically be performed quickly during normal operation. However, this adds heuristics and algorithmic complexity to identify updated graph elements and the extreme challenges posed by the DARPA Subterranean Challenge (SubT), like smoke, and dynamic obstacles, triggered maintenance operations more frequently than previously assumed. This cancelled most of the performance benefits

originally achieved through incremental map updates. We therefore altered our graph creation method to create a new graph every time the map is updated: We lazily sample candidate pose vertices using our density-based sampling algorithm [3] until we have reached a chosen number of valid poses. This ensures that we exceed a minimal vertex density, because we operate on a fixed-size map. We then validate all graph edges at once by applying a locomotion risk threshold, leveraging massively parallel execution of the cost prediction network on GPU.

Note that while batch risk querying for edge validation would be compatible with our previously used persistent graph, the error-prone and potentially expensive accounting for updated graph edges would still be necessary. This, however means that the algorithm proposed in this work is not probabilistically complete, because we terminate sampling at a fixed number of graph nodes. This is a limitation of this work but has shown to be unproblematic in practice, since the planner runs fast enough to sample the map densely, while maintaining real-time update rates.

II. Related Work

Navigation planning for mobile robots is a vast field of research with a manifold of different approaches.

Most navigation approaches for mobile robots use a geometric environment representation as their basis for planning [1], [2], [10], [11]. They use various different terrain representations for planning, most commonly 2.5D height maps [1], [10], point clouds [2] or truncated signed-distance field (TSDF)s [11]. Because planning in full 3D representations is currently computationally prohibited [12], we chose to work with 2.5D height maps as environment representation. 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, even other SubT competitors which deployed legged robots [13]. Thereby, they neglect the much higher mobility which legged robots provide due to their ability to step over obstacles. Other work on navigation planning specifically for legged robots either only considers cases of obstacle avoidance on flat terrain [14], [15] or does additional contact planning, which pushes computational complexity past the real-time mark [16], [17], [18]. Approaches which learn traversability [10] or motion cost [7], [8] are powerful, but are either too slow due to the sequential querying of neural networks during sampling-based planning [7] or struggle in tight spaces where precise motion checking is necessary [8].

We use a reachability-based robot representation [4], [3] and combine our previous work with batched motion cost computation. This is the first work which combines geometric collision checking and learned motion costs in a real-time capable navigation planner.

III. Method

Our approach extends our previous reachability-based planner [3] using a learned cost prediction network [7], [8] for faster planning and higher quality paths. We will briefly outline the components adopted from our previous approach essential to understanding the new additions and refer to our previous work [3] for details.

A. Reachability Planning

To check for validity of sampled robot poses, we use a reachability-based approach [4]. 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. To check for this condition, we decompose the robot body into volumes representing torso collisions and leg reachability.

In order to account for geometry which should not be used as support surface (i.e. walls), we trained a CNN to predict a foothold score based on height map information and use this to constrain the regions considered for valid footholds. Geometry which has a low foothold score is disregarded for collision checking of the limb reachability volumes but is still considered for the torso. A valid pose is therefore a pose where all reachability volumes are in contact with valid geometry, while the torso volume is not in collision with any geometry.

Our planner is based on the sampling-based LazyPRM* [19] algorithm and only checks the sampled pose for validity when adding a new node to the graph, but does not immediately check the validity of newly added edges. Edge checking in basic LazyPRM* is only performed when querying a path. In our previous work [3] we also check the edges part of the optimal solution between a randomly sampled node and a newly added node but this was abandoned, as discussed in Section III-C.

We use a custom sampling scheme which uses a 2D pose sample augmented to a 3D pose using map information and biases sampling towards regions with low node density.

B. Motion Cost

While our previous work simply used a shortest-path cost for planning [3] due to the difficulty of analytically determining locomotion cost and risk, other recent work has proposed to use a learned neural network to compute cost [7], [8]. Building upon these works, we use a neural network which computes the energy and time required and failure risk associated with moving from a query location in the height map to a given relative 2D pose [8].

The inputs to the network are a patch of the height map centered around the robot, the current yaw orientation of the robot, as well as the 2D goal pose relative to the robot. The network outputs the time c_t and energy c_e required to move to the target pose as well as the motion risk c_r, which is the probability that this transition fails. The network is trained using data generated in simulation, where the robot is spawned in a random location on a randomly generated height map and a goal pose in a certain range around the robot is randomly selected. We then naively give a directional command to the robot to move towards this goal pose while the robot experiences external disturbances. This is repeated a number of times to compute the failure probability. Time

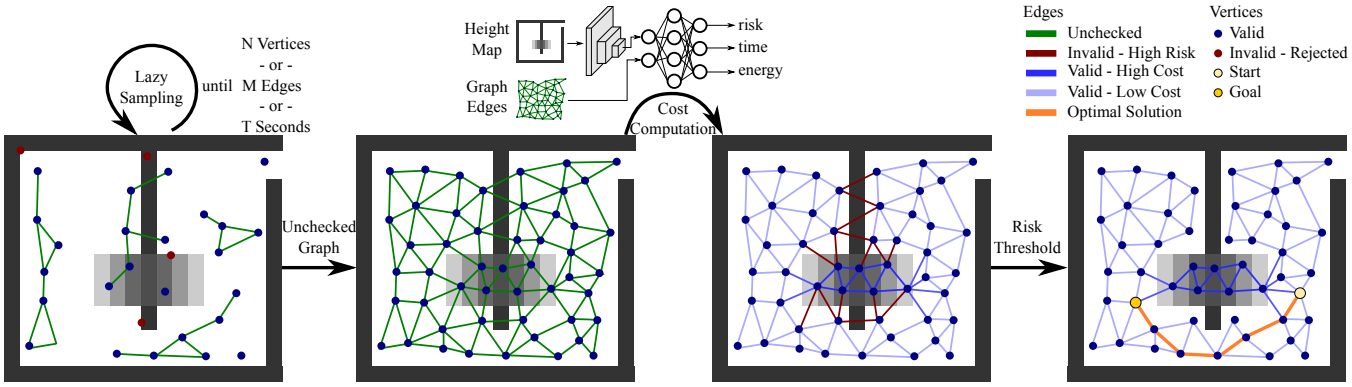


Fig. 2: After every map update we lazily sample a planning graph until a set number of vertices or edges is reached or a time threshold has been exceeded. The height map and planning graph are then fed to a neural network, which computes the motion risk and cost for every graph edge. High risk edges are removed from the graph and we perform an A* query to find the optimal path.

and energy values are averaged over all attempts which successfully reached the goal.

The risk value is used to determine validity of planning graph edges, while the graph edge cost is computed as a weighted sum of time c_t, energy c_e and risk c_r:

$$c = w_t \cdot c_t + w_e \cdot c_e + w_r \cdot c_r \tag{1}$$

c_t and c_e are normalized with their respective maximal value in the training dataset, whereas c_r expresses the probability of failure. Battery runtime of the ANYmal robots was not of concern and energy consumption c_e showed to be highly correlated with time c_t, so we chose to disregard this cost term. Since any navigation error would be potentially mission-ending during the SubT *Finals*, we put a high weight on risk. Consequently, our chosen cost weights were $w_t := 1$, $w_e := 0$ and $w_r := 5$.

C. Graph Construction

We construct our planning graph using lazy sampling, where only the added vertices but not edges are immediately checked for validity. When querying a solution in a regular LazyPRM* graph, an A* search is performed and all edges which are part of the optimal solution are checked for validity. Any invalid edges are removed from the graph and the process is repeated until either all edges returned by the A* search are valid (success), or the start and goal vertices are no longer connected (failure). This makes the path query time highly non-deterministic since the number of A* search iterations strongly depends on the complexity of the environment which determines the number of invalid edges. As discussed in our previous work [3], this can lead to excessive planning times which make real-time applications impractical or impossible. While our previously suggested graph expansion method [3] can compensate for this issue in most cases, during extensive real-world testing in preparation for the SubT challenge we still encountered some instances of long planning times in difficult environments where the number of invalid edges in the graph is exceedingly large.

To achieve consistent and low planning times, we therefore leverage batched edge cost computation using our motion cost network, detailed in Figure 2. Every time a new planning query is received and the map has updated, we build a new planning graph by lazily sampling until the graph exceeds either N valid vertices, M unchecked edges or we have sample for more than T seconds. Since the number of edges determines the batch size for the cost query, its upper bound limits the maximal computation time used by the cost query which validates graph edges. The other two termination criteria prevent unnecessarily long and dense sampling in certain environments where either the traversable area of the map is very small (T limit) or the geometry allows every node to only have a small number of neighbors (N limit), like in narrow corridors.

A batched query of the cost network is then performed for every edge in the graph at once, which can be executed efficiently on GPU. Edges which exceed a risk threshold R are removed and valid edges are assigned a cost, according to Equation (1).

We now have a fully validated graph with assigned edge costs and consequently, a single A* search will return the the optimal path between two query nodes.

IV. Results

A. DARPA Subterranean Challenge

The DARPA Subterranean Challenge (SubT) Finals took place from September 21th - 24th in the Louisville Mega Caverns in Kentucky, USA. Eight participating teams competed for a $5 million prize purse and deployed autonomous systems to map, navigate and search underground tunnel, urban and cave spaces, which had been constructed inside of the Mega Caverns.

In each competition run, teams were required to find a number of known artifacts, like backpacks or cell phones, which were hidden inside the course in unknown locations, and report their identity and position with an accuracy of 5m. One point was awarded per correctly reported artifact, while the number of reporting attempts was limited to avoid brute-forcing the solution. Only a single operator was allowed to remotely supervise all robots and no network connectivity inside the competition area was available, unless created by

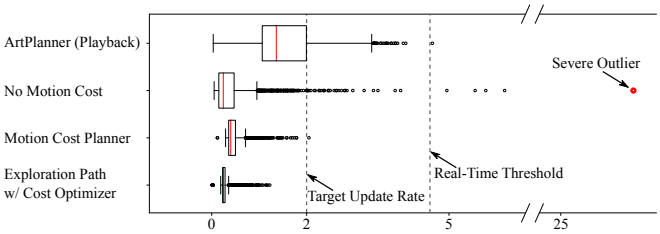


Fig. 3: Planning times for four navigation planners on data from the SubT *Finals* Prize Run.

the robots. Hence, robots had to operate autonomously for large parts of the competition.

Three competition runs were spread over the three days of the finals, two 30 minute pre-rounds and a 60 minute grand final which was the only run used to determine the winners.

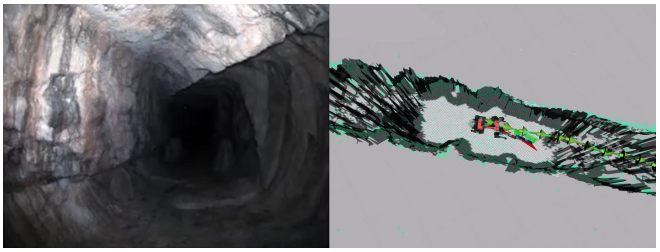
Team CERBERUS deployed four ANYmal [20] robots as part of their solution, which ran the navigation planner proposed in this work guided by waypoints from a global exploration planner (GBPlanner2) [21].

B. Computational Performance

We compare the computational performance of ArtPlanner to our previous work without motion cost [3], a planner based only on motion cost [8] as well as the exploration path, refined by a cost optimizer [8]. We feed all planners the same data recorded during the competition Prize Run, played back on a desktop machine with an Intel i7-8700K CPU, a Nvidia RTX 2080 GPU and 32 GB of RAM. Planning times are shown as box plots in Figure 3. Our chosen target update rate was to publish a new path every 2 seconds and the real-time threshold for our $8\,\mathrm{m}\times 8\,\mathrm{m}$ map at a locomotion speed of $0.9\,\mathrm{m\,s^{-1}}$ was $4.44\,\mathrm{s}$. The real-time threshold [3] is the time the robot requires to reach the edge of the height map at maximal speed.

We set ArtPlanner's maximal sampling time T to 2 seconds, which does not factor in map processing and motion cost query time. Therefore, ArtPlanner can exceed the target time if the maximum sampling time is reached, with a maximal planning time of $4.65\,\mathrm{s}$. Consequently, we achieved the target time in 75% of cases, exceeding the real-time threshold only a single time, by $0.21\,\mathrm{s}$. The *No Motion Cost* planner has fast query times in the median. Unfortunately, this method can produce severe planning time outliers on rare occasion due to the graph validation at query time, as discussed in Section III-C. We observed a time of $26.53\,\mathrm{s}$ in the data we used to generate Figure 3 but observed even longer times in other instances. This shows the benefit of our graph validation method through batch motion cost query. We can limit our planning time and can therefore avoid the planner being unresponsive for a long period of time. The *Motion Cost Planner* uses a fixed planning graph and therefore only has to perform a batch motion cost query, but no sampling. Since the optimization stage also always performs a fixed number of iterations, planning is fast and the target time can be reached in all cases. Since the *Exploration Path w/ Cost Optimizer* only deploys the optimization stage of the *Motion Cost Planner* it is even faster.

(a) Successful planning in narrow and cluttered tunnel section of the SubT course.



(b) Very low ceiling in the cave section produces artifacts in the height map.

Fig. 4: Onboard data recorded during the SubT final run. The left column shows an image from the forward-facing RGB camera. The right column visualizes mapping and planning data. The height map has geometry with low foothold score colored in black, while white indicates steppable terrain. The thick green arrows show the reference path computed by the exploration planner, which serves as goal for our navigation planner. The sequence of coordinate axes is the path output by our planner.

Overall, all evaluated planners would be fast enough for real-time operation in the nominal case. However, the worst-case time of the *No Motion Cost* planner is too long to deploy it in a competition environment.

C. General Performance

Team CERBERUS won SubT and all four robots finished the final hour-long mission in an operational state with no navigation failures. We only observed a single incident, where one robot briefly got stuck on a thin, vertical pole, when the computed collision-free path was not properly followed by the path tracker. This was due to a software bug in the path follower, discovered only after the competition, which caused a $300\mathrm{ms}$ delay before accepting new paths.

Other than that no collisions with the environment were observed and our planner was crucial in ensuring robustness of our autonomy solution. The exploration planner, which uses a very simple collision model on a much coarser map, frequently proposed paths which would have lead to collisions, which our navigation planner prevented, as shown in Figure 4(a).

The biggest challenge for the planner was the very low ceiling height which barely exceeded the stance height of the robot in some parts of the course. This was lower than we anticipated and lead to severe artifacts in the height map, as shown in Figure 4(b). While the robots were still able to plan, the short planning distance and the fixed update rate lead to slow progression in these sections.

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] 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)*. IEEE, 2021, pp. 6914–6921.

[4] 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.

[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] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Agha-Mohammadi, "Step: Stochastic traversability evaluation and planning for risk-aware off-road navigation," *Robotics: Science and Systems*, 2021.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, May 2022.