ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

RSL Robotic Systems Lab
Prof. Dr. Marco Hutter

Master Thesis

Global Path Planning in a Digital-Twin

Spring Term 2024

Supervised by:
Marko Bjelonic, Lorenz Wellhausen
David Hoeller, Joonho Lee

Author:
Ajaykumar Unagar

Intellectual Property Agreement

The student acted under the supervision of Prof. Hutter and contributed to research of his group. Research results of students outside the scope of an employment contract with ETH Zurich belong to the students themselves. The results of the student within the present thesis shall be exploited by ETH Zurich, possibly together with results of other contributors in the same field. To facilitate and to enable a common exploitation of all combined research results, the student hereby assigns his rights to the research results to ETH Zurich. In exchange, the student shall be treated like an employee of ETH Zurich with respect to any income generated due to the research results.
This agreement regulates the rights to the created research results.

1. Intellectual Property Rights

1. The student assigns his/her rights to the research results, including inventions and works protected by copyright, but not including his moral rights ("Urheberpersönlichkeitsrechte"), to ETH Zurich. Herewith, he cedes, in particular, all rights for commercial exploitations of research results to ETH Zurich. He is doing this voluntarily and with full awareness, in order to facilitate the commercial exploitation of the created Research Results. The student's moral rights ("Urheberpersönlichkeitsrechte") shall not be affected by this assignment.
2. In exchange, the student will be compensated by ETH Zurich in the case of income through the commercial exploitation of research results. Compensation will be made as if the student was an employee of ETH Zurich and according to the guidelines "Richtlinien für die wirtschaftliche Verwertung von Forschungsergebnissen der ETH Zürich".
3. The student agrees to keep all research results confidential. This obligation to confidentiality shall persist until he or she is informed by ETH Zurich that the intellectual property rights to the research results have been protected through patent applications or other adequate measures or that no protection is sought, but not longer than 12 months after the collaborator has signed this agreement.
4. If a patent application is filed for an invention based on the research results, the student will duly provide all necessary signatures. He/she also agrees to be available whenever his aid is necessary in the course of the patent application process, e.g. to respond to questions of patent examiners or the like.

2. Settlement of Disagreements

Should disagreements arise out between the parties, the parties will make an effort to settle them between them in good faith. In case of failure of these agreements, Swiss Law shall be applied and the Courts of Zurich shall have exclusive jurisdiction.

Place and date	Signature

Contents

Acknowledgements

To Prof. Marco Hutter – thanks for allowing me to carry out this thesis with the RSL lab and providing infrastructure for the research.

To Marko, Lorenz, David, and Joonho – These were fun 6 months. The current state of the thesis would not have been possible without our interesting discussions. Thanks for continuously guiding me and giving me autonomy in pursuing my own ideas.

To Mom, Dad, Sahil, and Ankita – Thanks for handling all my tantrums in these 6 months of the thesis.

To Philipp, Athina, Carla, Daniel – Thanks for the fun lunch breaks :) They were very much needed for going through each day.

To Philipp, Nikolaos, Romeo – You guys are the best. I continuously look up to you for the motivation and to strive for the best in my work.

Abstract

The global path planning problem focuses on finding a safe and short path for the robot to traverse between two places in the world. A Digital-Twin of the real-world represents most of the information required for path planning. In this thesis we present methods that use the Digital-Twin of the environment as a prior and build navigation graphs. These navigation graphs are used for real-time path planning during robots' autonomous operation. First, we show a simulation based approach that builds a navigation graph on a large triangular mesh of an environment. We also show planned paths of more than 100m in length, spanning indoor and outdoor terrains, and passing through multiple floors. Second, we propose a deep learning-based navigation cost predictor from the point-cloud data. This trained cost predictor can be used to build the navigation graphs an order of magnitude faster than the simulation-based approach. We also demonstrate that how our point-cloud-based navigation cost predictor helps in real-time global re-planning to handle large static map changes.

Symbols

Symbols

ψ	yaw angle
α, β	weights
f, g	functions

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
CAD	Computer Assisted Design
DEM	Digital Elevation Model
DNN	Deep Neural Network
CNN	Convolution Neural Network
DGCNN	Dynamic Graph Convolution Neural Network
RL	Reinforcement Learning
PRM	Probabilistic Roadmap
RRT	Rapidly-exploring Random Tree
SLAM	Simultaneoud Localization And Mapping
VIN	Value Iteration Network
IRL	Inverse Reinforcement Learning
IL	Imitation Learning
GPU	Graphics Processing Unit
MAE	Mean Absolute Error

Chapter 1

Introduction

Global path planning is the problem of estimating a safe path for the robot to traverse between two points in its operational world. This is a crucial problem in autonomous deployments of the robots in applications such as last-mile delivery, search-and-rescue, or inspection of large industrial facilities. Solving global path planning problem allows us to expand the autonomous capabilities of the robots such that they require minimal human intervention during the operation. Global path planning becomes very crucial for highly functional autonomous robots, such as Anymal [1], which can traverse through difficult terrains like vegetation, stairs, or slippery slopes [2]. For such robots, a well defined path allows us to exploit their true capabilities and finishing the operation in a minimum amount of time.

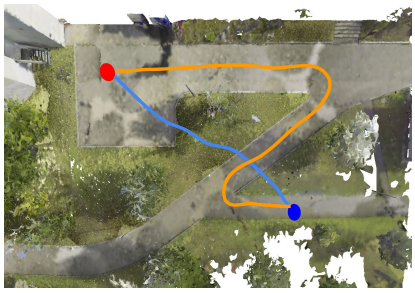

Figure 1.1: Global Path Planning

To solve the global path planning problem the robot needs access to a representation of its operational world. Past research in this direction used various environment representations, that are either explicit such as topological graph [3, 4, 5, 6], obstacle map [7, 8, 9], height-maps [10, 11, 12, 13, 14], or implicit, such as images of expert path following [15, 16, 17], exploration experience in an environment [18, 19, 20, 21]. However, most of these representations do not scale to the large-scale 3D environments, which can have multi-floor structures and span to indoor as well as outdoor areas. Also, converting a real-world environment in any such representation results in information loss, and paths planned using such representations can be sub-optimal. On the other hand, a digital-twin of the environment can accurately represent its terrain characteristics. Furthermore, such twins can be extended with varied levels of richness such as visual, semantic, or topological, with a minimal information loss. Digital-Twin can be represented by various 3D data structures like point cloud, triangular mesh, or digital elevation model (DEM).

In this thesis, we explore an application of the digital-twin for building the navigation graphs for the large-scale environments that allow robots to plan a global path between any two points in the real world. This approach has shown promise in the previous works. In PRM-RL [22] authors combine reinforcement learning-based local-planner with a sampling-based global planner (PRM) to build a navigation graph in a digital twin. Also, in RL-RRT [23] authors trained a reachability estimator using RL policy in a simulation, and applied RRT planner in a digital-twin using a trained reachability estimator to bias the tree growth towards promising

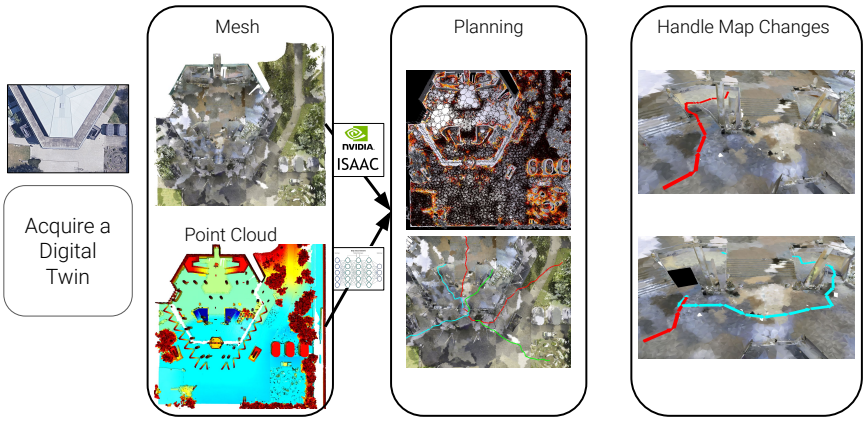

Figure 1.2: Overview

regions. In both these approaches, the authors used a simplified 2D building plan as a digital twin. However, with the new LiDAR sensor technologies, creating detailed digital twins of the environments is becoming faster and cheaper. In some cases these digital twins can even be built autonomously by the robots [24]. With a digital-twin stored as a point cloud or a mesh, it can represent varied 3D multi-floor and overhanging structures along with outdoor and indoor terrains. Since the targeted robot platforms for this thesis are legged robots such as Anymal, the rich terrain information represented by a mesh or a point cloud allows us to find near-optimal paths.
Hence, in this thesis, we present a method to do global path planning in the real world using its mesh or point cloud as an input. However, these 3D models can be large in size and they can not be stored as a whole on the Anymal platform. We alleviate this problem by pre-building the navigation graphs using PRM global planner in a digital twin. To establish connectivity between two nodes in a PRM planner, we propose a different method for mesh and point-cloud-based digital twins. First, for the triangular mesh-based digital twin, we estimate the locomotion costs between two nodes through simulations. With a given controller for the anymal, we propose a GPU-aided PRM building algorithm that relies on NVIDIA's Isaac Gym simulator [25] and builds navigation graphs for large and diverse environments with indoor and outdoor terrains. While, for the point cloud as a digital-twin, we train a deep neural network that predicts the locomotion cost connecting two nodes using a point cloud as an input. Here, we estimate the navigation costs for multiple edges in parallel by exploiting batch-processing on GPU.
These pre-built navigation graphs can be stored in the robots' memory. While operating in the real world we use a graph search method to find a path between any two points within a second. However, the real-world environment is dynamic and there can be large map changes that can completely block some paths. To handle such adverse situations, we propose a real-time navigation graph update that relies on the point cloud acquired by Anymal's LiDAR sensors to find alternative paths.

1.1 Main Contributions

We present overview of the thesis in Fig. 1.2. Overall, the main contribution for this thesis are:

1. A controller aware fast navigation graph building on a triangular-mesh with a GPU aided simulation

2. A neural network-based cost predictor to estimate locomotion costs between two points using point-cloud as an input
3. Real-time global replanning using point cloud acquired using robots LiDAR sensor to handle large map changes
4. Demonstrating the planning capabilities of the pre-built navigation graphs in the real-world digital twin environments spanning multiple floors. We also demonstrate the effectiveness of our real-time global replanning algorithm to find alternative paths in the events of paths planned using navigation graphs are blocked

1.2 Thesis Organization

Thesis is organized in the following way

- **Chapter 2**: We discuss the past research in the direction of global path-planning in robotics. We convey the limitations of some of these methods and motivate the need for new method development for our digital-twin-based planning.
- **Chapter 3**: In this chapter, we discuss navigation graph generation by simulating the Anymal robots on the triangular mesh in the Isaac Gym simulator. We show path planning results in a simulated construction environment and on the triangular of the HPH Building in ETH Hoenggerberg campus.
- **Chapter 4**: In this chapter, we introduce our method to generate a navigation graph from the Point Cloud-based digital twins. We present the training pipeline for our navigation cost predictor and show results on the point clouds acquired using the SLAM method on simulated environments and the real-life large-scale point cloud of HPH Building in ETH Hoenngerberg campus. We also show how our trained network can be used for real-time global path-replanning in case of large map changes.
- **Chapter 5**: We conclude our results and propose further direction to improve upon our method.

Chapter 2

Related Work

To plan a path in an environment the robot needs to have an understanding of the terrain and its own capabilities. Previously, the global path planning problem has been tackled by building topological or semantic maps, using the RL-based method to learn the goal-directed policy, using the pre-trained navigation cost predictors, or pre-building the navigation graphs for a specific environment. In this chapter, we discuss the research in global path planning from these perspectives and put forward our method as a valid alternative for large-scale, multi-terrain environments.

2.1 Topological Maps

Topological maps are built by clustering semantic or metric information of the environments. These higher-level maps allow robots to do efficient planning through clustered regions of the map. In his seminal work, Sebastien Thrun [26] described an approach in which the topological maps are extracted by finding critical lines on the Voronoi diagram on 2D occupancy grids. Since then, building topological graphs has become much autonomous [4, 5, 27] and semantically richer [28, 29]. In [4] authors iteratively build convex free-space clusters from the point cloud acquired by visual slam system and extract a topological map. While Gomez et al [5] build an autonomous mapping pipeline that connects local 3D sub-maps of different regions through a topological graph. In this, authors classify doors, hallways, and rooms and use this information to connect different clusters of the topology. In their recent work, Chaplot et al [27] explore a combination of real-time topological map building and learned policy for goal-oriented path planning. Topological maps can also store semantic information, which allows the robot to do intelligent decision-making. In [28], authors used multi-modal place classification of different indoor areas such as office, hallways, meeting room, etc. to build a semantic map of an indoor environment. Similar to this authors in [29] builds semantic maps on top of topological maps using Sum-Product Networks (SPNs).
While topological maps allow us to do efficient planning, they often have no information of the underlying terrain. Furthermore, creating topological maps require us to classify regions in free space vs occupied space, which is difficult for the legged robots, since they can step over many obstacles. Most of the work in topological mapping focuses on indoor environments, while we are interested in planning in indoor, outdoor, or mixed environments. Topological maps for such environments require much of the manual effort in parameter tuning and these parameters hardly transfer to new environments. Also, since topological maps are built on either metric or semantic information, resulting graphs are hardly aware of the robot's navigation capabilities.

2.2 RL based navigation

Another way to do path-planning is to directly learn a planning policy for global navigation. If the robot has access to the environment representation and the target goal position, it can use a higher-level navigation policy to decide its next waypoint along the path. Such policy can be either trained using reinforcement learning in simulation [30, 31, 18] or through memory augmented from the real-world experience [17, 21, 19]. In [30] authors proposes an algorithm that aggregates goal-directed long-range value function for any given state in the map through the iterative aggregation of short-term rewards. Further, authors of [31] extend this idea to relatively larger environments by applying network proposed in [30] on multiple abstract level representation of the environment. Application of RL in a large-scale environment has been shown in [18], where the authors proposed a policy that learns higher level navigation actions on a city scale map through RL in Google Maps simulation. However, such end-to-end RL navigation policies struggle with sparse rewards and hardly scales to a large-scale realistic environment for the robots. Another direction of work in RL-based navigation tries to combine sub-goal graphs with the local navigation policies. In such a case, the environment is represented by the organization of acquired experience in the form of sub-goals for the navigation agents. For example, authors in [17] used a memory graph obtained from the experience coupled with a trained waypoint retrieval network as a topological memory, which was used to extract next waypoint for the goal-oriented path planning. In another very similar approach [19], authors represent observations in a replay buffer in form of a memory graph and use the goal-conditioned value function to find the next waypoint on this graph. In [21], authors organized images acquired during exploration of unknown environments in form of local subgoals and used a next waypoint predictor network to find the next sub-goal for the visual-based navigation agent.

Another form of RL-related work in planning involves inverse RL (IRL) or imitation learning (IL) to learn adequate navigation cost function from the expert demonstrations. During the planning phase, this learned cost function can be used by the navigation agent to avoid risky areas and find a safe path to the goal. For example, authors in [15] authors used boosted Maximum Margin Planning (MMP) algorithm, to map the features to the navigation costs, where this mapping is learned via expert planning demonstrations. A very similar approach using functional gradient technique [16] to plan a safe path for the Unmanned Ground Vehicles (UGVs) from the satellite images. Authors in [32] presented a deep IRL method that maps the LiDAR observations of the autonomous car to the cost function.

These methods allow an interesting use of learning algorithms for end-to-end navigation. However, they are rather limited to small-scale environments or simple planning environments such as open outdoor where planning problems might just consist of avoiding some obstacles. For large-scale 3D environments with multi-DOF robots like Anymal, these methods are not yet adequate.

2.3 Learning Navigation Costs

For long-range planning of legged robots through difficult terrain, we can use navigation cost estimate combined with a sampling-based planner. Navigation costs can be estimated through typical terrain characteristics such as slope, roughness, and steps such as in [33]. In this paper authors combined this estimated navigation costs with RRT* [34] planner for navigation through rough terrains. Navigation costs can also be learned for specific robot types by machine learning, where supervised data are generated through simulation. In [35], authors used CNN on

height-map grids to learn the navigation costs of the respective grids. Learned motion costs allow planning of hybrid driving-stepping locomotion. Guzzi et al [14] train a navigation cost predictor using the height-map data and the labels acquired in a self-supervised fashion and use it in a sampling-based planner for the long-range path planning problems. However, the sampling-based planner needs to evaluate motion costs for each edge as they are sampled, which puts their applicability for real-time path planning in question. Bowen et al [36] alleviate this problem in which they introduce grid-based, GPU-aided, parallel roadmap construction method that relies on the motion cost network trained in a simulation.
These methods have been applied to complex legged robots for global path-planning through unstructured terrains. However, they rely on height-map for learning the motion costs. However, the 3D environments that we are interested in are multi-floor and span indoor as well as outdoor which can have a lot of overhanging structures. Such terrains can not be represented by heightmap, and more detailed data structures such as point cloud or triangular mesh are needed for an accurate representation.

2.4 Navigation graphs using a Digital-Twin

If we have access to the digital twin of the environment, we can build a navigation graph beforehand that can be used by the robot while doing real-time planning. In one such work [37], authors first assign either a trotting or a walking controller to each point in a point cloud by thresholding the maximum curvature in its neighborhood and its normal direction. Further, the authors reconstruct the meshes of this annotated point cloud and assign a controller to each polygon by majority voting procedure. With these navigation meshes, authors plan a path where the robot chooses to either walk or trot within a particular polygon. This method is purely based on the heuristic do not necessarily focus on the learning-based controller which does not necessarily follow any particular gait patterns.
More detailed use of digital-twin was proposed by Faust et al [22], where they construct a probabilistic roadmap of the environment in a simulation. For the roadmap generation, first, they train a local obstacle-avoidance policy that does not have access to the global map. Using this local policy to connect the nearby nodes in PRM, authors generate a navigation graph that is aware of the robot's controller capabilities and scale to large environments as long as we have the digital twin. Our approach was mainly inspired by this method because it allows us to use digital-twin effectively. However, the authors have only tested in the planner environments with the obstacle in between and without any terrain variability. Also, the approach was only shown for the indoor environment using a 2D building plan as a digital twin and requires to train a local navigation policy.
On the contrary, our approach differs from this in multiple ways. First, we do not train a local obstacle avoidance policy. We take a pre-defined control policy and rely on our PRM graph to avoid obstacles. Second, the environment we are focusing on has varied terrains where the robot needs to make the decisions like, which terrains are traversable and which are not instead of just avoiding obstacles. Furthermore, we demonstrate our approach on large-scale digital twins spanning multiple floors and with outdoor and indoor terrains. We also show a real-time global replanning method by modifying the edge costs of the navigation graph in real-time using the point cloud data acquired from robots' LiDAR sensors.

Chapter 3

Navigation Graphs in Simulation

3.1 Motivation

We are interested in building navigation graphs that allow the Anymal robot to exploit its true capabilities. For example, as shown in Fig. 1.1, for most of the wheeled robots, the smooth path via road (in orange) will be faster, but for anymal, the path through vegetation (in blue) can be faster given that the controller is able to walk on such terrains. To make such decisions efficiently while planning a path between two points in the map, the robots need to have access to a pre-built navigation graph. These navigation graphs can be built using a digital twin of the environment. Specifically, with the triangular mesh of the terrain, we used NVIDIA's Isaac Gym [25] simulator to build navigation graphs. Isaac Gym allows to replicate the real-world physical interaction between anymal and the terrain and simulate thousands of anymal robots in parallel on a single GPU. Authors in [38] already demonstrated these capabilities by training a walking policy for Anymal in Isaac Gym within minutes and transferring it to the real-world terrains. In this chapter, we demonstrate how parallel simulation capabilities of the Isaac Gym can be used to build terrain and controller-aware navigation graphs for the Anymal.

3.2 Methodology

We use a variant of PRM to build navigation graphs. PRM has two main advantages: First is that PRM can generate the roadmap offline. While doing planning we only need to do a shortest-path search via algorithms like Dijktras, A*, Bellman-Ford, etc. This allows for efficient path-planning in real-time. Second, while building PRM we can exploit the parallel simulation capabilities of the Isaac Gym. For example, authors in [36] used parallel batched-edge predictions to build a grid-based roadmap, while we use parallel robot simulations to build PRMs for large-scale maps. The pseudo-code for our PRM variant is shown in Alg 1
There are three main steps involved in building PRM.

3.2.1 Sampling

Usually, the mesh has a highly uneven distribution of the triangles, as a result of different meshing techniques. First, we apply mesh decimation in the Meshlab software [39] to reduce the number of triangles in a mesh (Fig. 3.2). This results

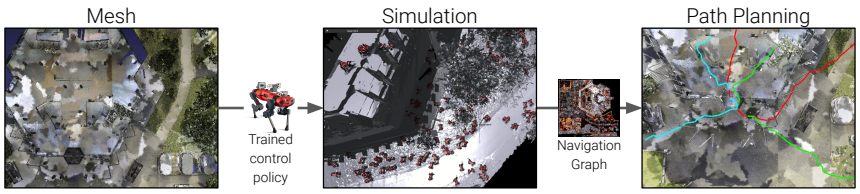

Figure 3.1: Simulation based navigation cost

Algorithm 1 Parallel PRM building in Simulation

Require: n_{init}, n_{final}, $\pi_{trained}$, B

1: $V \leftarrow Sample(n_{init})$ ▷ Sample n_{init} nodes initially
2: $E \leftarrow Connect(V, V)$
3: $G \leftarrow (V, E)$
4: **while** $V \leq n_{final}$ **do**
5: **if** $E \leq B$ **then**
6: $V \leftarrow V \cup Sample(n)$ ▷ Sample fixed number of new nodes
7: $E \leftarrow E \cup Connect(n, V)$
8: **else if** $E \geq B$ **then**
9: $e_{batched} \leftarrow Batch(B)$
10: $c_{batched} \leftarrow Simulate(e_{batched}, \pi_{trained})$ ▷ Simulate using fixed policy
11: $G \leftarrow Update(G(V, E), c_{batched})$
12: **end if**
13: **end while**

in more triangles for the difficult terrains such as stairs and fewer triangles for the flat terrains like a floor. The centroids of these triangles act as a sampling pool.

Sampling happens in two stages: First, we sample n_{init} nodes from the pool with sampling probability proportional to the area of the triangle. After this, we update the probabilities of all the nodes in the pool by node density within a fixed radius. So, while sampling in the other stages (line 6 in Algo.1), we use these updated probabilities to sample new nodes. The graph building process runs until a total number of nodes in the graph reaches predefined n_{final}.

As shown in Fig.3.2, the nodes in the sampling pool might lie on the vertical wall and on the trees. While doing simulation these nodes can be automatically removed since the robot will not be able to stand there. However, to reduce the simulation burden we use rejection sampling to reject the nodes for which the normal of the triangle is too far from the vertical direction.

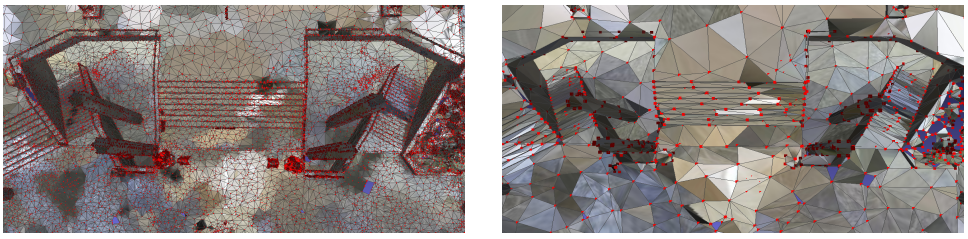

Figure 3.2: Sampling nodes

3.2.2 Node Connectivity

The second step for PRM is to establish connectivity between nearby nodes. In PRM this is usually done by connecting all nodes within a certain radius. However, the anymal robot has a specific footprint and it does not make sense to connect nodes that are within its footprint. For this reason we define r_{min} and r_{max} and connect nodes which are farther than r_{min} but within r_{max} radius.

3.2.3 Cost Estimate

Each edge in our variant of PRM has a cost associated with it. While planning robot uses this cost within a graph search algorithm to find the shortest path between two nodes. In our case, we monitor 4 different costs for each edge: success rate, distance, time, and energy consumption.

We model our graph G as a directed roadmap. To estimate the cost for any directed edge, we do s simulations of anymal traversing that particular edge. For each simulation, the robot starts from the starting node with yaw orientation facing the target node and traverses along the edge using a P-control for the linear and angular velocity commands (Fig.3.3). We also add white noise to the initial pose and orientation of the anymal such that the cost estimations are robust. Furthermore, anymal uses a pre-trained controller which generates actuation commands from velocity commands and other observations. These actuation commands are converted into torques using either a PD-controller or a trained actuator network.

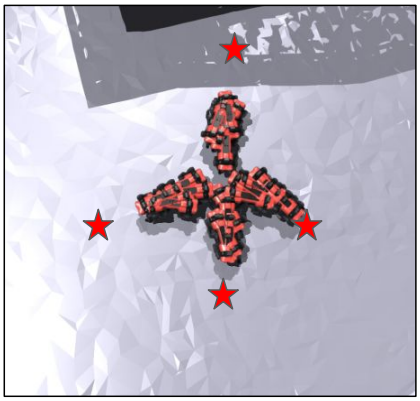

Figure 3.3: Parallel Simulation

The whole simulation process happens in Isaac Gym, which allows seamless integration between physical simulation of the robots and actuation commands prediction using RL-trained policy [25]. To use parallel processing of the GPU in Isaac Gym, we batch B number of edges ($e_{batched}$) and evaluate these costs in parallel by simulating the robots as described above (video: B.3). After s anymal simulation for each edge, the four different cost metrics are averaged across iterations. After batched cost estimates ($c_{batched}$), we add them into underlying graph $G(V, E)$.

3.2.4 Shortest Path Search

Pre-built navigation graphs help anymal to plan a path while traversing the real world. To do this, however, we need to assign a single cost to each edge. For the purpose of this thesis, we focus on a cost which is a combination of traversal success rate (i.e. safety) and the distance.

$$c_e = \alpha * f(c_{safety}) + \beta * g(c_{distance}) \tag{3.1}$$

where, f, g are real-valued functions and α, β are the relative weights.

To plan a path between any two positions, we first prune the whole graph based on a certain safety threshold. We find the closest nodes on the navigation graph for the requested positions in the real world and do the shortest path search using the Bellman-ford algorithm on the pruned graph. Only if the path search is unsuccessful, we further decrease the safety threshold for the pruning and redo the path search.

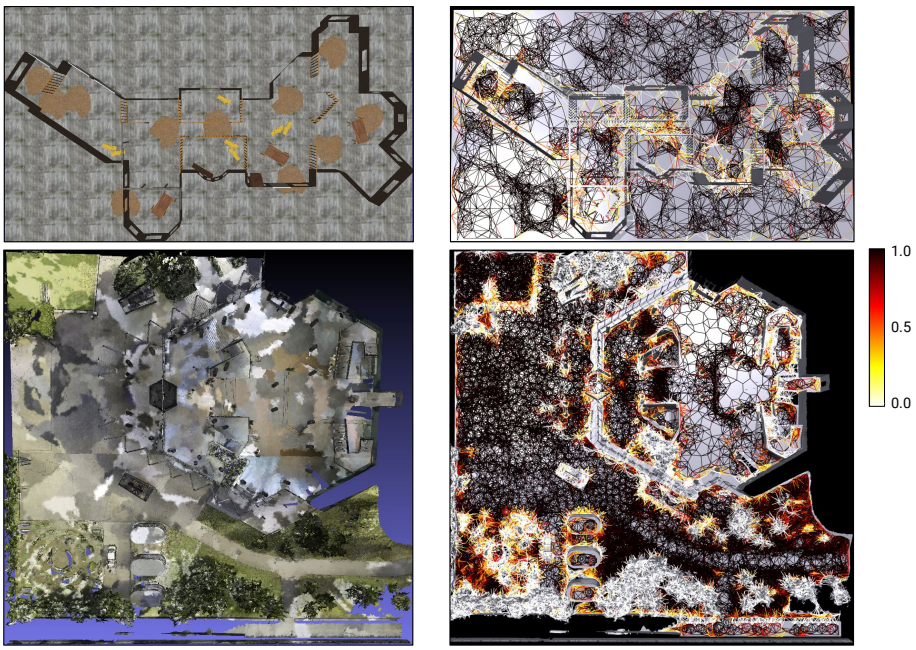

Figure 3.4: Navigation graphs for construction office environment (top) and ETH HPH building (bottom)

This two-stage method allows us to find a safer path always and in case of no safe path is available, the user can be instructed about possible places where the robot might fail.

3.3 Results

We show the results of simulation-based navigation graphs for indoor office construction environment proposed in [40] as well as the real-world mesh acquired of the HPH building in the ETH Hoenggerberg campus. The office construction environment consists of a narrow passage, multiple rooms, and different types of obstacles. This indoor environment has a size of around 25m x 40m. On the other hand, the real-world mesh of the HPH building was acquired using hand-held laser scanners. This scanned environment is 100m x 100m in size and consists of multiple floors, stairs, indoor halls, and outdoor terrains with various roughness and slopes. This environment provides a realistic condition for anymal operations.

3.3.1 Navigation Graphs

In Fig. 3.4, we present the navigation graph built using the method described in the previous section. The navigation graph for the office consists of 2k nodes and 20k edges, while the navigation graph for the HPH building consists of 10k nodes with 120k edges. Each edge color represents a probability of successful traversal along that edge for the Anymal robot, and the probability scale is shown in the right of Fig. 3.4. As we can see in the office environment, the edges only pass inside the room through the open doors, as well as the robot has some successful edges passing through the narrow corridor. For the HPH mesh, all outdoor flat ground is traversable with very high probability. Further, the robot also has edges passing through the door going inside the building and going down or up the stairs. Also, we can see that node density is much higher for difficult terrains, such as the stairs,

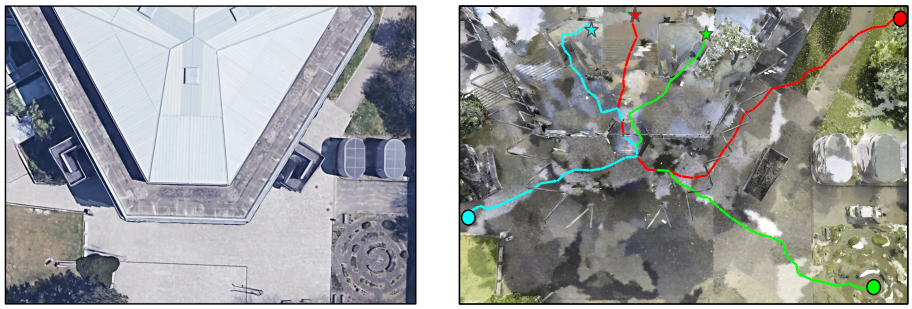

Figure 3.5: Planned Paths for ETH HPH building

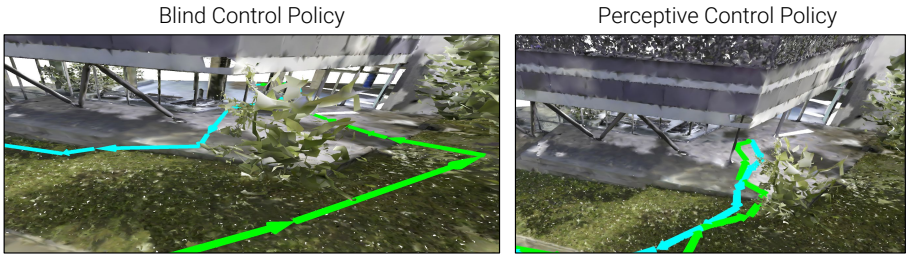

Figure 3.6: Paths going inside and outside the building using a Blind (left) and Perceptive (right) policy

rock garden in the left bottom, and vegetation. The higher density in the difficult terrains ensures that we find a safer path through such regions.

3.3.2 Planned Paths

With access to these navigation graphs, the robot can plan a path between any two locations in the real world. In Fig. 3.5, we show multiple paths planned for the ETH-HPH building. On the left of Fig. 3.5 is the real image of ETH-HPH building from Google Earth, and on the right is the mesh of the same area with the roof section removed for better visibility. Starting points for the paths are shown in the circles with the goal positions shown with the stars. These planned paths are ∼100m in length and pass through difficult terrains: e.g., the green path finds a safer path to get out of the rock garden, all paths pass through the open door and traverse through the stairs to a different floor. Also, assuming that the environment is static these planned paths can be reliably traversed using a P-controller for robots' linear and yaw velocity commands (video: B.3).

Since the simulation uses pre-trained policy as an input (Alg. 1), the paths planned using simulation-based navigation graphs are aware of the robot's control policy. For example, in Fig.3.6, we show the paths planned using the policy that does not use height information during the simulation (left), and the policy that uses the heights (right). With the blind policy, while going inside the building anymal chooses a path where it doesn't need to step up (green), since the blind policy usually fails in such a scenario. However, while going out of the building anymal can step down and hence chooses a shorter path (blue). On the other hand, with the perceptive policy, the anymal can step up and step down the stairs, and hence both the paths are relatively the same.

3.3.3 Run-times

The run-time for the graph building is proportional to the number of edges in the graph and scales linearly with the higher GPU memory. For example, the run-time for ETH-HPH mesh with 120k edges is 6h30m on NVIDIA-RTX-2060Ti GPU with 11GiB memory. However, the graph building process happens offline and needs to be repeated only if there are large environment changes. During the online phase only the navigation graph needs to be stored in the robot's memory.
The path-planning time is proportional to the edge density in the graph and implementation of the shortest-path search algorithm. In this thesis, we used Bellman-Ford algorithm from the networkx python package [41] for path-planning. The average run-time across 50 random path search on ETH-HPH navigation graph was 0.72s.

Chapter 4

Navigation Costs from Point Cloud

4.1 Motivation

In Chapter 3, we built navigation graphs by simulating the robots to estimate traversal cost between any two nodes in PRM. However, for the simulation to be reliable the meshing needs to accurately represent all the terrains and need to have continuous surface topology without holes or bumps. However, creating such accurate meshing of the real-world environment is difficult and the resulting mesh usually has some noise.
On the other hand, the underlying point cloud which is used for the meshing is easy to acquire. Point Cloud of the real-world environment can be either acquired using hand-held LiDAR scanners or by manually operating the robots in the environment and building a map. Furthermore, point clouds can be easily extended by merging with the point clouds acquired during different times. This allows us to continuously extend the environment by remapping and adding new areas. Also, the point cloud information is available in real-time for anymal robots via their LiDAR scanners. This point cloud information can be used to find alternative paths in case of large map changes (e.g. blocking of one entrance to the building, road blocked due to construction). Hence, using point cloud to estimate navigation costs is a good alternative to simulation-based costs. In this chapter, we introduce a supervised training method that predicts the navigation costs from the input point cloud.

4.2 Methodology

We use the same method for building PRM as discussed in Chapter 3, with the only difference being in step-3 (3.2.3), where we use a deep neural network (DNN) to predict navigation costs using point-cloud instead of doing a simulation on the mesh. In this section, we describe the training data generation pipeline and the architecture used for training the navigation cost predictor.

4.2.1 Training Data Generation

To predict the navigation costs for an edge in PRM, we use the point cloud sample along the edge as an input to the network. To generate input data and ground truth samples, we rely on simulated terrains. We simulate the anymal robots in these terrains and estimate the safety (success rate) for anymal traversability between two points on the map. The terrains used in simulation and corresponding costs

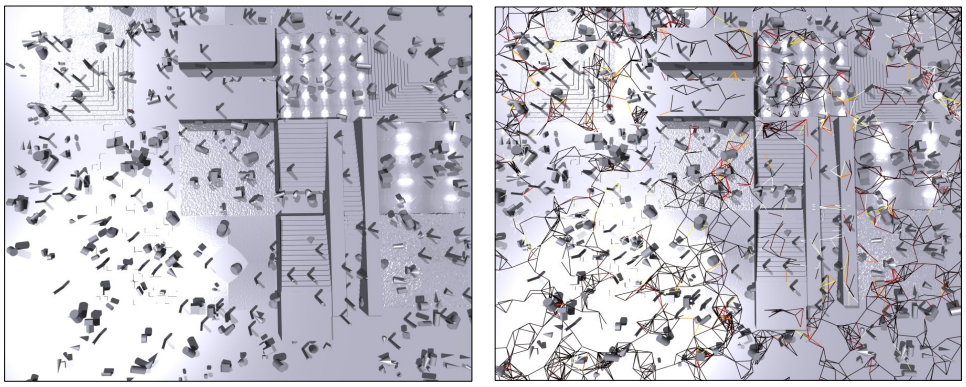

Figure 4.1: Simulated terrain (left) and training cost labels (right)

for different traversal paths are shown in Fig. 4.1. Our simulation environment consists of varied terrains, such as sloped, rough surfaces, steps of different sizes, and narrow corridors. Moreover, it also includes surface and overhanging obstacles of various sizes and shapes. This detailed environment allows us the navigation cost predictor to learn diverse scenarios the robots need to traverse.

Each edge in Fig. 4.1, represents one sample for the network. To pass the network enough information about the underlying terrain of an edge, we fit a rectangular box with the height and width proportional to anymal's footprint and oriented along the length of the edge as shown in Fig. 4.2. All the points enclosed within this box consist of a sample that is passed to the neural network for cost prediction. We use the box with the cross-section of 2m x 2m, hence the sampled point cloud within this box might contain overhanging obstacles, which the robot can pass through.

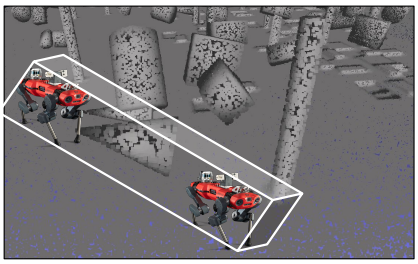

Figure 4.2: Training Sample

4.2.2 Network Architecture and Training

To learn a traversability from the point cloud sample, we need specialized architectures that operate on an unordered point-set. The previous work on applying deep learning to unordered point-set data mostly focuses on shape classification and segmentation [42, 43, 44, 45]. We exploit architectures developed for the point cloud classification purposes to estimate anymal's traversability. Specifically, we explore PointNet++ [42] and Dynamic Graph CNN (DGCNN)[43]. PointNet++ is a hierarchical neural network, which extracts the point cloud features at multiple levels. Through learning in the metric space this network is able to learn local features of the point cloud with increasing contextual scales. While DGCNN applies graph convolution to the unordered point-set learning problem. In this work, authors dynamically build graphs at different feature levels by exploiting feature space proximity. Graph convolution on these dynamically built graphs allows learning not just in a metric space but also in the feature space, and the network is able to learn local topology as well as a semantic understanding of the point-set. In our work, we apply DGCNN [43] for the navigation cost prediction.

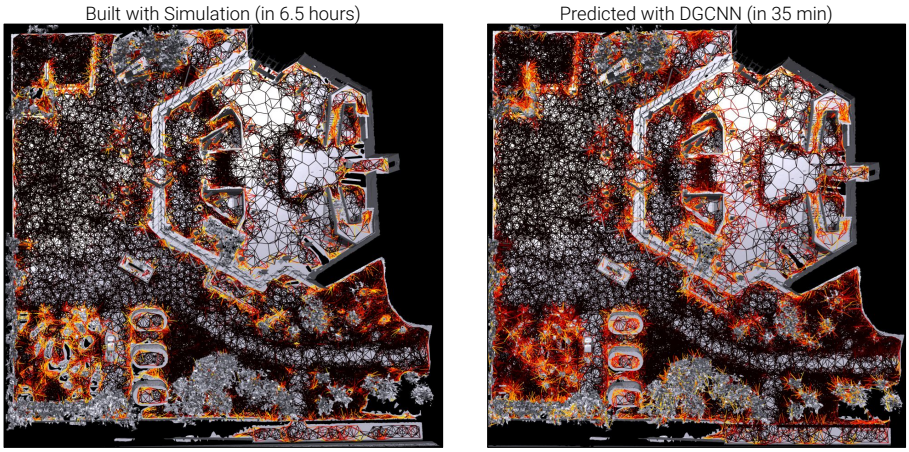

Figure 4.3: Comparison of Costs

To train DGCNN we use the data-generated using procedure described in Section 4.2.1. For each point cloud input, we first randomly sample 1048 points and pass them to the network. DGCNN is trained with $k = 60$ nearest neighbors and with an embedding dimension of 512. The output of the network is the success probability of a particular edge, which is regressed using a binary-cross-entropy (BCE) loss. The network is trained with a batch size of 16, using Adam optimizer with 0.01 learning rate. These hyperparameters for training are chosen by 70-30 train-validation split of the simulated data.

4.3 Results

In this section, we present the navigation graphs built for the ETH-HPH building, but we only use the point cloud of the building. To sample the point cloud from the mesh we use Poisson sampling to uniformly sample points on the mesh. Furthermore, we add a small noise to the resulting point cloud to replicate realistic point cloud acquisition. To acquire the point cloud for the office construction environment, we import this environment in Gazebo along with a full stack of the Anymal-C [1] robot. We map the environment using the onboard LiDAR-based mapping pipeline of the robot. Hence, this acquired point cloud has noise that a typical point cloud acquired by the robot will have.

4.3.1 Navigation Graph

In Fig. 4.3, we present the navigation graph using the costs predicted by DGCNN and compare it with the costs estimated by the simulation. For the better visibility of both the graphs, we only show the edges with a success probability greater than 0.3. Costs predicted by the DGCNN correspond well with the simulation-based costs, as can be seen by comparing both graphs. It is important to note that while training DGCNN, we did not use any data from this environment and the network was only trained using the environment shown in Fig. 4.1. Hence, our training procedure shows great generalization capabilities to the new obstacle types. Furthermore, our network handles the noise in the mesh well, such as in the rock garden where we predict more successful edges over the holes in the mesh, while simulation fails there because the robot can not walk through the holes.
Furthermore, the navigation graphs can be built order of magnitude faster with the point-cloud-based navigation cost predictor. The simulation of the costs took

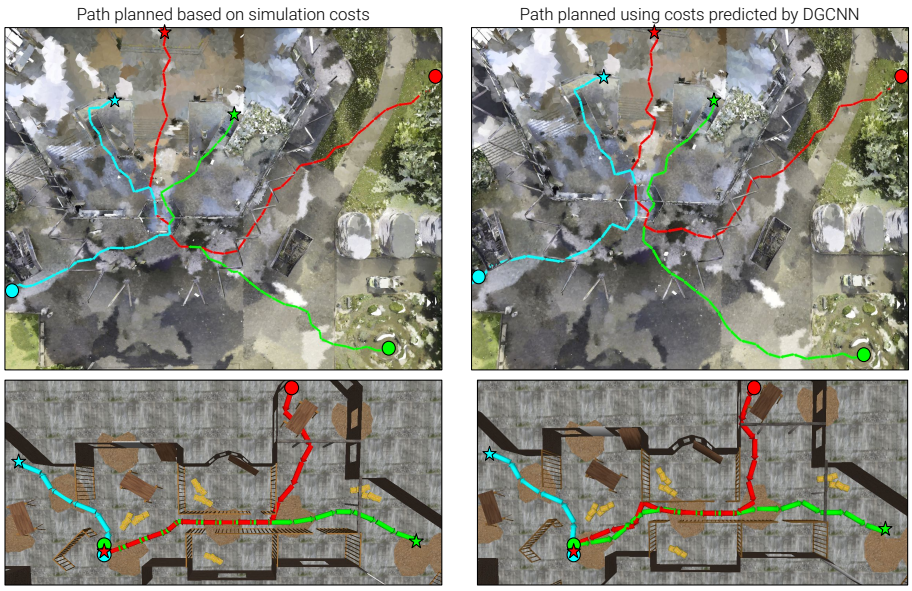

Figure 4.4: Planned Paths comparison for ETH-HPH building (top) and office construction environment (bottom)

around 6h30m on NVIDIA RTX 2060Ti GPU with 11GiB of memory. While all navigation costs are predicted within 35m on the same GPU. The mean absolute error (MAE) rate of navigation costs compared with the simulation-based cost is 7.45% on the scale of 100.

4.3.2 Planned Paths

We compare the path planning using the simulation-based navigation costs and point cloud-based navigation costs in Fig. 4.4. For the ETH-HPH environment, planned paths using both methods are almost identical. Since the point cloud for HPH building was acquire by sampling on the mesh, it did not have too much noise and hence the planned paths using predicted navigation costs compare well with the paths planned using simulation-based costs (Fig. 4.3). However, for the office construction environment, the point cloud was acquired by manually operating the robot in this environment in Gazebo, and acquiring the point cloud by onboard mapping pipeline. Since this generated point cloud is noisier, the planned paths differ a bit. For example, the red path planned in simulation-based graph finds a safer path through under the table, such that it can avoid the table legs. While in the case of point cloud the planned path pass too close to the wall. Nevertheless, the simulation of the Anymal in Gazebo shows that the robot is able to follow both paths successfully.

4.3.3 Real-time Global Path-Replanning

The pre-built navigation graphs allow efficient planning, but they are usually static. The paths planned using such navigation graphs are fixed. However, the real-world environment may observe large map changes (for example one path from the stair is blocked, or a door entry to the room is closed), in which case the robot might need to find an alternative path. It is important to make a distinction between large map changes and static/dynamic obstacles. Static or dynamic obstacles in the paths can be avoided by taking a small detour along the path. Such obstacle avoidance can

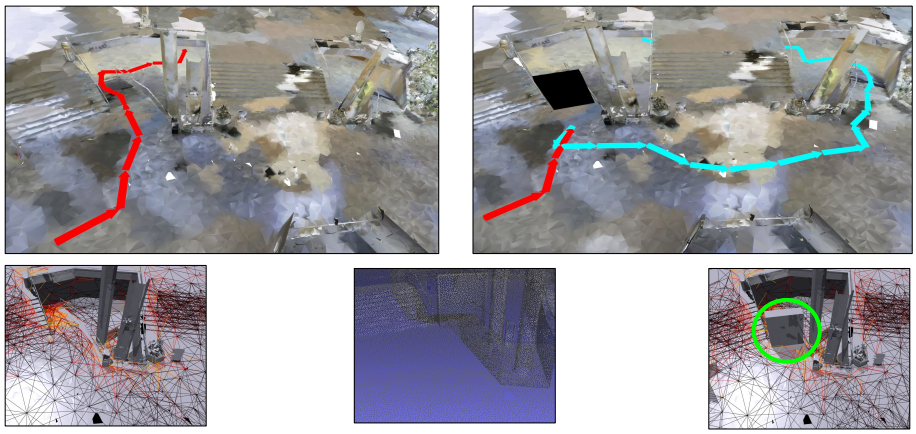

Figure 4.5: Real-time global replanning using point cloud information

be learned in the simulation and be applied to real-world scenarios [46]. However, we are focused on large static changes, where one path to the goal is completely blocked and the robot needs to do replanning through a completely different path. In this section, we simulate one such scenario of global path-re-planning.
In Fig. 4.5 (top-left) shows a path planned through the stairs using a pre-built navigation graph (Fig. 4.5 bottom-left). In one scenario, the entry to the stairs is completely blocked by a big obstacle in the path, and the robot needs to find an alternative path. Such cases can not be handled by local obstacle avoidance planners, since the robot needs to have access to the global graph to plan an alternative route. Here, we propose the use of a point-cloud-based navigation cost-predictor. For example, the point cloud in the robot's vicinity can be acquired by onboard LiDAR sensors (bottom-middle). We use this point cloud to re-evaluate all the edges around the obstacle and the navigation graph is updated with the new costs (bottom-right). With this updated navigation graph the robot replans a path through different stairs (top-right).
We also show the real-time replanning experiment for the office construction environment in Gazebo in this video B.3.

4.3.4 Run-times

The point-cloud-based cost predictor allows us to build navigation graphs 10x faster than the simulation. On the NVIDIA-RTX-2060Ti GPU with 11GiB of memory, we are able to do 96 edge cost predictions in a second on an average. These parallel predictions can be increased with higher GPU memory and by increasing the batch size for the prediction. For the global path-replanning, we re-evaluate the costs of all the edges around the current robot location. The overhead of acquiring point-cloud data, processing it to pass through the network, and prediction in total adds 3s at each way-point for the global replanning.

Chapter 5

Conclusion and Directions

5.1 Conclusion

In this thesis, we proposed two methods for building navigation graphs using either a mesh or a point-cloud-based digital twin. We demonstrated a PRM building algorithm that relies on parallel simulation of the Anymal robots in Isaac Gym and builds graphs for multi-floor environments within hours. With these pre-built graphs, the Anymal can plan a path between any two points within a second. Furthermore, the simulation allows us to discover different navigation costs and we can plan a path for a specific application (e.g. minimum time vs maximum safety). We also proposed a training pipeline for the point-cloud-based navigation cost predictor. With the navigation cost predictor, we can build graphs 10 times faster than the simulation-based approach. Furthermore, it allows us to do real-time global replanning using the point cloud to handle large, static map changes. With the Anymal robot experiments, we showed planning capabilities of our methods in simulation as well as real-world digital-twin.

5.2 Future Directions

While building navigation graphs by simulating Anymal robots on a mesh, we need to assign a friction coefficient to each triangle. Real-world terrains such as vegetation, gravels, slopes can have varied friction properties. Assigning the right friction values to a different area in the mesh can lead to better navigation graphs and close-to-optimal path planning. Such terrain properties can be learned via self-supervised learning by acquiring the real-world experience for the Anymal's traversals[47]. Furthermore, the paths planned on the pre-built navigation graphs are not necessarily smooth as can be seen in Fig 4.4. While traversing such paths robot might have to rotate many times. Path smoothing methods such as one proposed in [36], can help in reducing unnecessary rotation of the Anymal while following a path.
We find that to train out point cloud-based cost predictor we need diverse simulated terrain and obstacles data. With better training data generation and improved architectures, we can increase the confidence in the predictions of the navigation costs. Also, the global replanning adds 3s overhead at each waypoint. This can be reduced by look-ahead replanning, where we update the future edge cost while traversing the previous edge.

Bibliography

[1] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Blösch *et al.*, "Anymal-toward legged robots for harsh environments," *Advanced Robotics*, vol. 31, no. 17, pp. 918–931, 2017.

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

[3] G. Singh and J. Košecká, "Acquiring semantics induced topology in urban environments," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3509–3514.

[4] F. Blochliger, M. Fehr, M. Dymczyk, T. Schneider, and R. Siegwart, "Topomap: Topological mapping and navigation based on visual slam maps," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3818–3825.

[5] C. Gomez, M. Fehr, A. Millane, A. C. Hernandez, J. Nieto, R. Barber, and R. Siegwart, "Hybrid topological and 3d dense mapping through autonomous exploration for large indoor environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9673–9679.

[6] S. Niijima, R. Umeyama, Y. Sasaki, and H. Mizoguchi, "City-scale grid-topological hybrid maps for autonomous mobile robot navigation in urban area," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2065–2071.

[7] A. C. Murtra, E. Trulls, O. Sandoval, J. Pérez-Ibarz, D. Vasquez, J. M. Mirats-Tur, M. Ferrer, and A. Sanfeliu, "Autonomous navigation for urban service mobile robots," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4141–4146.

[8] F. Ruetz, E. Hernández, M. Pfeiffer, H. Oleynikova, M. Cox, T. Lowe, and P. Borges, "Ovpc mesh: 3d free-space representation for local ground vehicle navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8648–8654.

[9] 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*, vol. 38, no. 1, pp. 68–84, 2021.

[10] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *2006 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2006, pp. 2276–2282.

[11] P. Karkowski and M. Bennewitz, "Real-time footstep planning using a geometric approach," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1782–1787.

[12] R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, and A. Giusti, "Learning ground traversability from simulations," *IEEE Robotics and Automation letters*, vol. 3, no. 3, pp. 1695–1702, 2018.

[13] 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 2021)*, 2021.

[14] 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.

[15] N. Ratliff, D. Bradley, J. A. Bagnell, and J. Chestnutt, "Boosting structured prediction for imitation learning," 2007.

[16] N. D. Ratliff, D. Silver, and J. A. Bagnell, "Learning to search: Functional gradient techniques for imitation learning," *Autonomous Robots*, vol. 27, no. 1, pp. 25–53, 2009.

[17] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," *arXiv preprint arXiv:1803.00653*, 2018.

[18] P. Mirowski, M. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, A. Zisserman, R. Hadsell *et al.*, "Learning to navigate in cities without a map," *Advances in Neural Information Processing Systems*, vol. 31, pp. 2419–2430, 2018.

[19] B. Eysenbach, R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," *arXiv preprint arXiv:1906.05253*, 2019.

[20] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Scaling local control to large-scale topological navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 672–678.

[21] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine, "Ving: Learning open-world navigation with visual goals," *arXiv preprint arXiv:2012.09812*, 2020.

[22] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5113–5120.

[23] H.-T. L. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust, "Rl-rrt: Kinodynamic motion planning via learning reachability estimators from rl policies," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4298–4305, 2019.

[24] "skydio 3d scan," https://www.skydio.com/3d-scan, 2021.

[25] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[26] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[27] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 875–12 884.

[28] A. Pronobis, O. Martinez Mozos, B. Caputo, and P. Jensfelt, "Multi-modal semantic place classification," *The International Journal of Robotics Research*, vol. 29, no. 2-3, pp. 298–320, 2010.

[29] K. Zheng, A. Pronobis, and R. Rao, "Learning graph-structured sum-product networks for probabilistic semantic maps," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[30] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," *arXiv preprint arXiv:1602.02867*, 2016.

[31] D. Schleich, T. Klamt, and S. Behnke, "Value iteration networks on multiple levels of abstraction," *arXiv preprint arXiv:1905.11068*, 2019.

[32] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[33] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1184–1189.

[34] 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.

[35] T. Klamt and S. Behnke, "Towards learning abstract representations for locomotion planning in high-dimensional state spaces," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 922–928.

[36] B. Yang, L. Wellhausen, T. Miki, M. Liu, and M. Hutter, "Real-time optimal navigation planning using learned motion costs," in *IEEE International Conference on Robotics and Automation (ICRA 2021)*, 2021, p. 699.

[37] M. Brandao, O. B. Aladag, and I. Havoutis, "Gaitmesh: controller-aware navigation meshes for long-range legged locomotion planning in multi-layered environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3596–3603, 2020.

[38] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *5th Annual Conference on Robot Learning*, 2021.

[39] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool," in *Eurographics Italian Chapter Conference*, V. Scarano, R. D. Chiara, and U. Erra, Eds. The Eurographics Association, 2008.

[40] "Clearpath robotics gazebo worlds," https://github.com/clearpathrobotics/cpr_gazebo, 2020.

[41] "Netowrkx," https://github.com/networkx/networkx, 2004.

[42] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[43] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[44] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6411–6420.

[45] N. Sharp, S. Attaiki, K. Crane, and M. Ovsjanikov, "Diffusion is all you need for learning on surfaces," *arXiv preprint arXiv:2012.00888*, 2020.

[46] D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter, "Learning a state representation and navigation in cluttered and dynamic environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5081–5088, 2021.

[47] 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.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Appendix A

Topological Map

At the very beginning of the thesis, we attempted to do global navigation via topological maps for the indoor and outdoor environment. In this section, we described our approach for topological mapping. For topological map building, we received the point clouds for one outdoor building and one indoor floor. We followed these steps.

- To build a topological map means the clustering of the regions. For clustering, we first need to identify which points to cluster. For our experiment, we first extracted the ground by fitting multiple planes with +Z axis normal to the point cloud data and chose the ones that have fitted points above a certain threshold. Hence, we only take care of the multi-floor environments through multiple plane fittings.
- Usually, the ground points are very large in number, and clustering of all such points can be slower. Hence, we first downsample the ground points and use them as the set of locations we need to cluster.
- For clustering we first extracted the features of the selected location. We use points XYZ position, principal components of the visible points from each location, and RGB values.
- For clustering, we attempted multiple algorithms, such as KMeans, DBSCAN, Region Growing [48]. We found the region growing clustering to be optimal. We had to hand-tune the parameters of the algorithm and the number of clusters by inspection of the extracted clusters.

In Fig. A.1, we show the topological clusters extracted for the outdoor building and the indoor floor. As we can see the clustered points do make sense in some cases, however sometimes the clustering merges points that are at different heights, and finding good clusters which can be used by the robots for navigation requires big effort in parameter tuning. Also, to start the clustering process, we need to first identify the ground, which is a difficult process in itself if we do not have good prior of the ground regions for a specific environment. Furthermore, extracted clusters do not take into account the robots' capabilities, and they are the same for wheeled robots as well as legged robots such as anymal. For controller-aware navigation graphs, topological maps are rather limited.

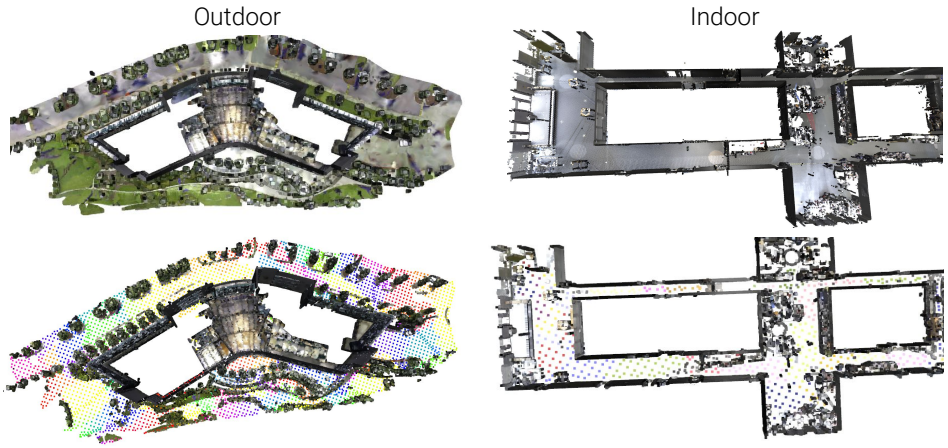

Figure A.1: Topological clusters extracted for the outdoor (left) and indoor (right) environments

Appendix B

Extras

B.1 Meshing Errors

Noise in the meshes results in wrong simulation costs. We found these noises (Fig. B.1) in the ETH-HPH mesh and going forward these need to be addressed to build reliable navigation graphs.

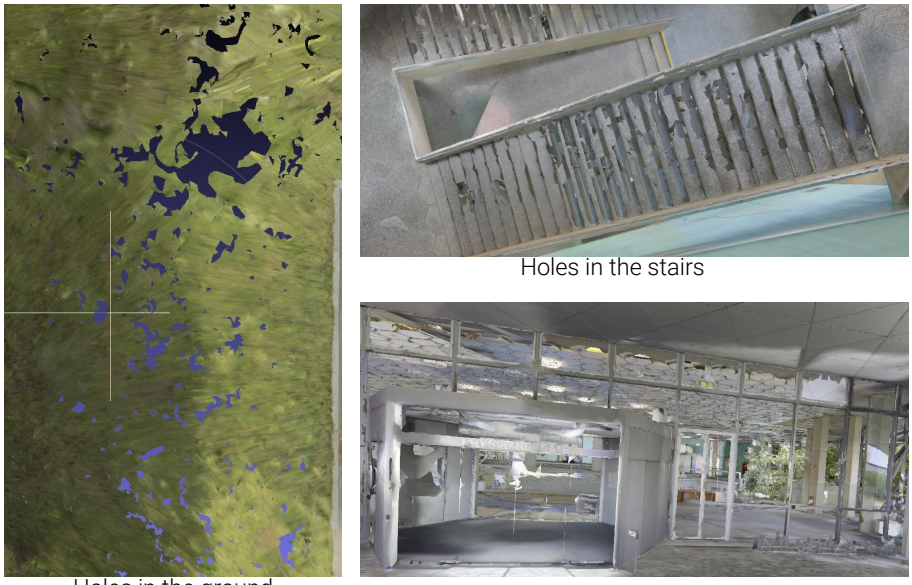

Figure B.1: Noise in the meshes

B.2 Useful Mesh Processing

There are many functionalities in the meshlab that have been used throughout the thesis for mesh processing.

- **To merge multiple meshes**
  Import into Meshlab $\longrightarrow$ *Filters* $\longrightarrow$ *Mesh Layer* $\longrightarrow$ *Flatten Visible Layers*
- **To reduce the number of triangles**
  Import into Meshlab $\longrightarrow$ *Filters* $\longrightarrow$ *Remeshing* $\longrightarrow$ *Quadratic Edge Collapse*

- **To increase the number of triangles**
  Import into Meshlab $\longrightarrow$ *Filters* $\longrightarrow$ *Remeshing* $\longrightarrow$ *Midpoint Sub-division*
- **Sampling points on Mesh**
  Import into Meshlab $\longrightarrow$ *Filters* $\longrightarrow$ *Sampling* $\longrightarrow$ *Poisson-Disk Sampling*

B.3 Videos

Here are some sample videos of our methods

- **Anymals building graph in Isaac Gym**
  https://youtu.be/jlqAoUXQBqw
- **Anymal following a path**
  https://youtu.be/rn2AD-OPwTk
- **Anymal replanning in Office construction environment in Gazebo**
  https://youtu.be/6fgLKVx3HPA

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Global Path Planning in a Digital-Twin

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Ajaykumar	Unagar

Student supervisor(s)

Marko	Bjelonic
Lorenz	Wellhausen
David	Hoeller
Joonho	Lee

Supervising lecturer

Marco	Hutter

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date

Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.