Hybrid Motion Planning and Control for Legged-Wheeled Robots: Application to Walking Excavators

Edo Jelavić

Diss. ETH No. 2899

DISS. ETH NO. 28999

Hybrid Motion Planning and Control for Legged-Wheeled Robots: Application to Walking Excavators

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

Edo Jelavić

M. Sc. in Electrical Engineering and Information Technology, ETH Zurich

born on 27.02.1991

citizen of Croatia

accepted on the recommendation of

Prof. Dr. Marco Hutter
Prof. Dr. Emilio Frazzoli
Dr. Steve Tonneau

2023

Robotic Systems Lab
Institute for Robotics and Intelligent Systems
ETH Zurich
Switzerland

Abstract

While slowly finding their way into human-engineered environments, deploying robots in natural environments remains challenging today. Automation is especially lacking for large-scale hydraulic machinery, which would be indispensable for automating dangerous tasks such as natural disaster responses.

For robotic autonomy, motion planning plays an important role, especially in the presence of obstacles. Overcoming obstacles requires adapting locomotion strategy to the surrounding terrain, a pattern that can be observed in humans and animals. Humans walk on two limbs but can use all four if the situation requires so. The terrain around the robot imposes constraints on limb placement, stability, and contact timing, and accounting for all constraints in a single motion planning problem is demanding. Traditionally, the problem is decomposed into smaller subproblems using simplified models and heuristics, which often cannot capture the coupled dynamics in the system. Hence they often plan motions not fully utilizing the robot's capabilities. Treating the robot as a whole becomes especially important for complex systems such as legged-wheeled robots.

This dissertation extends the locomotion capabilities of legged robots, emphasizing legged excavators. It develops motion planning algorithms utilizing all degrees of freedom for overcoming challenging terrain. We formulate the motion planning problem in a general way for multiple robot types and explore concepts for solving it. To this end, optimization and randomized sampling play a central role in computing global, whole-body motions presented in this thesis.

We work towards increasing legged robotic mobility in a series of five publications. We start by deploying a hydraulic 12-tonne-legged machine in a natural forest for precision harvesting. The mapping, localization, planning, and control systems have been proposed and integrated into an autonomous harvesting solution. The whole-body planner developed later in the thesis is inspired by the shortcomings of the planning system deployed in the forest.

A local optimization-based planner is used to cope with many degrees of freedom on a legged excavator. The proposed terrain-aware planner can compute efficient driving and stepping motion and utilize the arm for locomotion. To allow more flexibility, we only command goal poses and do not stipulate what to do in between, thus giving the optimizer complete freedom.

To cope with the non-convexity of the planning problem, we employ randomized sampling. We shift some of the computation offline in the form of pre-computed roadmaps, which help keep the planning times low. The initial whole-body plan is found by randomized sampling; however, it may still violate some physical system constraints (e.g., wheel rolling constraints). The initial motion plan is then fed to the optimization for refinement. The optimization ensures constraint satisfaction, while the initial plan keeps the optimizer away from bad local minima. Together they compute smooth, global, whole-body plans. We have tested this method on a legged excavator, a legged-wheeled robot, and a legged robot with point feet.

Lastly, a control system for legged excavators is developed. Whole-body motion plan execution is a well-studied topic for more miniature robots such as quadrupeds. However, hardware deployment on full-size hydraulic machinery is lacking. We run the optimization in a receding horizon fashion, which helps to combat drift and tracking errors. The terrain adaptive control system allows the planner to plan on simplified geometries and then adapt to unseen landscapes at execution time. Planning on simplified geometries is essential for deployment since a very accurate map cannot be obtained with current sensors. Small details like steps and roughness produce non-smooth gradients that hamper the optimization convergence.

Keywords: Robotics, Legged Excavator, Motion Planning, Model Predictive Control, Forestry, Sampling, Optimization, Whole-Body Planning.

Zusammenfassung

Während sie langsam ihren Weg in von Menschenhand geschaffene Umgebungen finden, bleibt der Einsatz von Robotern in natürlichen Umgebungen eine Herausforderung. Vor allem bei großen hydraulischen Maschinen, die für die Automatisierung gefährlicher Aufgaben, wie der Bewältigung von Naturkatastrophen, unverzichtbar wären, fehlt es an Automatisierung.

Für die Autonomie von Robotern spielt die Bewegungsplanung eine wichtige Rolle. Die Überwindung von Hindernissen erfordert eine Anpassung der Fortbewegungsstrategie an das umgebende Terrain, ein Muster, das bei Menschen und Tieren beobachtet werden kann. Der Mensch geht auf zwei Gliedmaßen, kann aber alle vier benutzen, wenn es die Situation erfordert. Die Umgebung des Roboters stellt Anforderungen an die Positionierung der Gliedmaßen, die Stabilität und das Timing des Kontakts, und die Berücksichtigung aller Anforderungen in einem einzigen Bewegungsplanungsproblem ist anspruchsvoll. Traditionell wird das Problem in kleinere Teilprobleme zerlegt, wobei vereinfachte Modelle verwendet werden, die die gekoppelte Dynamik des Systems oft nicht erfassen können. Daher werden oft Bewegungen geplant, die die Fähigkeiten des Roboters nicht voll ausschöpfen. Die Betrachtung des Roboters als Ganzes ist vor allem bei komplexen Systemen wie Robotern mit Beinen und Rädern wichtig.

Diese Dissertation erweitert die Fortbewegungsmöglichkeiten von Robotern mit Beinen, wobei der Schwerpunkt auf Schreitbaggern liegt. Es werden Algorithmen zur Bewegungsplanung entwickelt, die alle Freiheitsgrade nutzen, um schwieriges Gelände zu überwinden. Es wird untersucht, wie das Bewegungsplanungsproblem allgemein für mehrere Robotertypen formuliert werden kann und welche Konzepte zur Lösung des Problems beitragen können. Optimierung und zufälliges Sampling spielen eine zentrale Rolle bei der Berechnung der in dieser Arbeit vorgestellten globalen Ganzkörperbewegungen.

Wir setzen eine hydraulische 12-Tonnen-Maschine mit Beinen in einem natürlichen Wald zur Präzisionsernte ein. Die Systeme zur Kartierung, Lokali-

sierung, Planung und Steuerung wurden entwickelt und in eine autonome Erntelösung integriert. Der später in dieser Arbeit entwickelte Ganzkörperplaner wurde durch die Unzulänglichkeiten des im Wald eingesetzten Planungssystems inspiriert.

Ein auf lokaler Optimierung basierender Planer wird verwendet, um mit den vielen Freiheitsgraden eines Schreitbaggers fertig zu werden. Der vorgeschlagene geländeabhängige Planer kann effiziente Fahr- und Schrittbewegungen berechnen und den Arm für die Fortbewegung nutzen. Um mehr Flexibilität zu ermöglichen, geben wir nur Zielposen vor und legen nicht fest, was dazwischen zu tun ist, so dass der Optimierer völlige Freiheit hat.

Um mit der Nicht-Konvexität des Planungsproblems umzugehen, verwenden wir zufälliges Sampling. Wir verlagern einen Teil der Berechnungen offline in Form von vorberechneten Roadmaps, die helfen, die Planungszeiten niedrig zu halten. Der anfängliche Ganzkörperplan wird durch randomisiertes Sampling gefunden; er kann jedoch immer noch einige Beschränkungen des physikalischen Systems verletzen (z. B. Radrollbeschränkungen). Der ursprüngliche Bewegungsplan wird dann zur Verfeinerung in die Optimierung eingespeist. Die Optimierung stellt sicher, dass die Nebenbedingungen erfüllt werden, während der ursprüngliche Plan den Optimierer von schlechten lokalen Minima fernhält. Zusammen berechnen sie glatte, globale Ganzkörperpläne. Wir haben diese Methode an einem Schreitbagger, einem Roboter mit Rädern auf Beinen und einem Roboter mit Punktfüßen auf Beinen getestet.

Schließlich wird ein Steuerungssystem für Schreitbagger entwickelt. Die Ausführung von Ganzkörper-Bewegungsplänen ist ein gut erforschtes Thema für kleinere Roboter wie Vierbeiner. Für hydraulische Maschinen in voller Größe wurden diese bisher jedoch nicht auf Hardware eingesetzt. Wir führen die Optimierung nach dem Prinzip des rückläufigen Horizonts durch, was dazu beiträgt, Drift- und Verfolgungsfehler zu bekämpfen. Das geländeadaptive Steuersystem ermöglicht es dem Planer, auf vereinfachten Geometrien zu planen und sich dann zur Ausführungszeit an nicht gesehene Landschaften anzupassen. Die Planung auf der Grundlage vereinfachter Geometrien ist für den Einsatz unerlässlich, da mit den derzeitigen Sensoren keine sehr genaue Karte erstellt werden kann. Kleine Details wie Stufen und Unebenheiten erzeugen nicht kontinuierliche Gradienten, die die Konvergenz der Optimierung beeinträchtigen.

Stichworte: Robotik, Schreitbagger, Bewegungsplanung, Modellprädiktive Regelung, Forstwirtschaft, Stichproben, Optimierung, Ganzkörperplanung, zufälliges Sampling.

Acknowledgments

I thank my advisors Prof. Marco Hutter, Prof. Emilio Frazzoli, and Dr. Steve Tonneau. Thank you Emilio for supporting my decision to switch groups and thank you, Marco, for accepting me at RSL and letting me start working on unique robotic hardware such as Menzi Muck M545. Over the years I watched it locomote over various stuff with excitement, sometimes hated it for not doing what I intended, and sometimes just being greasy up to my elbows when fixing it. I am grateful for the environment at RSL, where we could work freely, explore our own ideas, and collaborate with the industry. The forestry project was (in retrospect) my favorite and I have learned the most from it. Special thanks go to Simo Gröhn from Silvere for his help. Big thanks go to people from Intel and Blueriver-tech for making big machines do cool acrobatic moves together: Vladlen Koltun, Lee Redden, Mathias Müller, Ramitha Sundar, Harikrishnan Suresh, and Paul Montgomery. Special thanks to our students Cliff Li and Pierre-Jean Lorentz.

Thanks to the Menzi Team: to Dominic Jud who taught me ROS and always patiently explained to me why something was a bad idea to do on the platform. To Simon Kerscher and Phillip Leeman for keeping the machine going despite all the abuse it received from my (crude) software running on it. To Gabriel Hottiger for showing me the treacherous waters of C++ templating and teaching me about the Swiss internet culture. To Martin Wermelinger for teaching me ROS and point cloud processing. To Pascal Egli for sharing knowledge about Reinforcement learning, bike repairs, and ski touring. Thanks to all the other members of the Menzi Team: Julian Nubert, Lorenzo Terenzi, Burak Çizmeci, Fang Nan, Grzegorz Malczyk, Tom Lankhorst, Ryan Johns, and Koen Krämer. Besides working hours in the LEE H316 office and trenches of our testing field at Hönggerberg, we had a great time together hiking, ski-touring, making bonfires, or grilling.

Many thanks to RSL and ASL people outside of the Menzi Team. To

Maria Trodella, for always being positive, helpful, patient, and efficient in solving problems on the fly (even if it called for unorthodox methods). To Shehryar Khattak for the great discussions and tutoring on mapping. To the rest of the RSL and ASL crew with whom I shared drinks, skied, hiked, set stuff on fire, choked on spicy sauces, ran SOLA, or collaborated together: Klajd Lika, Johannes Pankert, David Höller, Marco Tranzatto, Vassilios Tsounis, Eris Sako, Michael Spieler, Giorgio Valsecchi, Marko Bjelonic, Konrad Meyer, Hendrik Kolvenbach, Joonho Lee, Nikita Rudin, Jan Preisig, Kaixian Qu, Jan Carius, Dario Bellicoso, Farbod Farshidian, Alexander Winkler, Lorenz Wellhausen, Perry Franklin, Maria Vittoria Minniti, Yves Zimmerman, Abel Gawel, Renaud Dube, Fadri Furrer, Christian Lanegger, Rik Bähnemann, Thomas James Stasny, Jen Jen Chung, Zachary Taylor, Nicholas Lawrance, and Karen Bodie.

Big thanks to all my students, some of whom later became my colleagues. Special thanks to Kaixian Qu, for his diligent work on our last paper together.

Thanks to my friends who helped me enjoy life outside of pursuing a doctoral degree: Matija, Pietro, Gianluca, and Filip. To my partner Helen, for her support, and encouragement not to quit. Thank you for the travels, love, fun, and many adventures together. Lastly, thanks to my family, my parents Alma and Hrvoje, and my brother Ivor for their support, belief in me, and many overnight train packages with delicious homemade food that reminded me of home.

Edo Jelavić, Spring 2023

Financial Support

This thesis was supported by the Swiss National Science Foundation (SNSF) as part of project No.188596, by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme grant agreement No 852044 and through the SNSF National Centre of Competence in Digital Fabrication (NCCR dfab). Additionally, this thesis has partly been funded by Silvere, a Finnish company working in the forestry sector.

Preface

This thesis is a *cumulative* dissertation and, as such, includes a selection of the most relevant publications created by the author during his doctoral studies. The thesis is organized as follows. Chapter 1 introduces the research topics, provides an overview of the state of the art, outlines the technical approach, and describes the contributions made in this thesis. Chapters 2 to 5 contain the individual scientific articles published in peer-reviewed journals or conference proceedings. Chapter 6 includes recently submitted work, which is under review at the time of writing. Each article is accompanied by *Lessons Learned* section offering developer insights. Chapter 7 summarize the achievements and provides potential directions for future research.

Contents

Notation

Unless otherwise stated, we use the following convention for mathematical symbols: Scalars are denoted in light (s), vectors in bold ($\mathbf{v}$), and matrices in uppercase bold ($\mathbf{M}$). For functions we use a similar notation referring to the output of the function: Light for scalar functions ($f(\cdot)$), and bold for vector-valued functions ($\mathbf{f}(\cdot)$).

Acronyms

AA	Abduction/Adduction
ATV	All Terrain Vehicle
BVP	Boundary Value Problem
CAD	Computer Aided Design
CAN	Controller Area Network
CDRM	Contact Dynamic Roadmap
CoM	Center of Mass
CoP	Center of Pressure
CPU	Central Processing Unit
CS	Contact Schedule
DBH	Diameter at Breast Height
DDP	Differential Dynamic Programming
DMS	Direct Multiple Shooting
DoF	Degrees of Freedom
ECC	Earliest Contact Creation
EE	End-Effector

EKF	Extended Kalman Filter
EoM	Equations of Motion
FE	Flexion/Extension
FOV	Field of View
FSS	Full-Support State
GBP	Graph Based Planner
GDP	Gross Domestic Product
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HBC	Hip Balancing Controller
HEAP	Hydraulic Excavator for Autonomous Purpose
HFE	Hip Flexion-Extension
HO	Hierarchical Optimization
HP	Hermite Parametrization
ICP	Iterative closest point
ID	Inverse Dynamics
IK	Inverse Kinematics
IMU	Inertial Measurement Unit
LCB	Latest Contact Break
LF	Left Front
LH	Left Hind
LIDAR	Light Detection And Ranging
LPC	Locomotion PC
LSTP	Long Short-Term Motion Planner
MIP	Mixed-Integer Program
MPC	Model Predictive Control
MPP	Motion Planning Problem
NLP	Nonlinear Program

NN	Nearest Neighbor
NO	Numerical Optimization
NPC	Navigation PC
NTP	Network Time Protocol
OCP	Optimal Control Problem
ODE	Open Dynamics Engine
OMPL	Open Motion Planning Library
OSM	One-Step Motion
PCA	Principal Component Analysis
PD	Proportional-Derivative
PRM	Probabilistic Roadmap
PTP	Precision Time Protocol
QP	Quadratic program
RBDL	Rigid Body Dynamics Library
RF	Right Front
RH	Right Hind
RL	Reinforcement Learning
ROS	Robot Operating System
RRT	Rapidly-Exploring Random Tree
RS	Reeds-Shepp
RT	Real-time
RTK	Real-time Kinematic
SBP	Sampling Based Planning
SDF	Signed Distance Function
SLAM	Simultaneous Localization and Mapping
SLQ	Sequential Linear Quadratic
SP	Support Polygon
SQP	Sequential Quadratic Programming
TELE	Telescopic
TO	Trajectory Optimization

UAV Unmanned Aerial Vehicle
UGV Unmanned Ground Vehicle
USB Universal Serial Bus
VIO Visual Inertial Odometry
VMC Virtual Model Controller
WBC Whole Body Controller
ZMP Zero Moment Point

1

Introduction

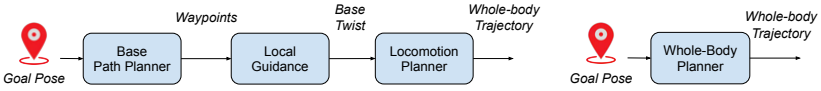
We are witnessing increasing mobile robot usage in various areas, e.g., warehouse and transportation automation. While successfully deployed in structured environments, operations in unstructured environments such as forests or construction site operations largely remain nonautomated. Automating deployments in unstructured environments is important since these environments often pose a danger to humans, e.g., natural disaster responses such as landslides or avalanche cleanups. Large machines like legged excavators are indispensable in these scenarios. However, the automation of large machines like legged excavators has received very little attention from the community compared to smaller platforms like quadrupeds, humanoids, or autonomous cars. We try to bridge this automation gap in this thesis.

Wheels remain the dominant locomotion strategy, especially for long mission times. While wheeled platforms can move fast, they cannot fundamentally negotiate challenging terrain. If we want to send robots outside of human-engineered environments, we should move away from purely wheeled robots and start deploying more complex systems such as legged or legged-wheeled robots (Bjelonic et al., 2019a; Hutter et al., 2016a). Legged-wheeled robots are an attractive solution since they combine the ability to traverse obstacles from legged systems with the efficiency of wheeled systems (Bjelonic et al., 2019a; Jud et al., 2021b; Kashiri et al., 2019).

Among other aspects, motion planning is one of the critical functionalities for autonomous operation. The motion planning system should be able to overcome challenging terrain by utilizing all the Degrees of Freedoms (DoFs) that a robot has to offer. On the other hand, we would like the planner to compute energy-efficient motion plans when the terrain is easy. A similar pattern can be observed in the natural world. Humans walk on two legs most

of the time, however we can use all four limbs to overcome steep terrain or obstacles.

Part of the reason we still struggle to find motions utilizing all DoFs is the use of simplified models (e.g., inverted pendulum). Simplified models constrain the system to types of motions where the model is valid which can prevent the planner from using the system's capabilities effectively. In contrast, utilizing whole-body planners often leads to discovering a richer set of motions as a whole-body planner can better utilize coupled dynamics inside a more complex model. Another culprit is using hierarchical structures, which often contain a high-level planner with a simplified model and a local/planner controller. The lower-level controller is often tracking base velocity twists. However, breaking down the motion planning problem results in pipelines with many points of failure (see Fig. 1.1a) and inherently limits the set of discoverable motions. A more flexible solution would be to give the goal pose to the planner and let it decide on the optimal trajectory, as shown in Fig. 1.1b.



(a) Hierarchical Planner **(b)** Whole-Body Planner

Figure 1.1: ***Left:*** Hierarchical planning structure often found in the literature. ***Right:*** Whole body planner

Motion planning for legged-wheeled systems is particularly challenging. Legged-wheeled robots combine the combinatorial nature of legged systems (computing the contact schedule) with the non-holonomic nature of the wheeled systems (rolling constraints). Introducing both legs and wheels increases the amount of DoFs that a planner has to handle. To cope with all DoFs and compute whole-body motion plans, we build a framework based on different prediction horizons. In particular, a Numerical Optimization (NO) technique is used for short-term planning with higher fidelity and randomized sampling for long-term planning with lower fidelity. As a testing platform, this thesis mainly uses a legged excavator, a legged machine with four wheeled legs, and an arm without wheels. However, the developed algorithms are also showcased on quadrupeds and wheeled quadrupeds.

The dominant strategy in the robotic community is to use NO for planning since it scales very well with the number of DoFs and can be run at high frequency. A great demonstration of NO capabilities are some of the acrobatic

maneuvers performed by ANYmal robot over challenging terrains in Winkler et al., 2018. NO produces motion plans involving jumps requiring precise coordination between the torso and the legs. Such behavior would be impossible to discover using simplified models or human-engineered heuristics. While an optimization-based planner can discover very complex motions (e.g., Mordatch, Todorov, and Popović, 2012), it typically is hard to utilize in the real world. The optimization problem becomes non-convex, especially when we introduce terrain constraints (e.g., collision constraints). Although NO ensures constraint satisfaction and physical motions, it needs a good initialization to avoid bad local minima. To provide a good initialization, roboticists often manually choreograph simplified motions and then refine them with optimization to ensure feasibility and smoothness. We can then compute motion libraries offline and track them at execution time. Bjelonic et al., 2022 is an example of such an approach. A similar approach has also been used by robots such as Spot or Atlas from Boston Dynamics (Boston Dynamics, 2022a,b,c) at the time of writing this thesis.

Unlike optimization, planners based on randomized sampling can deal with highly non-convex cost landscapes (Barraquand et al., 1997). Sampling Based Plannings (SBPs) perform well on terrains encountered in the real world and have been used with maps built from onboard sensory data (Wermelinger et al., 2016). The randomized nature of SBP can be beneficial for Contact Schedule (CS) discovery which is a complex combinatorial problem. However, SBPs suffer from the curse of dimensionality (Bellman, 1966), which hinders direct usage on systems with many DoFs. While successfully used in lower dimensional spaces such as *SE(2)*, achieving real-time planning for more complex systems such as legged systems is still an open problem. At their core, SBPs have two steps: sampling and connection of new sample to the tree. Efficiently connecting new samples to the tree remains an active area of research. A new sample connection is often made using simplified models to keep it computationally tractable Tonneau et al., 2018a. However, to use robots to their fullest in challenging terrains, we need to compute whole-body motion plans. This problem has been studied in the literature (e.g., Bretl, 2006; Escande, Kheddar, and Miossec, 2013), and it often yields long computational times, due to its complexity.

A widespread technique for motion planning is to break down the problem into sub-problems. A sampling-based planner uses a simplified model for guiding path computation; the local planner computes base velocities which are then tracked using a locomotion controller. While enjoying significant success in

field deployments with quadruped robots (e.g., Tranzatto et al., 2022a,b, such a paradigm does not utilize the complete robot's potential. We integrate higher fidelity models into the planning pipelines and optimize both motions and contact schedule. To summarize, this thesis aims to get the best of both worlds (randomized sampling and optimization). Combining the two approaches allows us to navigate complex terrains (thanks to SBP) and handle systems with many DoFs (thanks to NO).

1.1 Motion planning problem

In general form, the Motion Planning Problem (MPP) can be summarized with equations below:

$$\begin{aligned} &\min_{\boldsymbol{x},\dot{\boldsymbol{x}},\boldsymbol{t_F}} \int_0^{t_F} f(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t)\, dt \\ &\text{s.t.} \qquad \boldsymbol{x}(t=0) = \boldsymbol{x}_0 \qquad \boldsymbol{x}(t=t_F) = \boldsymbol{x}_F \\ &\qquad\quad \boldsymbol{g}(\boldsymbol{x}, \dot{\boldsymbol{x}}) \leq 0 \quad \boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = 0 \end{aligned} \tag{1.1}$$

where f is a planning objective, $\boldsymbol{g}$ is a set of inequality constraints, $\boldsymbol{h}$ represents the set of equality constraints. $\mathbf{x}(t)$ is a vector of decision variables, and t_F is the final time. The motion planning problem formulation is general enough that any high-level task can be described with Eq. 1.1. Thus, if we can solve Eq. 1.1 or report that a solution does not exist, we can solve motion planning on a general level.

1.2 Related Work

This thesis focuses on motion planning. However, deploying a robot in the field requires a great deal of work for building different submodules and integrating them into a fully functioning system, e.g., mapping, localization, state estimation, motion planning, and control. The related work on motion planning is divided into subsections. The first two subsections discuss different robot models and terrain representations. Subsequently, we discuss using NO for planning and then using SBP for planning. The last section describes hybrid approaches combining optimization and sampling to solve the MPP. Since contributions in some works are polyvalent, they are mentioned multiple times and discussed from different points of view.

The presented work is centered around legged excavators that share some common characteristics with other legged robots. Hence, the related work includes

many contributions from agile quadruped literature. We discuss motion planning for wheeled, legged, and legged-wheeled systems. Our contributions are summarized in Sec. 1.3.

1.2.1 Robot Models

The robot model is a mathematical abstraction that predicts the motion given control inputs and initial state. Models come in different levels of fidelity, e.g., simplified models like Zero Moment Point (ZMP), centroidal dynamics models, and kinematic models. The chosen model is often tailored to the particular robotic system under consideration. For example, for a legged excavator, there is little need for a dynamic model since the machine operates in quasi-static conditions. In this thesis, we rely on two open-source libraries implementing kinematics and dynamics for robotics: Pinocchio (Carpentier et al., 2019) and RBDL (Felis, 2017).

Simplified Models

For walking, one can abstract away the stance legs and the floating base of a legged robot as an inverted pendulum. The model captures the system's instability of bipeds and quadrupeds (Kajita et al., 2003; Kalakrishnan et al., 2010; Sardain and Bessonnet, 2004; Winkler et al., 2017). In some cases, a simple model has an analytical solution (Englsberger, Ott, and Albu-Schäffer, 2015). Widely used model for legged robots is the ZMP (Vukobratović and Borovac, 2004) and its derivatives (Holmes et al., 2006; Poulakakis and Grizzle, 2009). ZMP is an inverted pendulum model that accounts for contact legs position via the support polygon. Similarly to the dynamics, one can approximate the kinematics as well. For example, Sherikov, Dimitrov, and Wieber, 2015; Tonneau et al., 2018b; Winkler et al., 2018 approximates the robot's workspace with geometric shapes.

Simplified models allow for planning at high frequencies and simplify the MPP, leading to faster development. However, they only capture the dominant dynamics, which can be problematic in the presence of obstacles or uneven terrain. Furthermore, they do not allow full utilizing coupled dynamics that may exist in the system.

Kinodynamic Models

The Kinodynamic model considers kinematics and simplified dynamics; it is the model of choice in this thesis. Planning for a legged excavator uses a

kinodynamic model with full kinematics and limb's masses under quasi-static conditions (Chapter 3, 4, 5, 6). Examples of kinodynamic models for legged robots can be found in Farshidian et al., 2017a; Winkler et al., 2018. We compute motion plans for a legged robot in Chapter 6 using a floating base model with constant inertia (Farshidian et al., 2017a).

Another commonly used kinodynamic model is the centroidal dynamics model (Orin, Goswami, and Lee, 2013), which describes dynamics with linear and angular momentum equations while accounting for external torques and forces. For example, Sleiman et al., 2021 uses this model for a legged robot with an arm.

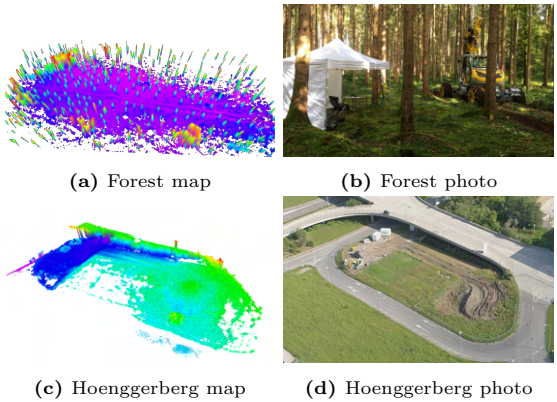
Full Rigid Body Dynamics

A full rigid body dynamics model treats each robot link as a separate rigid body. The model considers torques acting on the link through actuated joints and external forces. The full rigid body dynamics model has the highest fidelity of all aforementioned models. In Sentis and Khatib, 2006, the full rigid body has been used to compute optimal torques as a part of the feedback controller for a humanoid robot. We follow the same approach in Chapter 4 and 6 where we use a full rigid body dynamics model inside the Whole Body Controller (WBC) framework introduced in Bellicoso et al., 2016. Using the full rigid body dynamics model for planning remains challenging because of the computational cost associated with solving the Equations of Motion (EoM).

1.2.2 Environment Representation

Terrain information is essential in motion planning for any robot and directly influences the difficulty of the MPP. A suitable terrain representation should be easy to build from sensory data, robust to noise, and computationally cheap to query.

A 2.5D elevation map is one of the map representations that fit the requirements, and it has been used widely for legged and legged wheeled robots, notably in the early work Kweon et al., 1989. Examples shown in fig. 1.2. Nowadays, open-source libraries build elevation maps from raw sensory data, e.g., Fankhauser, Bloesch, and Hutter, 2018; Fankhauser and Hutter, 2016a; Miki et al., 2022b. The elevation map is the terrain representation of choice in this thesis. Adding raw elevation maps into the optimization-based planners can lead to a loss of convergence since raw data can have discontinuities. In Chapter 5, we discuss how to incorporate it into an optimization problem



(a) Forest map **(b)** Forest photo

(c) Hoenggerberg map **(d)** Hoenggerberg photo

Figure 1.2: Elevation maps and corresponding photos for forest environenment and our testing field at Hönggerberg in Zürich.

and retain optimization convergence. Other examples of using an elevation map with optimization-based planners can be found in Grandia et al., 2022; Jenelten et al., 2022.

An elevation map can capture complex obstacles and is computationally cheap to build and query. However, it cannot represent overhanging structures. In Buchanan et al., 2021, the authors propose using a double elevation map (one for ground, the other for overhanging structures) to overcome this problem. Alternatively, when planning with a legged robot, Gaertner et al., 2021 uses Signed Distance Function (SDF) to avoid the torso collision. While more versatile than an elevation map, SDF does not offer as high resolution (with today's computing power). The right choice of terrain representation ultimately depends on the application.

1.2.3 Numerical Optimization

This section reviews the use of NO for planning. We distinguish between Trajectory Optimization (TO), typically done offline, and Model Predictive Control (MPC) approaches where the system re-plans in the receding horizon fashion.

Trajectory Optimization

Planning both motion and CS using optimization is challenging; in particular CS optimization introduces lots of local minima. Hence, most planners proposed in the literature rely on externally provided CS, either a fixed cyclic

gait or a separate gait planner. This strategy is common for both legged and legged-wheeled robots. Examples can be found in Geilinger et al., 2018; Medeiros et al., 2020; Winkler et al., 2017, 2015. In Winkler et al., 2018, the authors add CS timings into the optimization problem as continuous variables, which is the approach we also explored for a legged excavator in Chapter 3.

Aceituno-Cabezas et al., 2017; Deits and Tedrake, 2014 formulated the MPP as a Mixed-Integer Program (MIP) where integer variables correspond to contact flags. Tonneau et al., 2020 experiments using L1 norm to approximate the integer variables. Alternatively, in the presence of *complementarity constraints*, the optimization can compute the CS without adding integer variables. *Complementarity constraints* are constraints of the form $\boldsymbol{f}^T \boldsymbol{d} = 0$, where $\boldsymbol{f}$ are contact forces and $\boldsymbol{d}$ is limb distance to the terrain; Dai, Valenzuela, and Tedrake, 2014; Posa, Cantu, and Tedrake, 2014 follow this approach. Mordatch, Todorov, and Popović, 2012 augments the optimization problem with additional contact indicator variables, which are activated if the motion violates the physics too much. Contact constraints can be incorporated into the system dynamics directly. In that case, the optimization does not need to be aware of contact phases. Carius et al., 2018a, 2019 add contact dynamics directly into the optimization.

While TO can optimize the motions, it struggles with optimizing the CS. Adding the CS decision variables into the optimization introduces many local minima, making the optimization problem vastly harder. In general, optimizing over the CS requires careful initialization or gradient shaping through cost function (Neunert, Farshidian, and Buchli, 2016; Neunert et al., 2018).

Model Predictive Control

Re-planning in an MPC fashion introduces robustness at execution time and helps combat model mismatch and state estimation drift. MPC with a single step prediction horizon was used to implement active suspension in space rovers (Cordes, Babu, and Kirchner, 2017; Giordano et al., 2009; Reid et al., 2016). MPC has also found application for biped robots (Powell, Cousineau, and Ames, 2015) or humanoids (Erez et al., 2013). In (Bellicoso et al., 2016, 2018b; Jenelten et al., 2020) legged robot continuously recomputes the motion plan using ZMP as stability criterion. The authors decouple motion planning for the robot's base and limbs. Subsequently, a similar approach has been extended to a legged wheeled robot (Bjelonic et al., 2019a; Sun et al., 2020; Viragh et al., 2019). While decoupled motion planning enables re-planning at

high frequencies (up to 200 Hz), the planner cannot account for uneven terrain and thus has reduced generality.

Early work of Farshidian et al., 2017c utilized differential dynamic programming to simultaneously optimize for both base and limbs of the legged robot. With more computing power available, more whole-body planners have been proposed (Brasseur et al., 2015; Grandia et al., 2022; Ishihara, Itoh, and Morimoto, 2019; Koenemann et al., 2015; Sleiman et al., 2021). Whole-body plans are beneficial since decoupling base and limb planning makes the motion plans less stable (Bjelonic et al., 2020a).

Optimizing in the presence of terrain constraints was long reserved for TO and offline planning methods. Recently, optimization problems with terrain constraints can be solved online in the MPC fashion (Gaertner et al., 2021; Grandia et al., 2021; Melon et al., 2021). The biggest challenge is keeping the computation times low and avoiding local minima. Some authors introduce offline motion libraries to aid the optimization at runtime (Bjelonic et al., 2022; Hauser et al., 2008; Melon et al., 2021). While we follow the same idea in Chapter 6 where we use MPC with terrain constraints to compute motion plans, we find good initialization using a separate planner.

The vast majority of MPC planners do not optimize over contact schedule since it introduces strong non-convexities. Instead, the contact schedule is either fixed to be cyclic Di Carlo et al., 2018; Grandia et al., 2022 or is computed by a separate gait planner (Bjelonic et al., 2020a). CS can be optimized to some extent using the bi-level optimization (Farshidian et al., 2017c; Li and Wensing, 2020). Bi-level optimization can refine the externally provided switching times between contact modes, extending or shortening contact phases, effectively changing the contact schedule. However, optimizing switching times can only make limited adjustments to the CS since the order of modes is fixed.

1.2.4 Sampling Based Planning

The fundamental limitation of optimization-based planners is that they can get trapped in bad local minima. Some researchers have tackled this problem by employing stochastic optimization (Mastalli et al., 2020), which is computationally expensive. Another option is to use a planner based on randomized sampling (Sampling Based Planning), which helps escape the local minima.

Majority of SBPs are based on Rapidly-Exploring Random Trees (RRTs) (Karaman and Frazzoli, 2011; LaValle and Kuffner Jr, 2001), Probabilistic Roadmaps (PRMs) (Kavraki et al., 1996) or Graph Based Planners (GBPs)

(e.g. A* Hart, Nilsson, and Raphael, 1968). Most of the motion planners proposed for legged robots (quadrupeds and humanoids) are based on RRTs (Geisert et al., 2019; Short and Bandyopadhyay, 2017; Tonneau et al., 2018a,b). Geisert et al., 2019; Tonneau et al., 2018a,b use RRT planner to sample base poses and compute a guiding path based on kinematic reachability. Foothold locations are computed by solving inverse kinematics, or failure is reported in case the planner can find no footholds. Short and Bandyopadhyay, 2017 also rely on RRT planner to sample base poses and introduces Contact Dynamic Roadmap (CDRM) to compute feasible footholds rapidly. Almost all related work plans statically stable motions for legged robots (humanoids or quadrupeds), e.g., (Bretl, 2006; Escande, Kheddar, and Miossec, 2013; Fernbach, Tonneau, and Taïx, 2018; Hauser et al., 2008).

Enforcing non-holonomic constraints inside SBPs or GBPs is problematic because of the inherent discrete nature of the planner. Hence, planning for legged-wheeled robots with SBPs or GBPs is relatively uncommon. In Chapter 2, we use Reeds-Shepp (RS) curves (Reeds and Shepp, 1990) to enforce minimal turning radius constraints. However, the planner presented in Chapter 2 considers only the pure driving mode and does not attempt to step. An A* algorithm was used in Klamt and Behnke, 2017, 2018 to navigate non-convex environments. The world is discretized in a grid, and the robot plans driving and stepping motions based on a carefully crafted cost function. Stepping relies on heuristics and is tailored for the Momaro robot, which can turn the wheels in place and drive sideways. Hence, it is not suitable for robots with minimal turning radius or non-steerable wheels in its current form.

Methods based on randomized sampling are much more suitable for CS discovery compared to NO methods as noted in Carius et al., 2022 where path integral sampling was used to escape the local minima. In Tonneau et al., 2018a, different contact schedules emerge based on which limb has to be grounded or which limb has to break contact. Similarly, we compute CS by reasoning about which limb is close to its kinematic limits.

1.2.5 Hybrid Methods

Hybrid methods combine different planners to gain the best of sampling and optimization. We follow the hybrid approach in Chapter 5 and 6. Combining sampling and optimization enables escaping local minima, computing global plans, and enforcing kinodynamic constraints, all of which are hard inside the SBP or SBP alone.

A straightforward way to compute global plans is to use a sampling-based or grid-search technique with reduced model order. Klamt and Behnke, 2017; Wermelinger et al., 2016 combine a reduced order global planner with a controller based on cyclic gait pattern. Tranzatto et al., 2022b; Wellhausen and Hutter, 2021 add a neural network to predict the terrain difficulty for the underlying tracking controller. Although this is a viable solution that has been tested in the field, the global planner still does not compute whole-body plans, and it cannot optimize over the contact schedule, thus possibly not utilizing all DoFs.

One can also connect sampled states by solving the Boundary Value Problem (BVP), which allows for whole-body planning. Ding et al., 2021; Fernbach, Tonneau, and Taïx, 2018; Hwan Jeon, Karaman, and Frazzoli, 2011; Kim, Kwon, and Yoon, 2018; Norby and Johnson, 2020; Xie et al., 2015 utilize such a strategy. BVP can usually be solved (fast enough) only for systems with few DoFs (e.g., a vehicle) or if the analytical solution exists. We follow the same approach in Chapter 2, where we treat our robot as a vehicle. Fernbach, Tonneau, and Taïx, 2018 compute transition feasibility between sampled for a humanoid robot by approximating centroidal dynamics and reducing the feasibility computation for solving a linear program. Similarly, Norby and Johnson, 2020 compute global plans for a quadruped by solving the BVP. Approaches based on solving the BVP are generally not (yet) suitable for real-time planning and so far have been shown only in simplified scenarios. Hardware results are generally lacking except in (Fernbach et al., 2020; Tonneau et al., 2018a).

SBP can be used to compute an initial plan which is then fed to the optimization for refinement. We follow the same idea in this thesis. The two-stage approach can be found in manipulation motion planning literature (Dai et al., 2018; Leu, Wang, and Tomizuka, 2022; Leu et al., 2021), humanoid motion planning (Li, Long, and Gennert, 2016) and ground vehicle motion planning (Li et al., 2021a). In Leu et al., 2021; Li et al., 2021a, an initialization is computed using grid-search and then optimized for a low number of DoFs (car, mobile manipulator). Instead of computing the initial plan from scratch, one can create an offline motion library (e.g., Bjelonic et al., 2022) and then feed it to the optimization at runtime for tracking. However, Bjelonic et al., 2022 is not a true hybrid planner since it relies on optimization to compute the motion library. While still limited to toy problems, advanced methods combining inference and second-order optimization (Layeghi, Tonneau, and Mistry, 2022) could be a promising future direction.

Using Reinforcement Learning (RL) policy instead of SBP has been proposed in the literature for foothold planning. In Tsounis et al., 2020, RL is used to plan footsteps which are then tracked by the RL based controller. On the other hand, a model-based tracking controller can also follow the footsteps of an RL planner (Gangapurwala et al., 2022), even if the tracking happens in the latent space (Li et al., 2021b). RL planning methods still require retraining for different types of terrains and cannot plan acyclic gait patterns. However, they look promising to use in the future.

1.3 Technical Approach and Contribution

This thesis tries to solve the following problem:

Compute whole-body motion plans for legged and legged-wheeled systems given the robot model, terrain information, starting point, and goal point.

We extend the capabilities of legged robots and heavy machines with legs in multiple ways. Firstly, we showcase an autonomous deployment of a full-size, 12-tonne heavy machine Hydraulic Excavator for Autonomous Purpose (HEAP) (Jud et al., 2021b) for precision harvesting missions in natural forests. Thus we open up the avenue for improving forestry efficiency, which is still based on almost 300 years old principles Von Carlowitz and Rohr, 1732. Having observed the forest deployment's shortcomings, we propose extending the locomotion capabilities with a new motion planner.

The proposed motion planner for legged excavators accounts for the machine's all DoFs. It precisely coordinates arm motions with the rest of the body, matching what only very skilled human operators can do. Furthermore, the planner has an intuitive behavior of driving whenever possible and stepping over obstacles when needed, exactly as a human operator would. With the newest planning developments, we bring heavy-legged machines closer to autonomous deployment away from human-engineered environments.

While HEAP deployment has demonstrated a vast potential for autonomous forestry work, it has also unveiled the limitations of hierarchical planning with simplified models. In the rest of the thesis, we follow the idea of motion planning with as few heuristics as possible and fully utilizing the robot's capabilities. This idea is reflected in the input to the system; we always command a goal pose and offer the planner freedom to decide on the optimal motion between the start and goal pose (structure shown in Fig. 1.1b). In contrast, many

proposed whole-body planners receive base velocities as input and require another local planner or human operator in the loop. Using the kinodynamic model, we compute whole-body motions, thus minimizing the usage of heuristics (e.g., switching between stepping and driving Klamt and Behnke, 2017). Furthermore, the calculated plans are global. All the complexity is offloaded to randomized sampling and optimization, which are the main two components in our approach.

The planner is divided into two steps. In the initialization step, the SBP computes a contact schedule and a sequence of whole-body states (full generalized coordinates and contact flags or all limbs). Subsequently, the optimization refines the initial solution. Whole-body states from SBP act as attractors for the optimization preventing it from falling into bad local minima. Thus, we can handle both complex terrains (thanks to SBP), many DoFs, and complex system dynamics (thanks to NO). Lastly, we showcase the generality of the developed solution by computing motion plans for a quadruped, a wheeled quadruped, and a legged excavator.

Lastly, motion planning using a very detailed map is problematic because some terrain features (e.g., roughness, vegetation) can render the gradients discontinuous. Hence it is desirable to plan on a simplified geometry and adapt to the terrain during the maneuver execution. In this thesis, we design a whole-body terrain-adaptive tracking controller for robots with heavy limbs. Unlike many mobile robots, heavy machines have actuators with low bandwidth and extremely non-linear behavior (hydraulics, friction). Thus choosing the control strategy needs to be done carefully.

The research objectives of this thesis can be stated as three questions below:

Q1 (optimization): *How can we formulate the motion planning problem into a discrete optimization problem? What is the right environmental representation for this?*

Q2 (initialization): *How to compute a good initial solution for the optimization? Can this be done quickly and efficiently?*

Q3 (control): *How can we execute motion plans on large machines with low bandwidth actuators? How to stay robust in the presence of unobserved terrain?*

Each of the papers below answers a subset of questions (Q1-Q3). Each article is accompanied by a summarizing paragraph putting it in the context with the rest of the thesis. The rest of this dissertation is organized as described in the *Preface*.

Paper I

Jelavic, E., Jud, D., Egli, P., and Hutter, M. (2022a). "Robotic Precision Harvesting: Mapping, Localization, Planning and Control for a Legged Tree Harvester". *Field Robotics* 2, pp. 1386–1431

This publication addresses the deployment of a legged-wheeled robot for precision tree harvesting in a natural forest. Precision tree harvesting removes some trees selectively while leaving neighboring trees intact. To our knowledge, this was the first time a full-sized machine was deployed in a natural forest. The robot operates in a confined, GPS-denied forest environment. As a result, this publication covers an extensive scope; it proposes mapping, planning, and control strategies and integrates them into a fully autonomous system. We highlight the aspects most relevant to this thesis, namely the ones that answer some of the questions listed above. Lessons from this deployment are used to improve the locomotion capabilities in the following papers.

We choose a multi-layered elevation map as a terrain representation since the minimal representation still enables us to carry out the mission, thus addressing parts of *Q1*. In every other paper that follows, we use the elevation map representation. The article also follows the approach of letting the algorithm synthesize motion plans instead of relying on heuristics. We merely give tree locations to the planner, which searches for an approach pose and a viable path in a single planning problem. The computed plan initializes the local path-following controller based on the pure pursuit algorithm (question *Q2*).

Lastly, we consider how to execute motions on a full-scale hydraulic machine (HEAP in our case) which falls under question *Q3*. To move the machine in the natural forest, we combine the terrain adaptation controller (Hip Balancing Controller (HBC)) with the path-following controller (pure pursuit). This combination allowed us to track motion plans, overcome small unmodelled obstacles (like stumps) and drive over muddy forest ground (thanks to even force distribution). All subsequent hardware deployments on the machine were equipped with a terrain adaptive controller.

In this paper, we do not compute whole-body motions; the machine merely drives using the legs as an active suspension system. Even during our short deployment, we saw that stepping and using an arm as support could be beneficial. The following papers strive to utilize the machine better and extend the locomotion capabilities.

Paper II

Jelavic, E. and Hutter, M. (2019). "Whole-body motion planning for walking excavators". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2292–2299

This publication addresses the problem of computing smooth local plans for a complex machine like a legged excavator. The platform of choice has 31 DoFs, legs, wheels, and an arm, making it a challenging test bed for the local planner. Naturally, we chose the Numerical Optimization as the backbone of our planner. We investigate how to formulate an optimization problem (question *Q1*) for a legged-wheeled robot with an arm and propose a direct method relying on Hermitian splines building on previous work Winkler et al., 2018. The resulting planner can synthesize motions over gaps and obstacles and discover efficient driving patterns. To the best of our knowledge, this was the first time that motion plans in rough terrain were computed for a legged-wheeled robot.

Lastly, we formulated a robust version of the well-known support polygon constraint. Instead of explicitly computing halfspaces, like most related work, we added a convex hull formulation with a tuning parameter ϵ between 0 and 1 to control how conservative the planner should be. We have used this constraint inside every optimization-based planner following this work.

Paper III

Jelavic, E., Berdou, Y., Jud, D., Kerscher, S., and Hutter, M. (2020). "Terrain-adaptive planning and control of complex motions for walking excavators". In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, pp. 2684–2691

This paper develops a controller capable of executing whole-body plans on real hardware, thus addressing question *Q3*. The developed controller relies on Hierarchical Optimization (HO) and uses a full-rigid body model to compute instantaneous torques. It can be considered an inverse dynamics algorithm with constraints. We encode the terrain adaptation into the controller through task prioritization. By setting a higher priority on base tracking, we found that the limbs adapt to the uneven terrain. The paper also considers allocating joint control modes for HEAP since only some joints are torque controllable. We demonstrated driving and stepping motions on a 12-tonne-legged excavator, the first time that whole-body motions were executed on a full-size machine.

Paper IV

Jelavic, E., Farshidian, F., and Hutter, M. (2021). "Combined sampling and optimization based planning for legged-wheeled robots". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 8366–8372

This article extends the local planner presented in Paper II with global planning capabilities. The extension combines sampling and optimization, thus introducing a hybrid planner. This work focuses on the development of the sampling-based planner, therefore directly trying to answer question *Q2*. The sampling-based stage computes a whole-body configuration and a contact schedule, which speeds up the optimization convergence. The optimization-based step ensures that all the system constraints, such as non-holonomic rolling constraints, are satisfied. By virtue of the sampling-based planner, we can navigate challenging terrain, and by virtue of the optimization, we can produce smooth motions for high DoF systems. We achieve fast planning rates by following the idea from Short and Bandyopadhyay, 2017 and moving some computations offline. In addition, we relax the massless limb assumption from Short and Bandyopadhyay, 2017 and extend the method to be applicable for robots with heavy limbs (such as HEAP).

Paper II used an analytical elevation map (parametrized by functions) of limited applicability in the real world. Here, we incorporate a general-purpose discretized elevation map into the optimization problem. Compared to their legged counterparts, legged-wheeled robots keep their limbs in contact with terrain over long distances. Hence more variables and constraints are affected by rounding errors and discretization from the elevation map, which makes the optimization more sensitive. This paper proposes a strategy to mitigate these problems using gradient clipping and higher-order interpolations.

Paper V

Jelavic, E., Qu, K., Farshidian, F., and Hutter, M. (2022c). "LSTP: Long Short-Term Motion Planning for Legged and Legged-Wheeled Systems". *submitted to IEEE Transactions on Robotics*

The last paper improves the preceding work in all aspects of the questions listed above. We extend the developed hybrid planner to handle a broader class of robotic systems. In particular, the SBP is extended to handle legged robots with point feet and legged-wheeled robots with non-steerable wheels. The extension is non-trivial and requires revisiting the initialization step (question *Q2*).

In this paper, we showcase some challenging maneuvers on real hardware, which leads us to revisit questions *Q1* and *Q3*. We reformulate the optimization problem such that we can execute it in an MPC fashion. The most significant change is that we move away from the collocation method formulated in Paper II, which is very robust but does not run in real time. Instead, we move towards shooting methods that better exploit the planning problem structure (Eq. 1.1). This also gives us robustness when executing on hardware, thus directly addressing question *Q3*. Furthermore, for the case of a legged excavator, we revisit the control system designed in Paper III. With the new design, the controller can execute more maneuvers, exhibits improved robustness, and requires less tuning while retaining good qualities such as terrain adaptation.

Lastly, we validate the improved planner on three different systems, a legged excavator, a legged robot with point feet, and a legged-wheeled robot. With the enhanced control system for the legged excavator, we can execute motions on par with what the best human operators can do. The results show that computing and executing hybrid locomotion plans is possible on hardware in real-time.

1.4 Robotic Platforms

The motion planner proposed in this thesis is generally conceived so that other robotic systems can use it. Implementation was also designed to be general by introducing relevant abstractions for robot-specific parts. Snapshots of the planner in action are shown in Fig. 1.3. The focus lies on the legged excavator HEAP introduced in Jud et al., 2021b. In Bjelonic et al., 2022; Jelavic et al., 2022c, we test the sampling-based planner on ANYmal on wheels robot (Bjelonic et al., 2019a). The proposed planner was also tested on the legged robot ANYmal (Chapter 6). Table 1.1 shows the specifications for all the robots.

(a) HEAP **(b)** ANYmal on Wheels **(c)** ANYmal

Figure 1.3: Different testing platforms used in this thesis. Left to right: legged excavator HEAP, legged-wheeled robot ANYmal on Wheels and legged robot ANYmal.

Spec / Robot	mass [kg]	torso length [m]	num legs	num wheels	num arms	num joints	toal DoF
ANYmal C	156	2.5	4	0	0	12	18
ANYmal C on wheels	60	2.45	4	4	0	16	22
HEAP	12300	6.19	4	4	1	25	31

Table 1.1: Datasheet for different robots used in this thesis.

2

Robotic Precision Harvesting: Mapping, Localization, Planning and Control for a Legged Tree Harvester

Jelavic, E., Jud, D., Egli, P., and Hutter, M. (2022a). "Robotic Precision Harvesting: Mapping, Localization, Planning and Control for a Legged Tree Harvester". *Field Robotics* 2, pp. 1386–1431

DOI: 10.55417/fr.2022046

Video: https://youtu.be/1FLD0djPFgU

This paper presents a system for autonomously conducting precision harvesting missions using a legged harvester. Precision tree harvesting removes some trees selectively, while leaving neighboring trees intact. Our robot performs the challenging task of navigation and tree grabbing in a confined, GPS-denied forest environment. We propose strategies for mapping, localization, planning, and control and integrate them into a fully autonomous system. The mission starts with a human mapping the area of interest using a detachable, custom sensor module. Subsequently, a human expert selects the specific trees for harvesting. The sensor module is then mounted on the machine and used for localization within the created map. A planning algorithm searches for both an approach-pose and a path in a single path planning problem. We design a path-following controller exploiting the legged harvester's capabilities for negotiating rough terrain. Upon reaching the approach-pose, the machine grabs a tree with a general-purpose gripper. Our system has been tested in both

emulated and natural forest settings. To the best of our knowledge, ours is the first robot to demonstrate such a level of autonomy on a full-size, hydraulic machine operating in a realistic environment.

2.1 Introduction

Efficient forest management is of interest to all humankind. Forests cover roughly 30% of the world's land surface United Nations Food and Agricultural Organization, 2018. Trees provide the human population with renewable raw materials and a significant source of both energy and oxygen. Besides, woodlands provide a habitat for animal wildlife and contribute to the fight against climate change.

The forestry industry contributes 1.2% of the global Gross Domestic Product (GDP) and employs about 30-45 million people Renner, Sweeney, and Kubit, 2008. In some countries forestry products have a significant share in the total value of exported goods, e.g., Finland (18%) and Latvia (16%) Swedish Forest Agency, 2014. Given the growing labor shortage Hawkinson, 2017 and the classification of tree harvesting as a *3D* job (dirty, difficult, and dangerous), automating forestry operations deserves a high economic priority.

2.1.1 Automation in Forestry

Forestry mechanization has increased productivity to the point that human operators have now become the critical bottleneck (Parker, Bayne, and Clinton, 2016). Since modern harvesting machines have many Degrees of Freedoms (DoFs), training efficient operators takes time and slows further productivity gains. To mitigate the problem, researchers have developed semi-autonomous modules employing Inverse Kinematics (IK) control for crane operation, such as those in Fig. 2.1a and Fig. 2.1b. Examples of such work can be found in Hellström et al., 2008; Hyyti, Lehtola, and Visala, 2018; Ortiz Morales et al., 2014; Westerberg, 2014.

Another common forestry task is thinning, a process wherein stewards selectively remove some trees to allow more space for others Forestry Focus, 2018. Thinning requires negotiating tight spaces (as opposed to conventional clear-cutting) and can be done with smaller machines that are remotely controlled within the operator's field of view. Fig. 2.2 shows two examples, Harveri and eBeaver. Despite their lower cost and reduced damage to the forest ground compared with large harvesters, machines such as Harveri are not very popular, since operators are reluctant to give up a cabin's comfort and safety

(a) Komatsu-895 forwarder **(b)** Komatsu-951 harvester

Figure 2.1: Different machines are typically used in modern forestry operations. They all have operator cabins to increase the comfort level in a possibly wet and muddy forest environment. Images were taken from Komatsu Corporation, 2019

(a) RCM Harveri harvester **(b)** eBeaver harvester

Figure 2.2: Smaller harvesters that are typically used for thinning operations. They do not have a cabin for the operator and are remotely controlled within the operator's line of sight. Images taken from Jukka Hämekoski, 2016 and Magnus Gustavsson, 2016.

(personal communication with Silvere, 2017). Our system is developed to be suitable for both conventional clear-cutting and thinning.

2.1.2 Precision Forestry

Traditional forestry operations are still primarily based on fundamentals developed about 300 years ago Von Carlowitz and Rohr, 1732. Nowadays, the trend is to move towards precision forestry, which practice uses automated data collection (versus manual measurement) and software-aided analysis to allow site- and tree-specific husbandry. Instead of compartment-based management relying on human experience and qualitative judgment, massive digital data enables much more granular and quantitative forest management. With the

newest sensing technologies (e.g. Light Detection And Ranging (LIDAR)), precision forestry can even be done on the single tree level Choudhry and O'Kelly, 2018, Holopainen, Vastaranta, and Hyyppä, 2014. Compared to the traditional techniques, precision forestry diminishes the error in inventory estimates by 400 percent Choudhry and O'Kelly, 2018. Other benefits such as granular fertilizing, fire monitoring, pest and disease monitoring increase yields and labor productivity up to 10 times Silvere, 2017.

2.1.3 Scope

This article describes a robotic system, comprising modified commercial hardware and custom software, designed specifically for tree-grabbing. We focus on automating the workflow until the moment of cutting the tree. Cutting and debranching are not investigated since there are existing harvesting tools for this process. We focus on executing an autonomous mission (referred to as harvesting in further text) and main aspects (other than cutting) that executing such a task entails, namely mapping, localization, planning, control, and tool positioning. Autonomy modules developed in this work are not task specific and could be used for either harvesting or thinning.

To this end, we develop a versatile sensor module for collecting data and localizing the machine. We present our hardware platform, an automated Menzi Muck M545 harvester Jud et al., 2021b augmented with custom sensors and actuators (see Fig. 2.3), and describe our technical approach to executing the critical, initial phase of a harvesting mission as specified by a human or algorithmic expert. Lastly, we present experimental results emulating a real harvesting mission in a forest and report our recommendations for integrating and tuning the system.

High-level decisions on which trees to cut and how to manage the forest inventory are beyond this article's scope. Nowadays, companies can create forest inventories and recommend future actions with aid from machine intelligence (e.g., Silvere, 2017), and our pipeline assumes using such a service.

2.1.4 Contribution

To our knowledge, we here present the first report of successfully deploying a large-scale, fully autonomous system for forest husbandry. Our approach enables operations under the forest canopy, which are significantly harder than clear cutting since the machine has to navigate among trees where Global

Figure 2.3: Legged excavator navigating to a tree and performing a grab. Human is in the cabin for safety reasons.

Positioning System (GPS) signal may be unreliable. In summary, our work extends state of the art with the following contributions:

- Design of a portable sensor module with a sensor setup that enables mapping and localization. We test our sensor module in handheld operation, and we deploy it on a robotic platform.

- Development of a robust algorithm for converting raw point clouds into 2.5D elevation maps. We demonstrated the algorithm's applicability in a real forest and in several other environments. Our implementation is available as open-source[1].

- We develop an approach-pose planning algorithm suited for tight spaces and both structured and unstructured environments. The algorithm is evaluated on synthetic data and maps produced from real data in a planning scenario including large-scale harvesters.

- Development of a control procedure for driving a legged harvester in rough terrain. The algorithm performs path tracking and chassis stabilization at the same time.

[1] `https://github.com/ANYbotics/grid_map/tree/master/grid_map_pcl`

- Both planning and path tracking algorithm implementations have been made publicly available for the community[2].
- We develop a lightweight method for tree detection based on LIDAR scans of local forest patches. Our method is purely geometry-based and operates directly on point clouds. It is suitable for online operation on the robot and we make it available for the community [3]
- We integrate all system components into a fully functioning autonomous system. Experimental verification of our pipeline is done on a full-size legged harvester in a realistic environment
- An evaluation of system components in the real-world scenario is presented. The evaluation includes chassis control, tree detection, approach-pose planning, mapping, localizing, and point-cloud to elevation-map conversion.

The rest of the paper is organized as follows: Chapter 2.2 introduces the related work. Chapter 2.3 describes the robotic platform and hardware used in the experiments. In Chapter 2.4, we give a high-level overview of the proposed harvesting system with brief explanations for each of the submodules. Chapter 2.5 describes mapping and localization, while Chapter 2.6 describes tree detection procedure. Control and planning are described in Chapters 2.7 and 2.8, respectively. Chapter 2.9 presents results and evaluation. Finally, Chapter 2.10 summarizes the paper and Chapter 2.11 discusses conclusions and future work.

2.2 Related Work

In this section, we give a brief overview of the work relevant for precision harvesting missions; a more in-depth survey on forestry robotics can be found in Oliveira, Moreira, and Silva, 2021. To the best of our four knowledge, no autonomous system for thinning or harvesting missions has been presented in the literature so far. The harvester presented in Rossmann et al., 2009 is the closest to our work in the sense that developed methods are showcased on a full-size machine. The authors develop a particle filter-based localization for forest environments; however, no attempt to automate other machine parts

[2] https://github.com/leggedrobotics/se2_navigation
[3] https://github.com/leggedrobotics/tree_detection

was made. Similarly, in Li et al., 2020, a full-size harvester is fitted with a 3D LIDAR the authors develop a place recognition algorithm based on tree stems.

To acquire an overview about the forest and plan for harvesting, maps are typically first acquired by airborne surveying Naesset, 1997, Roßmann, Krahwinkler, and Bücken, 2009. For mapping and localization, LIDAR sensors are a popular choice. GPS based localization can be inaccurate under a tree canopy, and vision-based sensors (cameras) can be sensitive to illumination changes in outdoor environments. For LIDARs, Iterative closest point (ICP) is a popular method for scan registration and has been used for mapping in forest surveying Morita et al., 2018; Tsubouchi et al., 2014; Yue et al., 2018. In Babin et al., 2019, ICP is fused with GPS for mapping subarctic forests with sparse canopy. The ICP is also used for mapping in Tremblay et al., 2020 and the authors propose a method for determining Diameter at Breast Height (DBH). Another possibility is to use filtering with trees as landmarks. Rossmann, Krahwinkler, and Schlette, 2010; Roßmann, Krahwinkler, and Bücken, 2009 build a map using an airborne laser sensor and localize a harvester in *SE(2)* space using a particle filter. In Miettinen et al., 2007, an Extended Kalman Filter (EKF) based Simultaneous Localization and Mapping (SLAM) is used on a skid steer robot to build a map and localize within it. The downside of relying on tree landmarks is that it can be susceptible to false positives.

However, LIDAR mapping does not have to rely on tree landmarks. Thus, mapping algorithms without tree stem detection can potentially handle the most general forest types (sparse, dense, thick vegetation). Many researchers have used LOAM (Zhang and Singh, 2014, 2015) for LIDAR mapping, and numerous variants of LOAM have been tested in urban environments. LOAM is a LIDAR odometry and accumulates drift which may be problematic for large areas. Recently, loop closing has been added to LOAM (Shan and Englot, 2018; Shan et al., 2020), which enabled the algorithm to correct for the accumulated drift. The loop closure mechanism is based on ICP to map matching and cannot handle large drift. In Chen et al., 2020; Nevalainen et al., 2020 LOAM has also been used in a forest. An alternative to LOAM is Cartographer Hess et al., 2016 which has been shown to work well for large-scale mapping. We use Cartographer as a mapping algorithm of choice.

Once in possession of a map, one typically wants to estimate the biomass or classify the tree species, which requires segmentation of both the canopy and the stems. The forestry community has extensively studied techniques for tree segmentation and classification. Example of model-based approaches include Zhang et al., 2019b, Burt, Disney, and Calders, 2019. Model-based approaches

typically rely on pointcloud processing such as euclidean clustering, surface normal computation, and RANSAC model fitting to segment out the trees. On the other hand, learning-based approaches train networks end to end to segment trees from point clouds Ayrey et al., 2017, Chen et al., 2021,Bryson, 2017. The need to accurately segment out whole trees in cluttered scenes renders most of the approaches complicated and with many steps. Hence, all algorithms are designed for offline operation, and some require powerful computational resources. In this work, a lightweight model-based approach operating online on the robot is used for tree detection.

A harvester can be treated as a big mobile manipulator. Planning and control for mobile manipulators is a well-studied problem in robotics. One can either treat the robot in a whole-body fashion Gawel et al., 2019; Giftthaler et al., 2017; Kim et al., 2019 or decouple the planning and control for the arm and the base Carius et al., 2018b; Schwarz et al., 2017. The latter approach has often been used for forestry automation. Controlling the harvester's (or forwarder's) crane is an integral part of forestry operations. Examples can be found in Kalmari, Backman, and Visala, 2014; La Hera et al., 2009; Lindroos et al., 2015; Westerberg, 2014. A major effort was put towards semi-automating the crane operation since coordinating many DoFs is one of the hardest tasks for a human operator. Most crane control algorithms are based on IK or Inverse Dynamics (ID) Siciliano et al., 2010. Compared to classical robotic manipulators, the biggest difference is that harvester cranes were seldom designed with autonomous operations in mind. They often come without sensing capabilities which means they have to be retrofitted with sensors to estimate the end-effector position. Moreover, the sensors have to be robust to withstand operation outdoors—such retrofitting results in increased automation effort. An additional difficulty is that hydraulic actuators are harder to control than their electric motors counterparts (nonlinearities, valve overlap, less bandwidth). To minimize the automation effort, Morales et al., 2011; Ortiz Morales et al., 2014 investigate possibilities for purely open-loop control. More recently, an attempt has been made to develop arm motion planning for a feller-buncher machine Song and Sharf, 2020, Song and Sharf, 2021. The authors rely on Zero Moment Point (ZMP) to compute a stable trajectory guaranteeing the machine's (tipping over) stability. The stability of the harvester becomes essential as soon as one uses an actuated end-effector for tree manipulation (as opposed to passive tools that let trees fall freely).

Little has been done to develop a fully autonomous system for forest environment missions; the robotic community has mostly focused on urban envi-

ronments. Mikhaylov and Lositskii, 2018 shows a very simplistic architecture without any validation. In Georgsson et al., 2005, a GPS based tracking approach is presented and validated on a forwarder machine outside of a forest environment. Tominaga, Eiji, and Mowshowitz, 2018 show experiments on an All Terrain Vehicle (ATV) without an arm in a small scale environment using GPS with Real-time Kinematic (RTK) correction. The authors use a global graph-based mission planner and a local state space sampling planner. A pure-pursuit algorithm is used for tracking. The proposed planning and control strategy cannot handle backward driving, limiting its ability to negotiate confined spaces. Zhang et al., 2019a presents an integrated system for navigation in a forest. The authors present a planner and a path follower that run on a small skid-steer robot without an arm. They show results in a structured forest (a rubber tree farm) where trees form a grid. This is reflected in the path generation algorithm, which involves heuristics to exploit the environment's structure and generates only straight-line paths.

Wooden et al., 2010 develop a navigation system for the Big Dog robot; it was tested in a forest, and it features a local planner based on a graph search A* algorithm. Hellström et al., 2008 discusses different planning algorithms (A*, elastic bands, potential fields) for forwarder machines; however, no results are shown since it is a pre-study only. More examples of forest navigation using graph search algorithms to compute paths can be found in Mowshowitz, Tominaga, and Hayashi, 2018; Tanaka et al., 2017. Compared to the Unmanned Ground Vehicles (UGVs), much more work has been done for Unmanned Aerial Vehicle (UAV) path planning in forests, and examples can be found in, e.g., Cui et al., 2014; Liao et al., 2016; Pizetta, Brandao, and Sarcinelli-Filho, 2018. There is a research gap for UGV path planning in natural environments, which this article tries to bridge. We compute plans for a non-holonomic constrained vehicle using RRT* Karaman and Frazzoli, 2011.

2.3 Hardware

The Hardware section introduces the robot used and it describes the sensor module we developed for the autonomous missions.

2.3.1 Platform

Our robotic platform is Hydraulic Excavator for Autonomous Purpose (HEAP). It is based on a Menzi Muck M545 multi-purpose legged machine often used for harvesting in challenging terrain. It is customized for teleoperation and fully autonomous operations. Besides forestry work HEAP, can be used for digging, landscaping and manipulation tasks as well (see Jud et al., 2019,Johns et al., 2020) . We use the terms HEAP and harvester interchangeably in further text. Our machine is fitted with custom hydraulic actuators with pressure sensors and a high-performance servo valve. The actuators allow for precise force and position control, thus enabling active chassis balancing and adaptation to the uneven ground (see Hutter et al., 2016b).

(a) Front sensors on the cabin

(b) Localization sensor module in the back

Figure 2.4: ***Left:*** two LIDAR sensors mounted on top of the cabin and used for scanning the area in front of the machine. They are primarily used for tree detection. ***Right:*** The localization module introduced in Sec. 2.3.2 mounted in the back of the machine.

Proprioceptive Sensing: Two SBG Ellipse2-A Inertial Measurement Unit (IMU)'s (one in the cabin and one in the chassis) gather the inertial data that are primarily used to determine the chassis' roll and pitch angle. HEAP

(a) Sensor module mounts **(b)** Sensor module packed

Figure 2.5: ***Left:*** Close-up photo of sensor module mounts. The module is mounted onto the aluminum profiles (shown with arrows), clamped to the machine using the o-ring clamps (encircled with dotted line). The lever fasteners are used to adjust the angle of the module (encircled with full line). ***Right:*** The sensor module dismantled and packed after deployment.

is equipped with a series of IMUs rigidly attached to each arm link. Measurements from these IMUs are fused together to estimate end-effector pose and joint angles in the machine frame. Note that, unlike in Jud et al., 2019, we do not use externally mounted draw-wire encoders on the arm. This way, we avoid a potential entanglement of draw wires with tree branches.

Exteroceptive Sensing: Two Velodyne Puck VLP-16 LIDAR's are used for tree detection. It is important to note that one of the Pucks is rotated by 90° around the rolling axis (see Figure 2.4a) in order to get a better resolution when mapping the area in front of the machine. A sensor module (see Sec. 2.3.2) is mounted on the back of the machine, as shown in Figure 2.4b. A close-up image of the sensor mounts is shown in Fig. 2.5a. The sensor module is mounted using aluminum profiles from *item*item, 2021, and lever hinges from the same company. We attach it to the machine using o-ring clamps with rubber to mitigate the effect of vibrations. The mounts are rigid; hence the whole module does not move w.r.t to the cabin frame. Note that the module is not mounted in the middle to reduce further engine vibrations (the engine is just below the sensor module).

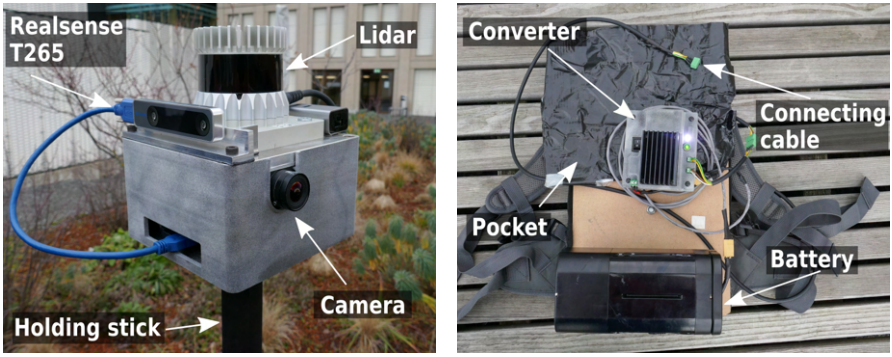
Trees can snap when manipulated, resulting in heavy debris falling on the machine. Hence, for a tree felling application (as opposed to tree grabbing), one needs to make the sensor module rugged and add mechanical protection (e.g., bulletproof glass). Another option would be to place the sensors directly

inside the cabin. It is important to place the module such that it has as large Field of View (FOV) as possible which is beneficial for the localization. Alternatively, one can use multiple synchronized sensors, each with a smaller FOV. For HEAP, the sensor module placement in the back as seen in Fig. 2.4b has an effective FOV of about 180° (instead of 360°), which was enough for localizing.

The planning, control, and tree detection software stack runs on one computer (Intel i7-5820K, 6x3.60GHz, Ubuntu 18.04, 32 GB RAM) installed in the cabin. The control loops work at 100 Hz and are triggered by the Controller Area Network (CAN) driver. All the algorithms presented are implemented using C++ with Robot Operating System (ROS) as integration middleware. For more details on HEAP, please refer to Jud et al., 2021b.

2.3.2 Sensor Module

We develop a sensor module for mapping and localization as displayed in Figure 2.6. The module can be used in the first step by a human to map the area of interest (see Fig. 2.7a). In a second step, the system can be mounted on HEAP (Fig. 2.4b) to localize the machine in the previously built map without GPS information. The whole module can be dismantled and packed, as shown in Fig. 2.5b.

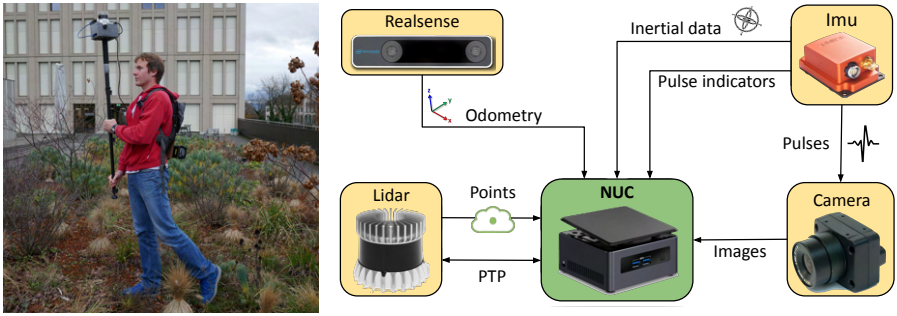


(a) Sensor module **(b)** Backpack with power

Figure 2.6: ***Left:*** Head of the sensor module. The Central Processing Unit (CPU) and the IMU are inside the grey box. ***Right:*** Power supply for the sensor module mounted on the backpack.

We design the sensor module to be lightweight such that handheld operation is possible. It has an integrated computer such that mission data can be

collected and saved without an external laptop. Besides, one can run the SLAM directly on the module in real-time. The module has a LIDAR, an IMU, and two visual sensors. LIDAR is used as a primary sensing modality with aid from visual and inertial sensors. This way, the motion distortion in the point cloud can be corrected. Extrinsic calibration between the sensors is obtained from *Kalibr*[4] Furgale, Rehder, and Siegwart, 2013 and *lidar_align*[5], both of which are available open-source. Lastly, all sensors installed on the module are time-synchronized. Time synchronization is essential for state estimation and mapping algorithms to function correctly.



(a) Sensor module in action **(b)** Sensor module schematic

Figure 2.7: ***Left:*** Human operator mapping an area of interest. ***Right:*** Schematic of the sensor module. It is composed of a camera, tracking camera (realsense), LIDAR and an IMU.

The sensor module in operation is shown in Fig. 2.7a and sensor components in Fig. 2.6a. We use an Ouster OS-1 LIDAR with 64 channels, an Intel Realsense T265 tracking camera, and a FLIR camera model BFS-U3-04S2M-CS. Inside the grey box in Fig. 2.6a, one can find a computing unit (Intel NUC with i7-8650U processor) and an IMU (Xsens MTI-100). Total cost of our sensor module is about 10500 USD. We implemented two complementary solutions for Visual Inertial Odometry (VIO): FLIR camera synced with IMU and the Intel Realsense T265 tracking camera. The FLIR camera has an excellent low light performance, while the T265 has almost 180° FOV and higher frequency. Hence, one can choose the VIO sensor used depending on the application. In this work we use T265 because of large FOV and because higher frequency odometry estimates were beneficial for the LIDAR based mapping.

[4] `https://github.com/ethz-asl/kalibr`
[5] `https://github.com/ethz-asl/lidar_align`

The sensor module can be mounted on a stick shown in Figure 2.6a. In this way, a human can walk around with the module and point it to the places of interest to map it in sufficient detail. The sensor module is powered from the backpack (shown in Fig. 2.6b), equipped with a 36 V battery. We choose to use a 300 W h e-bike lithium-ion battery (BiX Power BX3632H) with an integrated battery management system. All smaller sensors like cameras and the IMU are powered through Universal Serial Bus (USB).

The schematic of the sensor module is shown in Fig. 2.7b. The Intel Realsense camera provides software-synchronized VIO at a frequency of 200 Hz. The FLIR camera (20Hz) and the IMU (400 Hz) are synced using customized software. The LIDAR is connected directly to the computer, and it is configured to use Precision Time Protocol (PTP) for time synchronization Eidson, 2006. We modify the LIDAR's driver to send the packets as soon as they are received instead of batching them until the scan is completed. This way, the motion distortion is mitigated. We use the TICsync package Harrison and Newman, 2011 to synchronize the IMU with computer's clock. TICsync estimates the bias and the drift between the IMU's internal clock and the computer's clock to recover the exact time when the measurement happened. To increase the robustness, it is beneficial to assign the real-time priority to the IMU driver (minimizes the timestamp jitter).

2.4 Approach Overview

Fig. 2.8 presents an overview of the autonomous harvesting system and serves as a visual outline for the chapters in this paper. We briefly describe each subsystem and give a more detailed description in their respective chapters.

Mapping and Localization: We use a LIDAR based SLAM together with visual and inertial measurements to correct for scan distortion. The map is built by processing the data offline. At mission time, scans are registered against the existing map to compute the sensor module's pose.

Tree Detection: The LIDAR sensors mounted on the top of the cabin (see Figure 2.4a) deliver point clouds at 20 Hz. An intermediate node aggregates them and passes them to the tree detection module. Detection is done purely based on geometric features. The tree detection module forwards the detected tree's coordinates to the grasp pose planner and the approach-pose planner.

Control: The control subsystem is split into two parts. The arm controller ensures trajectory tracking by sending velocity commands to the arm joints (IK

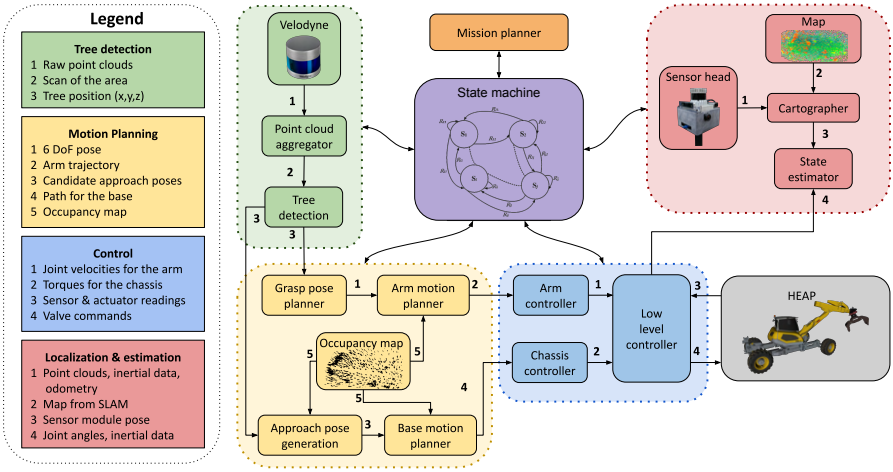


Figure 2.8: Overview of the system architecture deployed on HEAP. Components belonging to the same subsystem are shown in the same color. Different subsystems are shown in different colors. Arrows depict communication channels in the system, with some communication lines omitted for the sake of brevity.

controller). The base controller ensures path tracking, and it regulates contact forces such that the base stays upright when driving over uneven terrain. The low-level controller transcribes higher level references (velocity, torque, positions) to the valve commands.

Motion Planning (Arm and Base): The planning stack comprises three planners. The base motion planner plans a path and an approach-pose for the base of the harvester. The grasp pose planner computes a gripper pose in 3D space that encompasses the tree trunk. Finally, the arm motion planner produces a collision-free spline trajectory to move the gripper into the desired pose.

Mission Planner: This module determines which tree to grab next, and it sends the target position to the state machine, which then sends it to the approach-pose planner.

State Estimation: The state estimator uses the sensor module pose and proprioceptive measurements (IMU, joint angles) to compute the complete state of the system (6 DoF base pose + joint angles).

State Machine: The state machine coordinates the execution of different tasks. It works on a handshaking principle where the state machine sends a request to a subsystem, and the subsystem responds with an acknowledgment once the

Table 2.1: Order of operations and subsystems involved to grab a single tree

Task order	Task Description	Subsystems responsible
1	Get the next tree position.	Mission planner
2	Plan an approach-pose and a path. If it fails, go to step 1.	Planning
3	Drive the machine to the approach-pose found in step 2. In case tracking fails, go to step 2	Control
4	Retract the arm (in case it is not already retracted)	Planning, Control
5	Scan the area around the expected tree location.	Control
6	Run tree detection algorithm and plan a grasp pose	Tree detection, Planning
7	Plan and track an arm trajectory to reach the grasp pose	Planning, Control
8	Plan and track arm trajectory back to default position (arm retracted all the way).	Planning, Control
9	Go to step 1.	State Machine

task requested has been completed. Tasks that the state machine can request and the order of operations to grab a single tree are shown in Tab. 2.1.

2.5 Mapping and Localization

Precision forestry requires precise and consistent 3D geometric information about the forest. This data can be collected by a harvester itself, a smaller UGV or by a human carrying the sensors. The collected information can then be used for mission planning and forest inventory management. Moreover, the map serves as a reference for the harvester to localize. To retain flexibility, we would like to make no assumptions on the surroundings and allow the harvester to work in different types of forests (e.g., sparse, with vegetation, non-flat). Therefore we avoid using tree stems as landmarks (e.g., Roßmann, Krahwinkler, and Bücken, 2009) since they can be hard to detect (vegetation) or they may be scarce (e.g., a glade inside the forest).

We selected Google Cartographer Hess et al., 2016 as the backbone of our mapping and localization pipeline. Cartographer is a grid-based SLAM approach that uses the scan to sub-map matching for loop closure detection and discards unlikely matches using the branch and bound method. At mission time, localization can be achieved simply by scan to sub-map matching. Compared to other SLAM systems available at the time, Cartographer has the advantage that it is fully open source, can cover large spaces, features loop closures that work robustly, and has a large community of users. Below we provide a brief description of the working principle.

Cartographer divides the world into a raster of submaps where a number of accumulated scans determines the size of the submap. The submaps are grids

(3D) where each cell is assigned a probability of being occupied. Before inserting a scan into a submap, the scan is voxelized, and each grid cell inside the scan is classified as *occupied* of *free*. Subsequently, the range scans is registered within the submap by solving a nonlinear least-squares problem which maximizes the probabilities at the scan points in the submap. This local, grid-based SLAM relies on a good initial guess which is achieved by extrapolating the previous pose using the inertial/odometry data.

Cartographer follows the Sparse Pose Adjustment approach of optimizing all scans, and submaps Konolige et al., 2010. Each range scan is associated with a trajectory node in the pose graph. In the background, all scans are matched to nearby submaps to create loop closure constraints. Suppose a sufficiently good match (user-defined minimum score) is found in a search window around the currently estimated pose. In that case, it is added as a loop closing constraint to the global optimization problem. The constraint graph of submap and scan poses is periodically optimized in the background (every few seconds). When localizing in a known map, Cartographer keeps only the latest N submaps, which are then considered for loop closures against the submaps in the known map. The known map is not updated.

2.5.1 Mapping

Examples of maps generated with Cartographer are shown in Fig. 2.9.

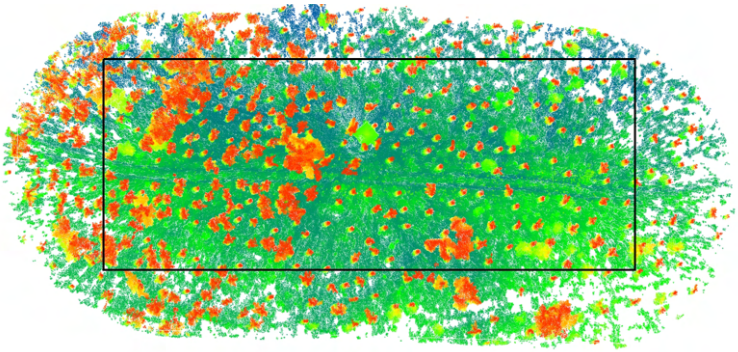
The map in Fig. 2.9a shows a point cloud of a forest viewed from above. Blue and green colors correspond to a lower elevation (ground), whereas yellow and red colors correspond to a higher elevation (vertical structures like tree trunks and canopy). The map is about 140 m long and 60 m wide, and it shows a part of the forest where we conducted the experiments with HEAP. Another, larger map (340 m x 170 m) is shown in Fig. 2.9b; the size of this map shows that Cartographer can map areas sufficiently large for an autonomous harvesting mission. Both maps have been processed offline and bundle adjusted. Compared to online processing, building the maps offline allows us to use more points from the LIDAR which results in denser maps, use finer resolution voxels, and sample for constraints between submaps more often. Furthermore, we can increase the search radius for loop closures and run the scan matcher optimization more often. All the changes above result in less drift and more loop closures, which produce more consistent maps.

Once we have a consistent 3D map of the space, we use it for localization at mission time. Since the planning is done in *SE(2)*, we convert the 3D map into

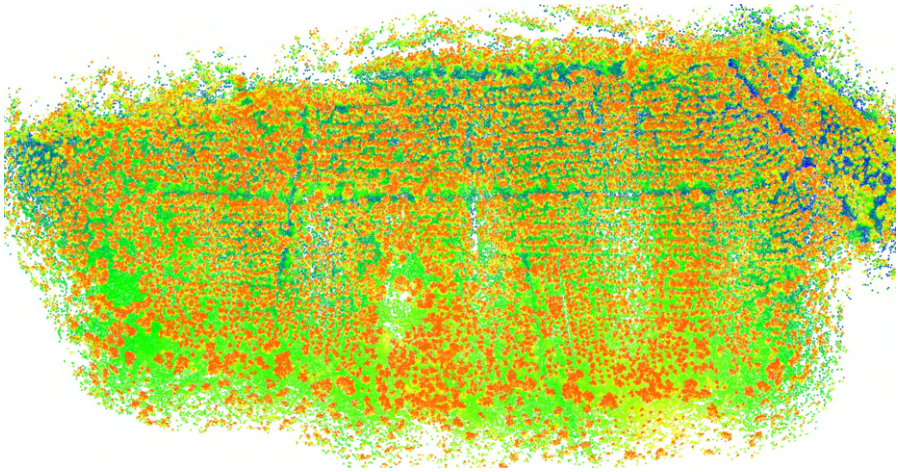
a 2.5D map which the planner then uses. The 3D map is first converted into an elevation map, and from the elevation map, we compute a traversability map. Both maps are functions $f : (x, y) \to \mathbb{R}$ mapping the 2D coordinates to height or traversability. Below, we detail the conversion of the point cloud into an elevation map used by the planner.

A grid map data structure Fankhauser and Hutter, 2016a is used to store the elevation map. From the raw point cloud in Fig. 2.9a, our algorithm builds a 2.5D elevation map of the area shown in Fig. 2.10. Conceptually, the algorithm is similar to the one used in Fankhauser, Bloesch, and Hutter, 2018. In contrast to Fankhauser, Bloesch, and Hutter, 2018, our algorithm can handle multiple elevations in the cell, i.e., multiple points with the same (x, y) coordinates and different z coordinates. Thus, it can filter out vegetation and clutter and recover true ground elevation more robustly. We successfully used the point cloud to elevation map conversion algorithm in both planar and non-planar environments. Implementation is available as an open-source package[6].

[6] `https://github.com/ANYbotics/grid_map/tree/master/grid_map_pcl`



(a) Testing forest patch



(b) Large forest area

Figure 2.9: Bird view of forest maps. Both maps are represented as point clouds. ***Top:*** Small forest patch where we conducted the experiments. Approximate size 140 m x 60 m The area encircled with a black rectangle was converted to an elevation map (Fig. 2.10). ***Bottom:*** Map of a larger forest area (top view), where range, odometry and inertial data has been collected in multiple tours, concatenated and the processed with Cartographer, data courtesy of Silvere, 2017. Approximate map dimensions: 340 m x 170 m.

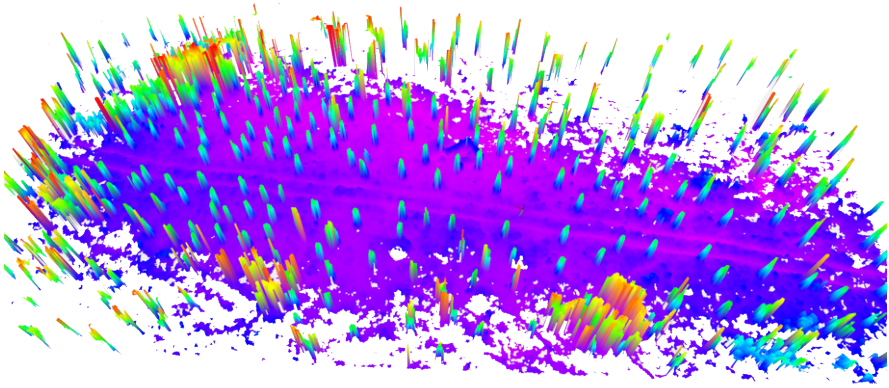


Figure 2.10: Elevation map of a forest patch encircled black in Fig. 2.9a. Purple color corresponds to the lowest elevation and red to the highest. The resulting grid map size is 1419 by 1318 cells with a resolution of 10 cm. Note how tree trunks are clearly visible in the elevation map.

The point cloud to elevation map conversion algorithm starts by removing the outliers and downsampling the input point cloud (implementation based on Rusu and Cousins, 2011). The main loop is parallelized to speed up computation. For each cell in the grid map, we pre-compute all the points contained within that cell (this enables lookup in $O(1)$ time), thus vastly speeding up the algorithm. Each cell is a rectangle in x, y coordinates. Once we have fetched the points inside the grid map cell, Euclidean clusters are computed. The cluster size is regulated with a cluster tolerance parameter (see Rusu, 2010). Any points withn the tolerance distance are considered the same cluster. After clustering, we calculate the centroid of each cluster. Finally, the centroid with the smallest z coordinate is deemed to be the ground elevation. Illustration of the process is shown in Fig. 2.11. In Fig. 2.11, grid map cells are extended in the z direction and form columns 1,2, and 3. Column 1 has one cluster which contains only the ground points (shown in light green), and the algorithm correctly extracts the ground height. In column 3, the algorithm extracts at least two clusters. The lower cluster is the ground (light green color), while branches and leaves form the upper cluster (dark green). Clusters are disjoint, and the algorithm correctly recovers the ground elevation as a mean value of the lower cluster. Column 2 contains one large cluster composed of both ground and the tree. In this case, we cannot recover ground information since the resolution of the map is too coarse; the height of the tree trunk determines the height. In

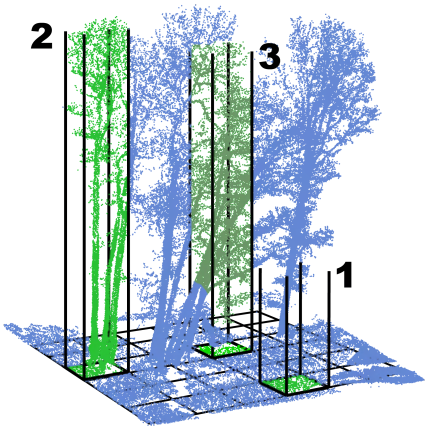


Figure 2.11: Generating a grid map from a raw point cloud. The grid is shown with black lines, and the input point cloud is shown in blue color. Clusters of points that the conversion algorithm might extract are shown in green. Note that image is not drawn to scale.

this work, we use the resolution of 10 cm for the grid. We analyze algorithm's behaviour in presence of vegetation and clutter in the Appendix 2.12.1.

Qualitative runtimes of our algorithm are shown in Table 2.2. The foreseen use case is to run the algorithm once offline to construct a global elevation map that can be used for planning. The times shown in Table 2.2 were obtained on an Intel Xeon E3-1535M (2.9 GHz, 4 cores). The algorithm is not limited to application in forests only but also generalizes to non-forest environments (see `https://github.com/ANYbotics/grid_map/tree/master/grid_map_pcl`).

Table 2.2: Qualitative runtimes of the point cloud to elevation map conversion. All the conversions were done on the same map with the size of 1419 by 1318 cells at 10 cm resolution.

Point cloud size	less than 10 million points	40-60 million points	100 - 140 million points
Algorithm runtime	1-2 minutes	5-15 minutes	30-60 min

2.5.2 Localization

Apart from mapping, the sensor module from Sec. 2.3.2 is used to localize HEAP within the map. The localization pipeline consists of components encircled in red color in Figure 2.8. The sensor module is mounted on the excavator (as shown in Fig. 2.4b). The sensor module's clock is synchronized

to the harvester's computer clock using Network Time Protocol (NTP). PTP and NTP run on different networks. We forward all the measurements to the main computer running Google Cartographer in the localization mode (see Hess, 2017).

Cartographer gives a pose estimate of the sensor module in the map frame used to compute the excavator's entire state. Extrinsic calibration of the complete sensor module mounted on HEAP is obtained from Computer Aided Design (CAD) model and manual measurement. Accurately calibrating sensors mounted on heavy machinery remains a challenging problem, and to increase overall localization accuracy one should use more advanced methods (e.g. API, 2021). Note that sensors on the sensor module itself are calibrated w.r.t. each other using techniques mentioned in Section 2.3.2. The setup with the sensor module mounted in the back resulted in end-effector position accuracy (coupled errors from sensor module localization and robot kinematics) of about 30 cm. For evaluation, we have asked the harvester to grab the same tree blindly multiple times. Such a level of accuracy may not be enough for high precision harvesting, and we mitigate the problem by detecting the grabbing target in a locally built map (see Sec. 2.6).

2.6 Tree Detection

The tree detection subsystem's responsibility is to detect a tree trunk and compute its position. n contrast to the methods mentioned in Chapter 2.2, our method is lightweight, suitable for online operation, and can be implemented in a dozen lines of code (available as open source[7]). While learning-based methods can still be fast to evaluate, we opted for a model-based method to avoid data labeling and for ease of implementation. Note that we are not interested in complete tree segmentation like most of the related work, but we are only interested in detecting a good grabbing spot. This circumstance allows simplifying the detection algorithm. A schematic of the subsystem with some intermediate processing steps is shown in Fig. 2.12. Once HEAP is positioned close to a tree, the state machine initiates a scanning maneuver to create a map of the scene in the local frame. Tree detection in the local frame is less affected by inaccuracies in the global localization system. It is worth noting that the tree detector also works on global maps and can be used to aid mission planning; this is discussed in Sec. 2.9.3.1.

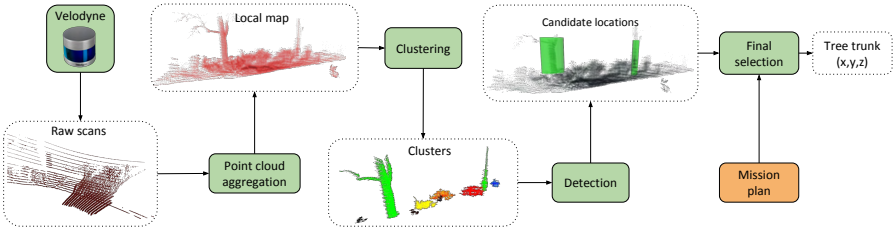
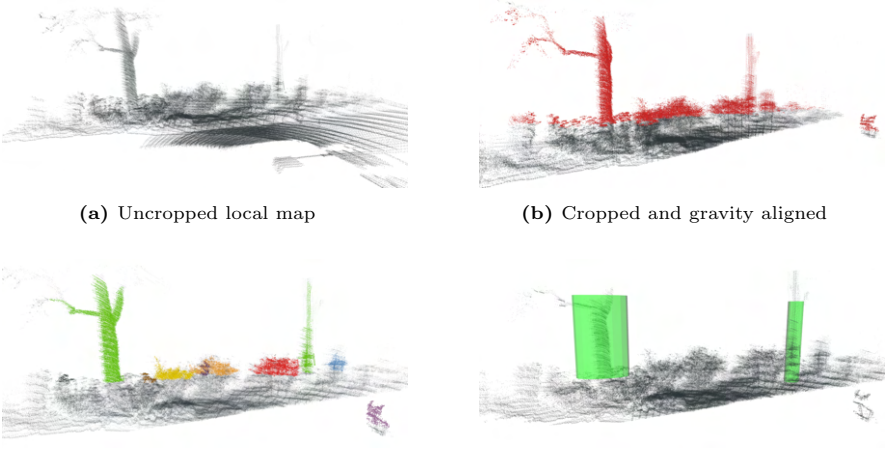


Figure 2.12: Flowchart of the tree detection subsystem procedure.

Tree detection starts with a scanning maneuver by rotating the cabin while the chassis is kept steady. The cabin is initially pointing in the direction where the harvester expects the target tree to be (known from the mission planner). The two LIDAR's mounted on the front of the machine (see Fig. 2.4a) stream the point clouds at 20 Hz. Note that the harvester's arm (or legs) might appear in the point cloud which is undesired. We use the *robot_self_filter* package from the ROS software stack to filter them out. The *robot_self_filter* uses robot's geometry (meshes or geometric primitives), state (joint angles, pose) and range sensor's extrinsic calibration to identify points belonging to robot's links in the point cloud. Subsequently, these points are filtered out.

[7] `https://github.com/leggedrobotics/tree_detection`

A receiver node stitches filtered point clouds together into a local map. The relative transformation between subsequent scans is recovered from the chassis's roll and pitch angle together with the cabin joint angle measurement. Note that in the absence of joint angle measurements or odometry, once could use point cloud registration methods (e.g. ICP). An example of a local map is shown in Figure 2.13a. The area scanned in Fig. 2.13a is somewhat larger than necessary to visualize steps in the tree detection algorithm better. We turn the cabin in its yaw angle $\pm 30°$ to scan the area and build a local map during the deployment. This corresponds to double the horizontal FOV of the tilted Velodyne LIDAR (see Fig. 2.4a) such that a dense local map can be built. We found that vertical Velodyne was more important for building dense maps, hence if one sensor is used, recommendation is to use it tilted.



(a) Uncropped local map

(b) Cropped and gravity aligned

(c) Clusters in the scene

(d) Trees and bounding ellipsoids

Figure 2.13: Intermediate results in the tree detection pipeline. ***Top Left:*** Scan of the area, without any cropping. One can observe the large ground plane in the point cloud. ***Top Right:*** Cropped point cloud after first and second cropping (red color). ***Bottom Left:*** Clusters in the scene. Each of these clusters is considered by the tree detection module. ***Bottom Right:*** Final tree detection with the resulting bounding ellipsoids.

The filtered scans from LIDARs are transformed into the cabin frame and cropped to speed up the subsequent steps. We use a box filter to crop the point clouds. The $x - y$ limits are set to drop all the points farther than the arm reach. Once cropped, the scans are transformed into a gravity-aligned frame where they are concatenated. The maximal density of the concatenated cloud is constrained to further speed up the computation. We use *libpointmatcher*

library for cropping, density filtering and concatenation Pomerleau et al., 2013. Point cloud assembled from cropped scans is shown in black in Fig. 2.13b (black). The assembled cloud is then gravity aligned and cropped again to filter out the ground plane and the tree crown (shown in red color, Fig. 2.13b). Again, we use a box filter to remove all the that are lower than the center of the highest wheel. The red point cloud is sent to the tree detection module, which selects all prominent trees in the scan, as shown in Fig. 2.13d.

The tree detection algorithm first computes Euclidean clusters in the input point cloud (shown in Fig. 2.13c). Euclidean clusters are searched using the algorithm from Rusu, 2010. We discard the clusters with too few points. Since most trees grow vertically, a point cloud of a tree trunk should have a majority of the points spread out along the z axis. Hence, Principal Component Analysis (PCA) is performed on each cluster, and we only keep clusters with a significant principal component along the z axis. Note that we can exploit the verticality assumption since our point cloud is gravity-aligned. The gravity alignment score ($\in [0, 1]$) is defined as the dot product of the largest principal component with a $[001]^T$ vector. Lastly, we check for the minimum height of the tree. The final detection result is shown in Fig. 2.13d. Sizes of bounding ellipsoids are computed based on principal components' eigenvalues in x and y direction. The final tree location is the ellipsoid's center. In the case of multiple tree detection (such as in Fig. 2.13d), the algorithm extracts coordinates of the tree closest to the expected tree position (from the mission planner).

2.7 Control

In this work, chassis control is responsible for locomotion and arm control for tree grabbing (harvesting). The respective control subsystems (shown in blue, Fig. 2.8) are described in more detail in this section.

2.7.1 Chassis Control

HEAP has four legs with wheels allowing it to drive and adapt to the terrain (see Fig. 2.14.) Thereby, the goal is to optimally distribute the four wheels' load to ensure traction and minimize terrain damage.

Terrain adaptation controller (also named Hip Balancing Controller (HBC)) is based on a virtual model control principle where the controller computes a net force/torque on the chassis to achieve the desired orientation (roll, pitch) and height. An optimal contact force distribution is computed from the net

force/torque for all the legs. Contact force tracking is achieved via force tracking on the hydraulic actuator level. For more details, refer to Hutter et al., 2016b. The described chassis controller can keep the base leveled while overcoming large irregularities in the terrain without getting stuck. Terrain adaptation is achieved using proprioceptive measurements only (joint sensing, IMU). We have extensively tested HBC performance in our previous work (Hutter et al., 2015, Hutter et al., 2016b). A video showing the machine overcoming challenging terrain can be found online[8].

(a) Chassis schematic **(b)** Chassis in action

Figure 2.14: ***Left:*** Illustration of the HEAP's chassis. The cabin and the arm are not shown for the sake of clarity. Steering joints axes are shown with blue arrows, flexion joint axes with red, and abduction joint axes with green color arrows. The path following controller actuates the steering joints while the HBC actuates the flexion joints. Abductions joints are not used. ***Right:*** Chassis control system overcoming a stump during the deployment.

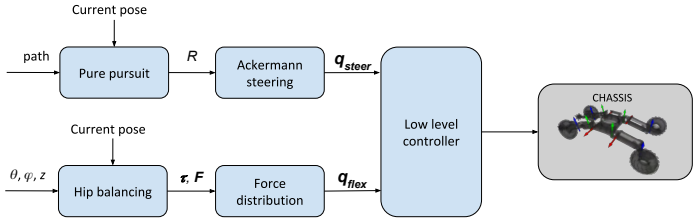


Figure 2.15: System diagram of the chassis control module deployed on HEAP.

For path tracking, we use a pure pursuit controller similar to the one outlined in Kuwata et al., 2008. The pure pursuit controller calculates the turning radius (denoted with R) required to bring the machine on the path computed by the planner. The Ackermann steering module then calculates steering joint

[8] https://youtu.be/5_Eq8CxKkvM

angles for each leg. Steering angles are computed such that all four wheels point to the same center of rotation with the radius R. The computation is implemented as a quadratic optimization that satisfies all the joint limits and finds a feasible center of rotation. For more details on the Ackermann steering module, the reader is referred to Jud et al., 2021b.

The HBC and the pure pursuit controller are combined to locomote the harvester, as shown in Fig. 2.15. Both controllers close the feedback loop over pose: HBC regulates the roll, pitch, and height, whereas pure pursuit ensures that x, y, and yaw are tracked correctly. The Ackermann steering module computes position reference for the steering joints ($\boldsymbol{q}_{steer}$ in Fig. 2.15), while the force distribution module computes joint torques assuming quasi-static conditions ($\boldsymbol{q}_{flex}$). The low-level controller translates the joint quantities (torques, positions) into the valve commands. In Fig. 2.14b, the proposed controller is driving over a stump. Note how the controller retracts the left hind leg to maximize the traction. In parallel, the pure pursuit controller controls the driving direction.

2.7.2 Arm Control

Reaching the end-effector target position determined from the tree detection module is achieved by following a trajectory from the planner described in Section 2.8.3. The trajectory following is done using an IK controller (see Siciliano et al., 2010) which uses the Hierarchical Optimization (HO), based on the implementation from Bellicoso et al., 2016. The main difference is that we tailor the tasks for HEAP instead of a quadruped robot. Besides tracking, HO computes joint velocities, enforces kinematic limits, and ensures that all flow constraints for hydraulic actuators are satisfied. The set of tasks optimized by the HO is given in Table 2.3, where *1* is the highest priority task. Opening and closing the gripper is controlled directly by the state machine and it is done in a purely open-loop fashion.

2.8 Planning

In this section we describe the mission planner and the motion planning stack in more detail. The motion planing stack is divided into three components shown in Fig. 2.8: base motion planner, grasp pose planner and the arm motion planner.

Table 2.3: Task priority inside the hierarchical optimization for the arm inverse kinematics controller.

Priority	Task
1	Equations of motion
2	Pump flow limit
3	Cylinder force limits
3	Cylinder velocity limits
3	Cylinder position limits
4	End-effector orientation
4	End-effector position

2.8.1 Mission planner

To conduct the experiments, we design a mission planner which determines which tree to grab next. Before the mission, a human manually selects the trees to be cut. Selection is accomplished using the Graphical User Interface (GUI) shown in Fig. 2.16, thus mimicking an algorithm for tree inventory management. When clicked on, a tree gets added to the tree list for harvesting with position coordinates extracted in the map frame. The mission planner passes the tree coordinates to the state machine in the same order as they were selected. A mission planner for optimizing some user-given objective and finding an optimal cutting order remains to be investigated in the future.

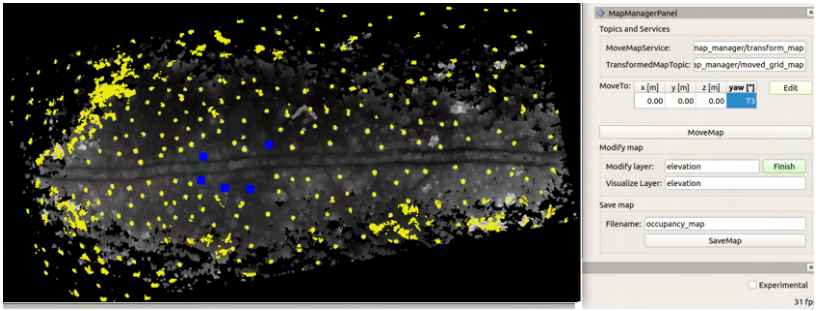


Figure 2.16: GUI used for mission planning. It is a panel in *Rviz* and it uses a *Qt* front-end. An elevation map of previously mapped environment is shown in gray-scale. All points higher than 2 m are colored in yellow; hence, the yellow color corresponds to the tree trunks and parts of the canopy. In this example, a total of five trees (marked with blue squares) are selected to be cut.

2.8.2 Base Pose Planning

We do not make any assumptions on the terrain properties; thus, our planner accounts for irregular terrain. Path planning in rough terrain is a complex problem, and in general, it has not been solved yet. In this work, we split it into two more manageable subproblems: planning in *SE(2)* (instead of *SE(3)*) and traversability estimation.

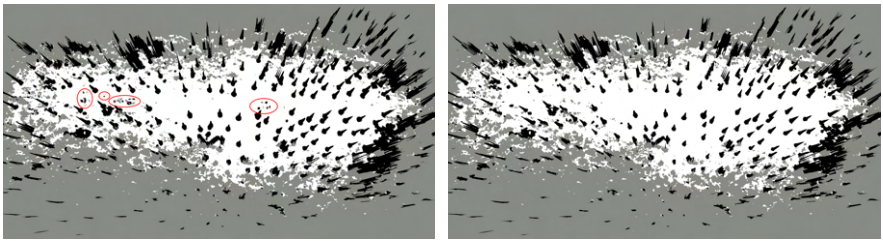
2.8.2.1 Traversability Estimation

The traversability estimation module discerns which areas can be driven versus which ones should be avoided. From the elevation map we compute a function $f : (x, y) \rightarrow [0, 1]$ that tells us for each coordinate (x, y) where HEAP can go (e.g. 1 - safe to drive, 0 - not safe). Traversability estimation has been previously studied, and different approaches for various sensing modalities exist. In this work, we use purely geometric traversability estimation, which is directly applied to elevation maps without any additional processing. We follow the approach presented in Wermelinger et al., 2016 that evaluates three criteria: terrain slope, roughness, and step height for local terrain patches (0.3 m radius). The final result is a weighted sum of all three components. We chose to mainly rely on step criterion (80 %) since the chassis control system introduced in Sec. 2.7.1 can overcome slopes and drive over rough terrain. The remaining 20 % was assigned to the slope criterion to prevent the harvester from driving on very steep slopes, which could result in slipping.

For storing the traversability map we use the grid map Fankhauser and Hutter, 2016a data structure. Fig. 2.17a shows the traversability map, a 2.5D map with a traversability layer; incorrectly classified areas (classified untraversable while it is traversable) are encircled red. Thick vegetation that occludes the ground from the LIDAR sensor during the mapping phase is the main culprit for false negatives. Since the overall misclassified area is small, a human can manually correct the errors. We used the GUI from Fig. 2.16 and the correction lasted about 2 minutes. Fig. 2.17b shows the corrected traversability map; note how the algorithm automatically classifies tree trunks as untraversable. Presented traversability estimation has an upside that is easy to implement; there is no need for elevation map processing or segmenting out the tree trunks explicitly. Fully automating the proposed pipeline would require additional work to eliminate the correction step and be implemented in future work.

The traversability map (see Fig. 2.17b) is converted into an occupancy map. Any traversability value smaller than 0.5 is deemed an obstacle, and higher

values are regarded as free space. Motion planners use the occupancy map, a 2.5D map with an occupancy layer.



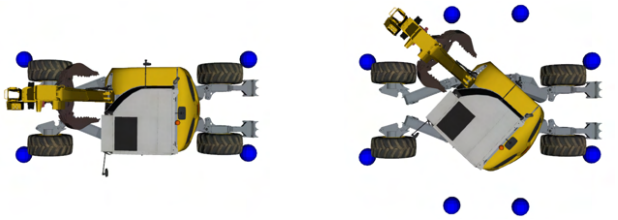
(a) Traversability Classification errors **(b)** Corrected traversability

Figure 2.17: Traversability maps computed by our approach shown against gray background. The maps shown are 2.5D maps with different layers (traversability layer displayed). White areas represent fully traversable terrain, whereas black areas represent terrain that is not traversable. ***Left:*** Red areas have been incorrectly classified as untraversable by the algorithm (false negatives). ***Right:*** Traversability map after applying the manual correction.

2.8.2.2 Base Approach-Pose and Path Planning

The planning subsystem (shown in yellow color in Fig. 2.8) gets a tree position from the mission planner. The tree position is an approximate target location for the end-effector. The harvester should not reach the tree position itself because this would result in a collision with the tree. To this end, we develop an algorithm for joint path and approach-pose planning. We evaluate the algorithm in both simulations and natural environments. Functionality described in this subsection corresponds to blocks *Approach-Pose Generation* and *Base Motion Planner* in Fig. 2.8.

Existing planning algorithms typically plan from starting pose to the goal pose. However, in our case, we do not know the goal pose (only approximate end-effector position in x and y). Hence, the proposed planning algorithm is split into two stages: first, the planner generates candidate poses and checks for their feasibility. Secondly, the planner attempts to compute a path to any of the feasible candidates. Thus the approach-pose planning is reduced to a common path planning problem, and we leverage a Rapidly-Exploring Random Tree (RRT)* algorithm for a joint path and approach-pose computation. Our implementation leverages Open Motion Planning Library (OMPL), Sucan, Moll, and Kavraki, 2012. We chose the OMPL because it is available as open-source and comes with efficient implementations of various sampling-based planners. We do not rely on optimization-based planners since forests



(a) Path planning footprint

(b) Approach-pose planning footprint

Figure 2.18: Top view of HEAP with the arm retracted with the footprint vertices shown in blue. Note: not drawn to scale. ***Left:*** Footprint used for path planning (see Alg. 2). ***Right:*** Footprint used for the approach-pose generation (see Alg. 1) is wider in the middle, thus allowing for cabin turning.

are cluttered environments with many local minima, which present a challenge for optimization.

The approach-pose generating subroutine is shown in Alg. 1. It starts by getting a target/tree location $\boldsymbol{x}_T \in \mathbb{R}^2$ (line 1), and computing a set of approach-poses around it (lines 18 - 31). We compute $M \times N$ approach positions in polar coordinates around $\boldsymbol{x}_T$ by pairing M different distances to the tree with N uniformly distributed polar angles. Lastly, the approach positions are paired with K yaw angles (lines 20-22) to generate a total of $N \times M \times K$ candidate approach-poses in *SE(2)*. To be a valid candidate, an approach-pose must be collision-free (lines 6-7), the harvester's arm must be able to reach the target (lines 6-9), and all heuristics must be satisfied (lines 10-11).

The blue points shown in Fig. 2.18 represent the harvester's collision footprint. The harvester is in a collision if there is an obstacle inside the blue points convex hull. Collision checks inside the Alg. 1 use the footprint shown in Fig. 2.18b. Note how the hull is wider in the middle to allow for cabin turns. The real-world map is abstracted away from the planner, and it only sees the obstacles computed by the traversability estimation step. Such a simplification is warranted by using a control system able to overcome slopes and rough terrain.

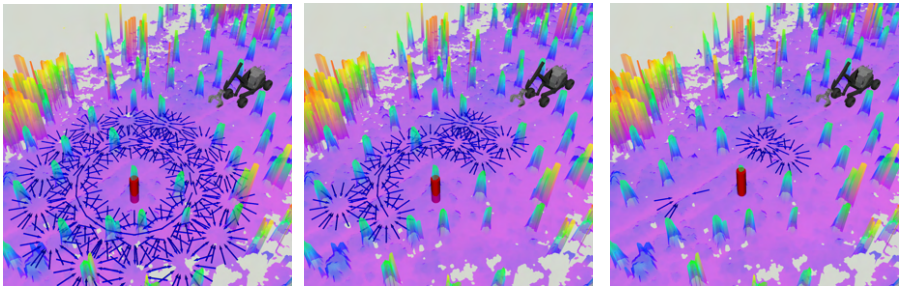
The target being reachable from an approach-pose means that the harvester can safely extend the arm to reach the goal position (see Alg. 1, line 9). We check whether there is a collision-free line of sight from the base to the target goal. Because of HEAP's kinematic structure, the arm always stays within a slab in the $x - z$ plane (in the cabin frame). Hence, we require no obstacles

Algorithm 1 Candidate approach-pose generator

$x_T, y_T \leftarrow$ getNextTargetLocation()
$isUseHeuristic \leftarrow$ readFromConfigFile()
procedure computeCandidateApproachPoses(x_T, y_T)
    $candidateApproachPoses \leftarrow$ getApproachPosesAroundTarget(x_T, y_T)
    for each *pose* **in** *candidateApproachPoses* **do**
        if isInCollision(*pose*) **then**
            candidateApproachPoses.delete(*pose*)
        end if
        if ¬isTargetReachableFrom(*pose*) **then**
            candidateApproachPoses.delete(*pose*)
        end if
        if *isUseHeuristic* AND ¬isHeuristicValid(*pose*) **then**
            candidateApproachPoses.delete(*pose*)
        end if
    end for
    return *candidateApproachPoses*
end procedure
procedure getApproachPosesAroundTarget(x_T, y_T)
    $distances \leftarrow \{d_1, d_2, ..., d_M\}$
    $polarAngles \leftarrow \{\phi_1, \phi_2, ..., \phi_N\}$
    $headings \leftarrow \{\psi_1, \psi_2, ..., \psi_K\}$
    $approachPoses \leftarrow \emptyset$
    for each d_i **in** *distances* **do**
        for each ϕ_i **in** *polarAngles* **do**
            $(x_i, y_i) \leftarrow (x_T + d\cos(\phi_i),\ y_T + d\sin(\phi_i))$
            for each ψ_i **in** *headings* **do**
                approachPoses.ADD($[x_i, y_i, \psi_i]$)
            end for
        end for
    end for
    return *approachPoses*
end procedure

inside the slab spanned by the target tree position and the base position. Therefore, there is no need for more complicated algorithms that are popular in mobile manipulation literature (e.g., Zucker et al., 2013). An example of the approach-pose generation in a forest environment is shown in Fig. 2.19.

In the second stage, the planner checks whether the feasible approach-poses are attainable, i.e., can the harvester drive to them. Note that a feasible approach-poses such as the red one in Fig. 2.20 is not necessarily attainable. The subroutine for attainability checking is summarized in Alg. 2. We build upon a standard RRT* planner that grows a random tree, as described in Karaman and Frazzoli, 2011. Instead of trying to connect the single goal pose (as standard RRT*) to the rest of the tree, we attempt to connect every can-



(a) All approach-poses **(b)** Goal target reachable **(c)** Goal reachable & no collision

Figure 2.19: Approach-pose generation in the forest environment. The elevation map with the target tree (red cylinder) and candidate approach-poses are shown with blue arrows. Some approach-poses have been omitted for the sake of clarity. ***Left:*** In total, 450 goal approach-poses are generated. So far they have not been checked for feasibility and this example we do not apply any heuristics. ***Middle:*** Out of 450 poses 155 do not satisfy the target reachability criterion (line 8 in Algorithm 1). The arm can extend and grab the target tree from 195 remaining poses. ***Right:***. Out of 195 poses, 168 of them are in collision with the environment (line 6 in Algorithm 1). The remaining 27 approach-poses both satisfy the reachability criterion and are not in collision. These remaining 27 approach-poses are then sampled inside the RRT* to determine which ones are attainable.

didate approach-pose. Planning terminates when the allotted planning time runs out. We found 5 s to be an adequate compromise between computation time vs mission progress time. The footprint used for collision checking inside Alg. 2 is shown in Fig. 2.18a. As a steering function inside the RRT*, we use the Reeds-Shepp (RS) curves Reeds and Shepp, 1990 which allows us to respect the minimal turning radius constraint (just like any car, HEAP cannot turn in place). The turning radius parameter was set to 8.3 m.

The rationale behind the planner's second stage is that an attainable approach-pose has a high probability of having a collision-free connection to the rest of the random tree. For example, the green pose is a better approach-pose compared to the blue one in Fig. 2.20. The environment around the green pose is less cluttered, so the probability of a successful connection is higher. A beneficial feature of the described approach (Alg. 2) is outsourcing the final approach-pose selection to the RRT*. Note that using the RRT* in its standard form would require the user to pick an approach-pose, a challenging task since we do not know which ones are attainable *a priori*. Since fewer approach-poses ensure faster convergence, the proposed framework allows the use of heuristics for pruning the approach-pose candidate set. Pruning heuristics can

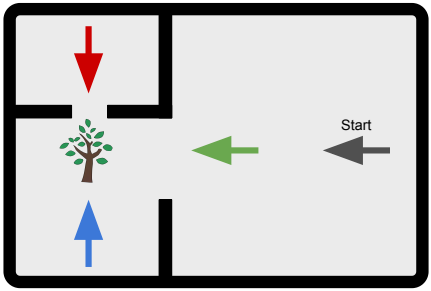


Figure 2.20: Example environment for approach-pose planning. All approach-poses are feasible; however, the red pose is not reachable from the start. The green pose is better than the blue pose since it is easier to reach.

be a wide array of constraints and rules, thus allowing for great flexibility (e.g., discard approach-poses where heading changes more than 90°).

Algorithm 2 Path and approach-pose planner

procedure computePathAndApproachPoses(x_T, y_T)
  $x_T, y_T \leftarrow$ getNextTargetLocation()
  $\boldsymbol{p}_s \leftarrow startingPose$
  rrt.initialize($\boldsymbol{p}_s$)
  $candidateApproachPoses \leftarrow$ computeCandidateApproachPoses(x_T, y_T)
  while $t_{cpu} < T_{max}$ **do**
    rrt.growTree()
    for each $\boldsymbol{p}_i$ **in** $candidateApproachPoses$ **do**
      rrt.connectToTree($\boldsymbol{p}_i$)
    end for
  end while
  end procedure
  return ($\boldsymbol{p}_i, rrt$.getPath())

The proposed approach-pose planning takes into account almost all DoFs that HEAP has to offer. It allows the machine to turn the arm and approach trees from any angle. Furthermore, the harvester can utilize both driving directions when navigating to the goal target. One could still improve the approach-pose generation to anticipate the arm turning direction. Algorithm 1 uses collision footprint shown in Fig. 2.18b which is clearly conservative since the arm doesn't have to make a full 360° turn. Anticipating the turning direction would allow shrinking the collision footprint, which is very beneficial in cluttered environments such as one shown in Fig. 2.19. Another possible improvement is to adapt the number of approach-pose candidates based on

the environment. Generating too many approach-pose candidates slows down the planning while marginally contributing to finding better solutions when the obstacle density is low.

2.8.3 Arm Grasp Pose and Motion Planning

The grasp pose planner receives a tree position from the tree detection subsystem and computes the desired gripper pose. Functionality in this section corresponds to *Approach-pose generation* and *Base motion planner* blocks in Fig. 2.8. Since for our demonstration, we use a gripper instead of a standard tree cutting tool (such as Menzi Muck, 2020), we emulate the same behavior by fixing the roll and pitch of the gripper and by computing the yaw angle such that the cabin faces the tree. The kinematic properties of HEAP and harvester machines in general with an arm that only moves in a plane allow us to come up with a simple arm planning algorithm. To reach the desired grasp pose, we design a three-stage maneuver that requires minimal space:

1. Retract the arm
2. Turn the cabin
3. Extend the arm

The approach-pose planner (see Sec. 2.8.2.2) ensures that there is enough space for the whole arm maneuver. An exemplary arm plan is shown in Fig. 2.21 (stages of the maneuver are indicated with numbers). Intermediate poses (waypoints) are visualized with coordinate systems. We compute Hermite polynomials between the adjacent waypoints to form a trajectory, and we limit the average linear and angular velocity along the spline. The Hermite polynomial tracjecory is then tracked using the inverse kinematics controller, as described in Section 2.7.2.

For tree felling, the proposed planning method would have to be augmented to consider tree geometry when the harvester is holding one. This could be achieved by adding visual sensors and tracking the falling tree such that the planner can optimize the pulling direction. Besides, one could also estimate the weight and adapt the arm controller tuning or add a feedforward command. The current arm controller is robust concerning weight changes in the gripper. We experimented with stone stacking in Johns et al., 2020 and we were able to manipulate stones between 400 kg and 2400 kg (1040 kg on average) which is enough to harvest a small tree. Lastly, one could extend approaches such as

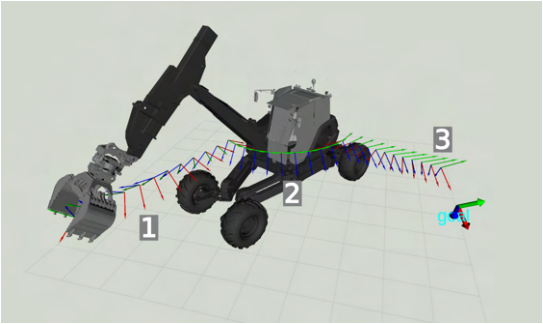


Figure 2.21: Plan for the arm end effector. Intermediate poses are visualized with coordinate systems. Each axis (x, y, z) corresponds to a different color (red, green, blue). A thick coordinate system and "goal" sign denote the final desired pose. Numbers correspond to different maneuver stages: 1) retract, 2) turn, 3) extend.

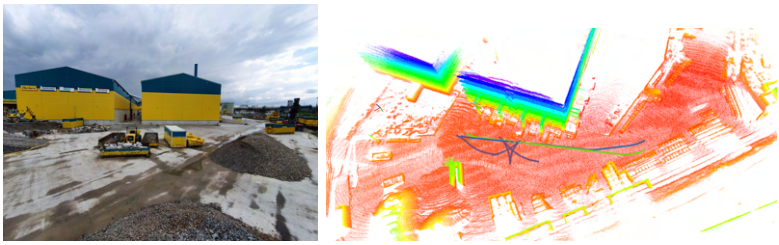
Song and Sharf, 2020 to prevent the machine from tipping over when holding a tree.

2.9 Results

The presented system components are individually tested and evaluated for complete harvesting missions in the forest and on our testing field. It is worth noting that we implemented the whole system in simulation first to catch as many mistakes as possible before field testing. As the simulation environment, we use *Gazebo* (with ROS integration) which is based on the Open Dynamics Engine (ODE) physics engine. For visualizing (e.g. trajectories, point clouds) we rely on *Rviz*, a ROS tool for visualization.

2.9.1 Path Tracking

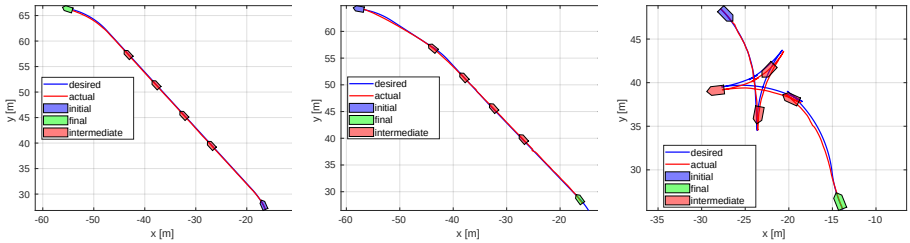
We evaluated the tracking performance of the proposed control approach in Oberglatt (Switzerland), where HEAP was located for another project at the time of writing this paper. The site is shown Fig. 2.22; it is a construction site with a mix of concrete and gravel surfaces. We evaluate the tracking performance using RTK GPS to measure HEAP's position accurately. HEAP was asked to track three different types of paths which consisted of: mostly forward driving, mostly backward driving, and tight maneuvering. An example of the forward driving path is shown in Fig. 2.23a, the backward driving path is shown in Fig. 2.23b and tight maneuvering path is shown in Fig. 2.23c. The



(a) Oberglatt site **(b)** Site map with overlaid paths

Figure 2.22: Testing site where tracking experiments were performed. The image and the map were taken few weeks apart. ***Left:*** Photo of the testing site where HEAP was located at the time of writing this paper. We were allowed to drive on the site to test the tracking controller's performance. ***Right:*** Map of the testing site with some example plans overlaid. The green path corresponds to the one in Fig. 2.23a, the blue path can also be seen in Fig. 2.23b and the purple one can be seen in Fig. 2.23c.

maneuvering scenarios require the machine to change its orientation in a tight space. Hence, the planner comes up with paths containing cusps and tight turns; we asked HEAP to change its orientation (heading) for either 90° or 180°.



(a) Forward driving **(b)** Reverse driving **(c)** Tight maneuvering

Figure 2.23: Example trajectories from tracking accuracy evaluation. Blue color shows output from the planner; red color denotes the actual tracked path. HEAP's initial orientation is shown in blue color and the final one in green. The size of the vehicle is not drawn to scale. ***Left:*** HEAP achieved 9 cm average tracking error over 55 m of forward driving distance. ***Middle:*** HEAP achieved 13 cm average error over 57 m of backward driving distance. ***Right:*** HEAP achieved an average error of 18 cm over 62 m distance for tight maneuvering scenario. In this particular example, HEAP does 7 cusps (changes of driving direction).

Our controller only tracks the path in $x - y$ coordinates, and hence, all the errors are computed in $x - y$ coordinates. The controller does not track the z coordinate since the planner plans in *SE(2)*. The tracking performance was evaluated over about 30 trials and about 1.5 km of driving. We summarize

the tracking performance in Table 2.4. Across all trials, the path tracking algorithm achieves the tracking error of about 17 cm. We note that the tracking errors achieved in the Oberglatt site are somewhat optimistic since tests took place on flat surfaces that have good traction. It is to be expected that the tracking performance deteriorates as the surfaces get more slippery. This effect we illustrate in a set of experiments conducted on our test field.

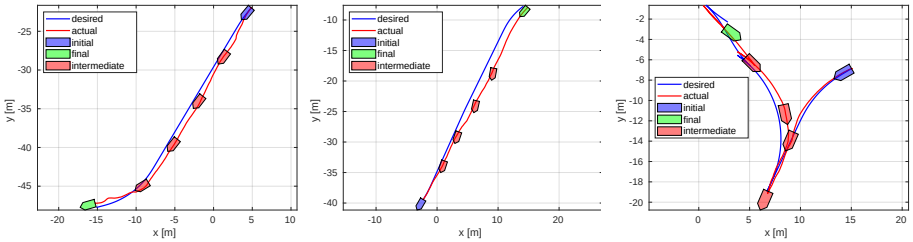
Table 2.4: Path tracking performance evaluation in Oberglatt site. All the errors are transitional errors in $x - y$ coordinates. *Avg length fwd* denotes the average distance traveled forward during the maneuver (analogously for the backward distance, *avg length bck*). *Traveled total* is the cumulative distance traveled across all trials.

scenario	num trials	avg track error [m]	standard deviation [m]	avg length fwd [m]	avg length bck [m]	avg num cusps	traveled total [m]
fwd driving	10	0.23	0.04	56.05	0.52	0.6	565.84
bck driving	8	0.16	0.04	0.76	55.93	0.75	453.56
maneuvering	14	0.13	0.03	17.99	17.44	2.93	496.19
total	32	0.17	0.02	24.93	24.63	1.43	1515.59

(a) test field SE view **(b)** aerial view

Figure 2.24: Photos of our testing field, not all images were taken at the same time. ***Left:*** View looking southeast with the machine parked. The area shown is flat. ***Right:*** Aerial view of the field with different classes of terrain labeled. We overlay the planned (blue) and executed (red) path from one tracking experiment (see Fig. 2.25a).

The testing field is shown in Fig. 2.24. HEAP was commanded to follow three different paths shown in Fig. 2.25. Fig. 2.25a and Fig. 2.25b show tracking a long path on an inclined terrain (see Fig. 2.24b). A combination of inclined and wet terrain caused the HEAP to slide sideways. This can also be observed in the path visualization: the heading is not tangential to the path. The machine is pointing towards the slope to compensate for sliding to the side. In Fig. 2.25c we asked HEAP to reorient itself on the flat part of the testing field.



(a) Forward on lateral slope **(b)** Reverse on lateral slope **(c)** Turns and cups on even ground

Figure 2.25: Experiments were conducted to determine the tracking accuracy of our chassis control approach. In blue color, the planner's desired plan has been shown; red color denotes the actual tracked path. The initial orientation of the excavator is shown in blue color and the final one in red. The size of the vehicle is not drawn to scale.

The chassis controller was able to achieve mean absolute tracking error of 0.461 m over 34.1 m distance for the run shown in Fig. 2.25a and for the run shown in Fig. 2.25b the error was 0.684 m over distance of 44.28 m. While performing reorientation on the flat terrain, the mean absolute error was 0.222 m over distance traveled of 44.48 m (with all direction changes). One can observe that inclined terrain presents a bigger challenge for the tracking controller than direction changes. We note that the test field was fairly wet during the experiments, which negatively affected the amount of traction available. Forestry operations are typically performed when the ground is either dry or frozen, which reduces these slipping issues. Hence, the set of results from our test field presents a hard case scenario which is not often encountered in regular harvesting operations.

2.9.2 Planning

We evaluate the approach-pose planning pipeline proposed in Sec. 2.8.2.2 in simulated scenarios and in two experiments on different terrain: test field and forest patch. We ask the planner to compute both path and an approach-pose to grab the selected target trees. All the map visualizations in this section have been created using *Rviz*.

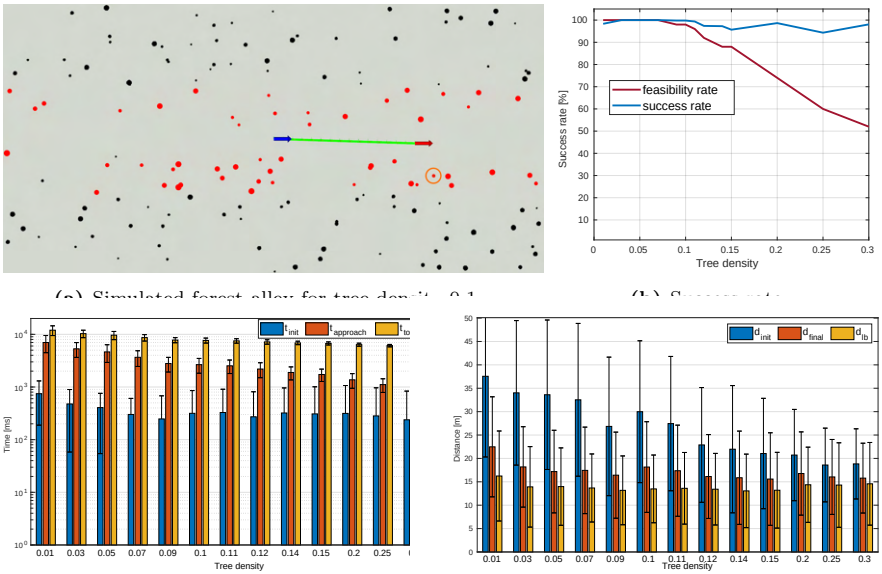
We created a simulated forest by sampling the number of trees from a Poisson distribution and their positions and radii from a uniform distribution. We consider two scenarios: in the first one, there is a forest alley that can be used for driving (see Fig. 2.26a). This resembles the situation encountered during the field experiments. There is no forest alley in the second scenario, and the

harvester has to navigate between the trees (see Fig. 2.27a). Such a scenario is common when using a smaller machine such as one shown in Fig. 2.2a. For each forest density, we run 10 planning trials and average the metrics. Within one trial, we to compute plans for several target trees; the number of trials is shown in Table 2.5.

For the forest alley scenario, we ensure the alley at least 2.8 m wide. For comparison, the harvester path planning footprint has a width of 2.4 m and the approach-pose planning footprint is 4.8 m wide at its widest point. Note that as the forest gets denser, there might not be enough space to turn the cabin, and therefore no feasible approach-poses. We choose several trees (max 50) at random within the 6 m distance from the forest alley middle (HEAP has a reach of about 8 m) to be the targets. Black dots represent trees while target trees are colored red in Fig. 2.26a. Each target is deemed feasible if we can find a path to it within 30 s of planning. An example path is shown in green; the starting pose and the planned approach-pose are denoted with a blue and red arrow, respectively. In this particular example, the plan requires the harvester to drive forward and turn the cabin to grab the tree encircled orange. Quantitative evaluations of the planner are shown in Fig. 2.26, we compute the metrics only for feasible targets.

The percentage of feasible targets (see Fig. 2.26b) drops as the forest's density grows. However, one can see that the planner maintains a high success rate. Besides the success rate, we evaluate times required to generate candidate approach-poses $t_{approach}$, time until the first solution inside the RRT* planner t_{init} and we show the total planning time t_{total} (Fig. 2.26c). One can observe that the RRT* planner finds the first solution rather quickly (worst case in 700 ms). The most expensive part of the pipeline is the approach-pose generation which in the worst case takes about 7 s. Planning times tend to shorten as the forest density (number of trees per m^2) increases since many approach-poses can be discarded in the early stage of collision checking (early termination). In contrast, for low forest densities, almost all approach-pose footprints have to be checked for collisions fully (no early termination).

For the experiments presented, the planner considers 14060 approach-pose candidates in total (no pruning heuristics were applied to showcase the generality of the approach). In this work, we use single-threaded implementation; however, the approach-pose generation can be easily paralleled to decrease computation time. Lastly, we measured the distances between the starting pose and the planned approach-poses (see Fig. 2.26d). We show the length of the first found path d_{init} and the length of the optimized path d_{final} (after



(c) Planning times

(d) Path lengths

Figure 2.26: Joint path and approach-pose planning for the forest alley scenario. Tree density is defined as a number of trees per m^2. ***Top Left:*** Top view of the random forest with the path and approach-pose planned. The harvester starts from the blue arrow to grab the tree encircled orange. Path is shown in green and the final approach-pose in red. Black dots represent the trees and red dots are the remaining target trees. ***Top Right:*** Percentage of feasible targets and the success rate as the forest density increases. Success rate is calculated as number of successful planning attempts divided by the number of feasible targets. ***Bottom Left:*** Planning times against varying forest density, note the logarithmic scale on the y axis ***Bottom Right:*** Path lengths as a function of varying forest density.

running RRT* for 5 s). Lastly, the graph shows Euclidean distance between start and finish d_{lb} which is a lower bound on the path length. We can see that the planned path is close to the lower bound in all cases. The paths tend to shorten with the increasing density since more feasible trees lie on the inside of the forest alley. Hence, HEAP does not turn to the side, which shortens the overall path length.

As a second scenario, we evaluated planning performance while navigating an unstructured forest. The forest was generated using the same probability distributions as in the scenario above. We leave a clear area (10 m by 10 m) in the middle of the map to ensure a feasible starting pose. The target trees were

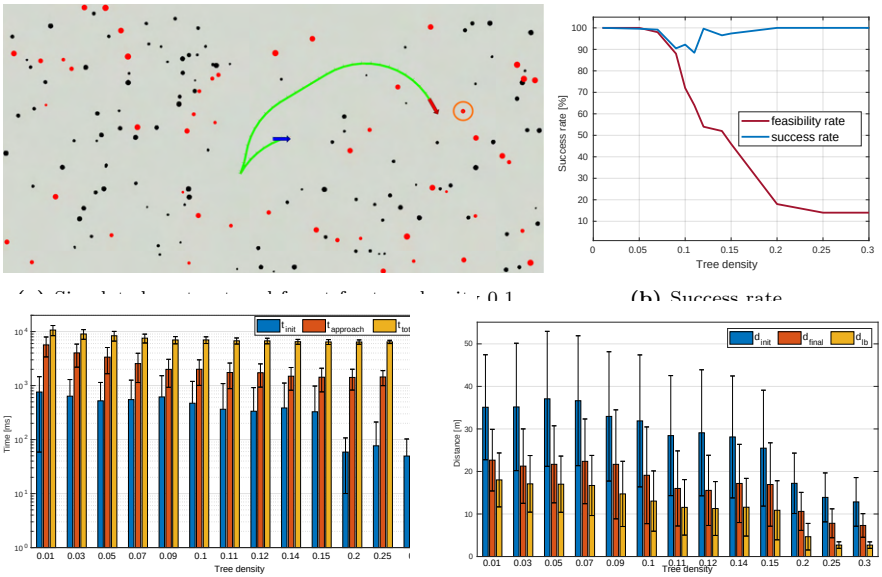
Table 2.5: Number of feasible (attainable) targets together with the total number of planning attempts for each forest density. We show the data for forest alley and unstructured forest scenario.

	forest alley			unstructured forest		
density	n targets	n feasible targets	n (successful) planning attempts	n targets	n feasible targets	n (successful) planning attempts
0.01	6	6	(59) 60	23	23	(230) 230
0.03	18	18	(180) 180	50	50	(499) 500
0.05	23	23	(230) 230	50	50	(498) 500
0.07	38	38	(380) 380	50	49	(486) 490
0.09	50	49	(489) 490	50	44	(398) 440
0.1	50	49	(489) 490	50	36	(332) 360
0.11	50	48	(477) 480	50	32	(283) 320
0.12	50	46	(448) 460	50	27	(269) 270
0.14	50	44	(428) 440	50	26	(251) 260
0.15	50	44	(421) 440	50	23	(224) 230
0.2	50	37	(365) 370	50	9	(90) 90
0.25	50	30	(283) 300	50	7	(70) 70
0.3	50	26	(255) 260	50	7	(70) 70

selected at random with a maximum of 50 trees. Again, we check the feasibility for each target by running the planner for 30 s and perform 10 planning trials.

Performance measures shown in Fig. 2.27b - Fig. 2.27d are calculated only for feasible targets. One can observe a much faster drop in the percent of feasible targets (compared to the forest alley scenario). In most cases, our planner finds the solution rather quickly (800 ms in the worst case). We can also notice that the success rate drops compared to the previous scenario since there is no forest alley. The planner has to find a path through narrow passages (hard for sampling-based planners). The forest density of 0.1 seems to be especially hard, i.e. the harvester can still fit through the trees, but just barely. The success rate for densities between 0.05 and 0.15 can be increased with longer planning times. For forest densities of 0.15 and higher the HEAP harvester simply dos not fit between the trees. Hence the set of feasible targets comprises the trees around the clear patch in the middle of the simulated forest. In this case, the success rate goes again to 100 %.

To evaluate the planning performance under realistic conditions, we planned on a map of our testing field and a map of forest patch where we conducted the tests. The forest patch map with target trees is shown in Fig. 2.28. Black dots represent trees and other obstacles, while target trees are denoted with red dots. We manually selected the trees along the forest alley, discarding any tree without path computed within 30 s of planning. The forest density at the testing site was estimated to be about 0.14 (trees/m^2). The planner



(b) Success rate

(c) Planning times

(d) Path lengths

Figure 2.27: Joint path and approach-pose planning for the unstructured forest scenario. ***Top Left:*** Top view of the random forest with the path and approach-pose planned. See Fig. 2.26 for explanations of the colors in the image. ***Top Right:*** Success rate as the forest density increases. Feasibility rate and success rate are calculated the same way as for Fig. 2.26. ***Bottom Left:*** Planning times against varying forest density, note the logarithmic scale on the y axis ***Bottom Right:*** Path lengths as a function of varying forest density.

achieved a success rate of about 98 % which is almost the same as the simulated scenario (95 %). Furthermore, the time $t_{approach}$ (1625.15 ms) is close to the simulated one (1728.75 ms). t_{init} (120.62 ms) is lower than the simulated one (308.36 ms); which would indicate that it has fewer narrow passages than the simulated forest.. On the testing field, the planner achieved the success rate of 100 % (omitted for the sake of brevity).

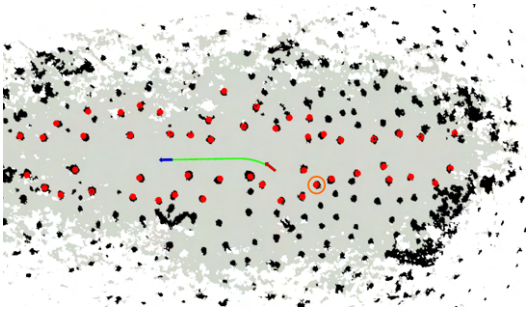


Figure 2.28: Obstacle map for the forest patch scenario. Obstacles are shown in black color whereas free space is shown in gray color. The targets are denoted with red cylinders

Table 2.6: Planning metrics for the forest patch scenario (averaged over 10 trials).

num targets	51
success rate	0.98
time until first solution [ms]	(120 ± 555)
approach-pose generation time [ms]	(1625 ± 621)
total planning time [s]	(6.6 ± 0.6)
initial path length [m]	(18.38 ± 10.86)
optimized path length [m]	(14.3 ± 9.84)
length lower bound [m]	(13.53 ± 9.48)

2.9.3 Tree Detection

We evaluated tree detection offline by mimicking local maps assembled from harvester sensors during the scanning maneuver. We select 6 m by 6 m patches inside the map shown in Fig. 2.9a and ask the tree detector to detect all trees inside the cropped map. The patch size roughly corresponds to the area covered by vertical Velodyne sensor frustum after the scanning maneuver. The detection procedure discarded clusters with diameter bigger than 2.5 m, fewer than 1000 points or with gravity alignment score less than 0.8 (as defined in Sec. 2.6); the result is shown in Fig. 2.29. One can observe successful tree detections even in the presence of vegetation. Heavy clutter in the scene, such as in patch five and patch twelve, causes the detector to reject segmented tree clusters, incorrectly producing false negatives. In case no trees are detected close to the target, the harvester resorts to blind grabbing at the target location (known *a priori* from the map). We evaluated the detection accuracy on three different forest patches (the ones shown in Fig. 2.32b, Fig. 2.32c and Fig. 2.37a). For evaluation, we selected 6 m by 6 m patches (about 30 of them) at random from each map and run the tree detection on them. The true number of trees was determined manually by inspecting the point cloud patch. The quantitative evaluation is shown in Table 2.7. The fail cases and extended analysis for the tree detector are presented in the Appendix 2.12.2.

2.9.3.1 Tree Detector for Mission Planning

In this section, we present the use of a tree detector as a mission planner. Although the tree detector was designed to be used on small local maps, it can also be used to extract tree coordinates for clear-cutting missions or as a

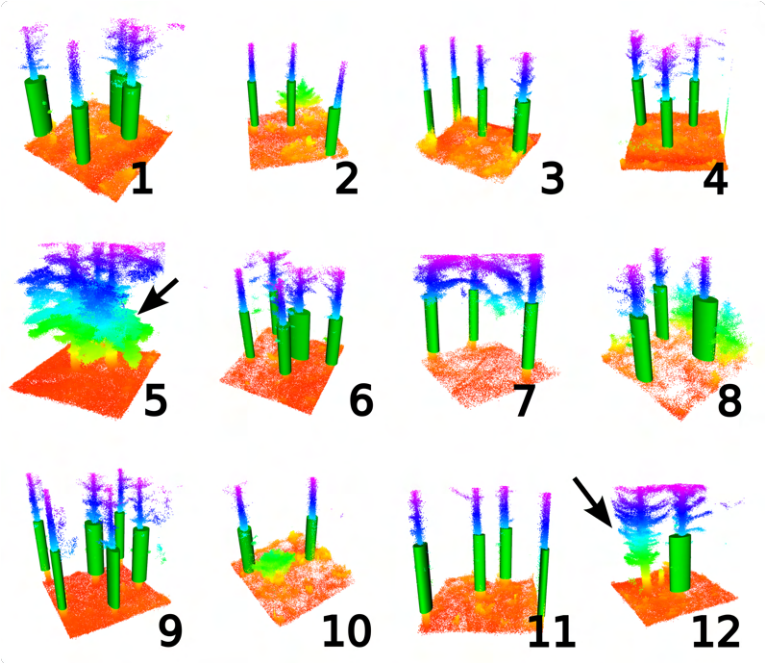


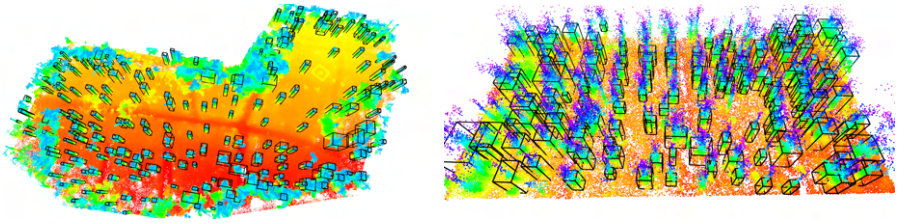
Figure 2.29: 6 m by 6 m point cloud patches from the testing site. Red corresponds to the lowest and purple to the highest elevation. Tree detections are shown with green cylinders. Tree detection has failed for snapshots 5 and 12 (marked with black arrows).

mission planning aid. We ran the tree detection offline on three forest patches of different styles, all shown in the figures below. The ground plane was filtered out, and the procedure presented in Sec. 2.6 was used. We manually identify the trees in each point cloud to obtain the actual number of trees.

Fig. 2.30a shows the first forest patch together with tree detections. The patch has about 246 (both evergreen and deciduous) trees, and the tree detector correctly detected 210 of them. There were 14 false positives, and 36 trees were not detected (false negatives), which amounts to a precision of 94 % and recall rate of about 85 %. The point cloud of the second forest patch and detected trees are shown in Fig. 2.30b. In total, there are 163 (spruce) trees, and the tree detector correctly detected 143 of them. There were six false positives, and 20 trees were not detected, which amounts to a precision of 96 % and recall rate of about 88 %. The last forest patch is shown Fig. 2.31. The Forest patch has about 285 trees (all deciduous trees). The tree detector detected in

Table 2.7: Tree detection evaluation.

map	num trials	num trees	recall	precision
Fig. 2.32b	32	4.13 ± 1.91	0.88 ± 0.2	0.92 ± 0.16
Fig. 2.37a	33	4.78 ± 2.39	0.82 ± 0.25	0.94 ± 0.11
Fig. 2.32c	32	3.28 ± 1.71	0.95 ± 0.14	1
total	97	4.06 ± 1.17	0.88 ± 0.12	0.95 ± 0.07



(a) Mixed forest with detected trees

(b) Spruce forest with detected trees

Figure 2.30: Maps and tree detections for two forest patches. Detected trees are denoted with black bounding boxes. Red/yellow color corresponds to the lowest elevation and blue/purple to the highest. ***Left:*** Forest with both evergreen and deciduous trees. Note the heavy clutter towards the edges of the map. ***Right:*** Forest with mainly spruce trees.

total 266 trees. Four detections were false positives, whereas the number of false negatives was 19. These numbers give the tree detector precision of 98 % and recall rate of 93 %. For a pointcloud with 26 million points, it takes our method about 300 s to complete.

We want to conclude that using our tree detector as a mission planner may yield inferior results than other methods. However, the strong points of our method are that it is lightweight, available as open-source, and can be used online on the robot. For tree detection on large forest areas, one is better off using some of the more advanced approaches, such as Burt, Disney, and Calders, 2019 which achieve recall rates up to 96 % on tropical forest datasets. One should note, however, that the method is computationally expensive; it takes two days to process a map with 17 million points on a computer with 24 cores. Some of the best results that we have seen have been obtained using the forestry module inside the LiDAR360 commercial software for pointcloud processing. Unfortunately, LiDAR360 is not available for free.

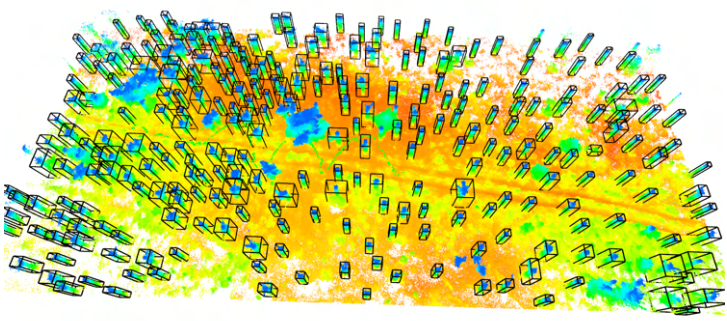
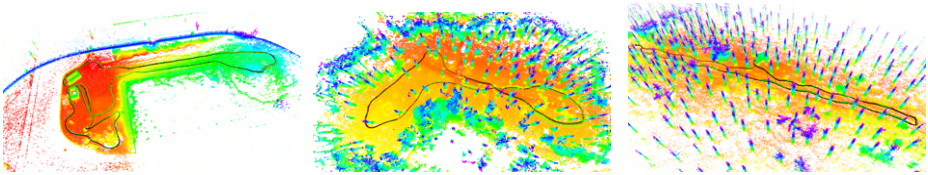


Figure 2.31: Forest patch and detected trees denoted with black bounding boxes. The forest patch consists of deciduous trees.

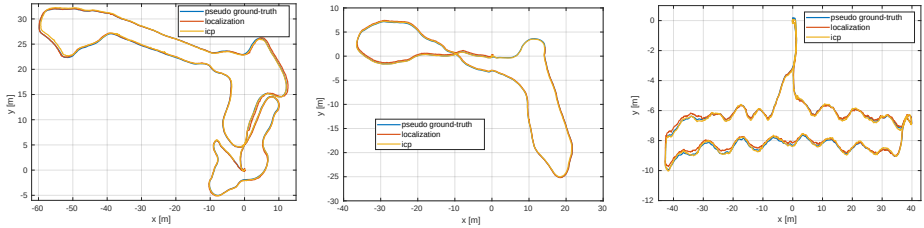
2.9.4 Localization

We evaluated the localization of the proposed sensor module running Google Cartographer SLAM in three different terrains: two forest patches and our testing field. We collect the dataset with the sensor module presented in 2.3.2, and we run the SLAM offline, which produces a consistent, optimized map. Subsequently, we set the Cartographer in the localization mode and tune it such that both the LIDAR odometry and loop closures run in real-time. Note that Cartographer running in the localization mode does not try to build a global map (see Hess, 2017). We are interested in quantifying performance degradation when running in real-time localization mode without joint map and trajectory bundle adjustment. We localize in a previously built map and compare the localized pose against the bundle adjusted trajectory (pseudo-ground-truth). Paths overlaid inside the map can be seen in Fig. 2.32 while the top view plots of the trajectories can be seen in Fig. 2.33.



(a) Testfield **(b)** Forest 1 **(c)** Forest 2

Figure 2.32: The trajectories from the localization field experiments overlaid with the map.



(a) Testfield **(b)** Forest 1 **(c)** Forest 2

Figure 2.33: The trajectories from the localization field experiments viewed from above. Forest 1 and Forest 2 are datasets from two different forests.

We evaluate the localization error as the euclidean distance between the corresponding points on the localized trajectory and pseudo ground-truth trajectory. For the test field, we obtain a translational error of 0.11 ± 0.09 meters; the trajectory length was 255.8 m. Trajectory length for the first forest dataset was 158.37 m, and Cartographer localizes with the error of 0.17 ± 0.28 meters. Inside, the second forest trajectory was 183.27 meters long, and the localization error was 0.25 ± 0.28 meters. The presented evaluation is biased since the cartographer (the same method) is used for both map building and localization. Hence, we evaluate localization performance using a different method, the ICP (using point to plane error metric). The ICP is used to register LIDAR scans in a map built using the cartographer (we do not have the ground truth). Subsequently, the translational error between the ICP localized trajectory and the pseudo-ground truth from the cartographer is computed. For the first forest, the error is 0.085 ± 0.038 meters, and in the second forest, the error is 0.078 ± 0.035 meters. For the test field, we obtain an average error of 0.13 ± 0.11 between the pseud-ground truth and the ICP.

To better understand the overall accuracy, we compare the mapping and localization performance against ground-truth data in two different environments: a forest and an urban setting (emergency services training ground in Wangen an der Aare, Switzerland). Both areas are shown in Fig. 2.34. The ground-truth data was produced using Leica's RTC 360 3D laser scanner. It is a rotating laser scanner that aligns point clouds and uses VIO to record moves from station to station for scan pre-registration automatically. For both the forest and the training ground, the RTC360 reported sub-centimeter accuracy.

We align the ground truth map and the cartographer map using the Cloudcompare software alignment tool. The aligned maps for the Wangen area are shown in Fig. 2.35. The area has dimensions of 294 x 577 x 66 meters (W x

(a) Wangen and der Aare (CH) **(b)** Forest 3

Figure 2.34: Locations for mapping ground truth evaluation. ***Left:*** Aerial photo of the emergency responders training ground, Wangen and der Aare. Image taken from `https://www.mediathek.admin.ch/media/image/da1c1259-8d9a-430a-893d-d325de139149` ***Right:*** Forest patch where the mapping accuracy was evaluated. The forest in the photo is an old forest.

L x H). The RMS error from the registration was 0.578 m with the average distance between the points of 0.76 ± 0.52 meters (see below for a discussion). To map the area using our proposed setup, about 10 min of walking around the site was required. To obtain the ground truth map, we made 55 scans which required a whole afternoon of work.

Besides the map quality, we also evaluate cartographer's trajectory quality (pseudo-ground truth in Fig. 2.32 and Fig. 2.33). To obtain the ground truth trajectory, we register laser scans in the ground truth map from the Leica scanner using the ICP. Then we align the trajectories estimate from the cartographer with the ground truth trajectory (using the transform obtained during the map alignment step). The trajectory length was 823 m and the overall error was 0.41 ± 0.28 meters, overlay visualized in Fig. 2.38a. This level of accuracy is acceptable for navigation; however, blind grabbing might not be accurate enough, which is why our method does an extra detection step in the local map. We note that the overall errors (both map and trajectory estimation) computed are pessimistic (worst case) because of the changes in the environment between the data collection. We observed new (or differently parked) vehicles, cranes, trailers, tents, and even a large water pool left on the main road of the training ground. All of these changes in the environment negatively influence the accuracy of scan matching and make the correct data association harder.

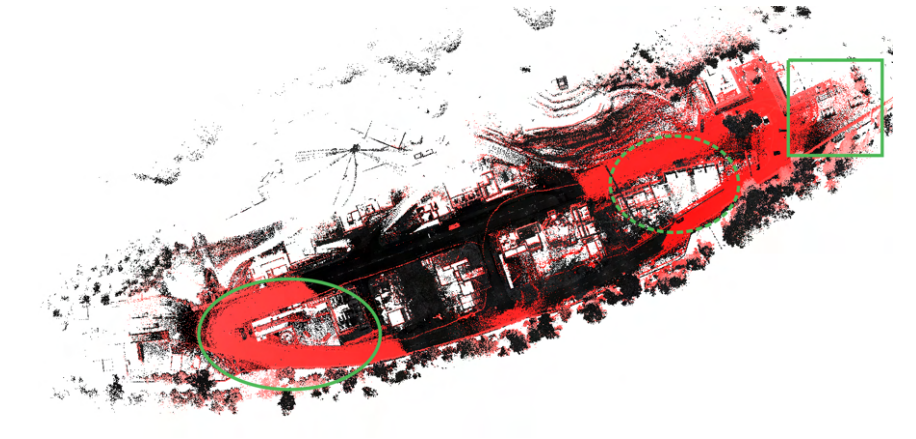
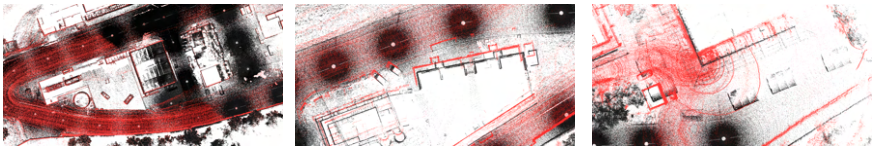


Figure 2.35: Map of the emergency responders training ground, Wangen an der Aare, Switzerland. The ground truth map is shown in black(from Leica RTC360), and in red, the cartographer map is shown. The green circle shows an example of good alignment, the dotted circle shows an example of bad alignment, and the green square shows an example of changes between two mapping sessions. The data was collected a couple of weeks apart.

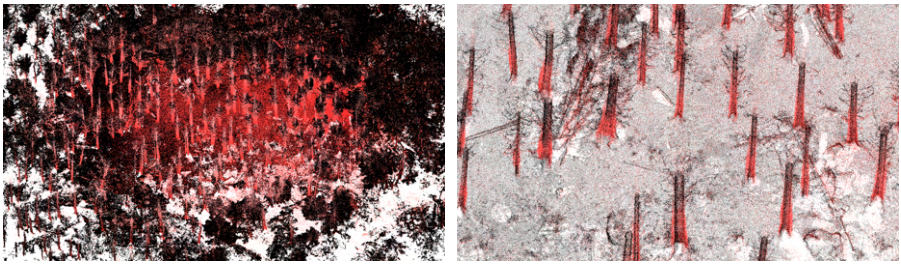
In addition to the urban environment, we collected the ground truth in two forest patches near Wangen an der Aare in Switzerland. Again, the Leica RTC360 3D scanner was used for generating the map. The cartographer map was registered against the ground truth map using Cloudcompare software. For the first forest patch, the RMS registration error was 0.042 m, and the distance between registered point clouds was 0.06 ± 0.05 meters. Dimensions of the forest area are 243 x 257 x 43 meters (W x L x H). The aligned maps are shown in Fig. 2.37. The close-up shot shown in Fig. 2.37b shows a snug fit between the aligned maps. One can also notice how the cartographer map is significantly noisier than the ground truth map from the RTC scanner. This comes to no surprise since RTC360 comes with more than an order of magnitude more accurate distance measurements (less than 5 mm at 40 meters) compared to the Ouster range sensor (5 cm at 40 meters).

ICP was used to register LIDAR scans against the ground truth map and create a ground truth trajectory. We compare it to the pseudo ground truth from the cartographer. Overall translational error was 0.05 ± 0.03 meters and the overlaid trajectories are shown in Fig. 2.38b. This is an order of magnitude lower error compared to the urban environment, which can be explained by the fact that we collected the data in succession, and hence both maps look the same. This allows for better scan registration. Additionally, the forest areas



(a) Good alignment **(b)** Poor alignment **(c)** Changes between data collection

Figure 2.36: Close up view of encircled areas from Fig. 2.35. Ground truth is shown in black and cartographer map shown in red. ***Left:*** Example of good alignment. Note how building edges from different maps snap well onto each other. ***Middle:*** Example of poor alignment. Note how the building walls are far apart from each other. ***Right:*** Changes between the dataset collections. Note the black tents in the ground truth map and the red firefighting trailer in the cartographer map.

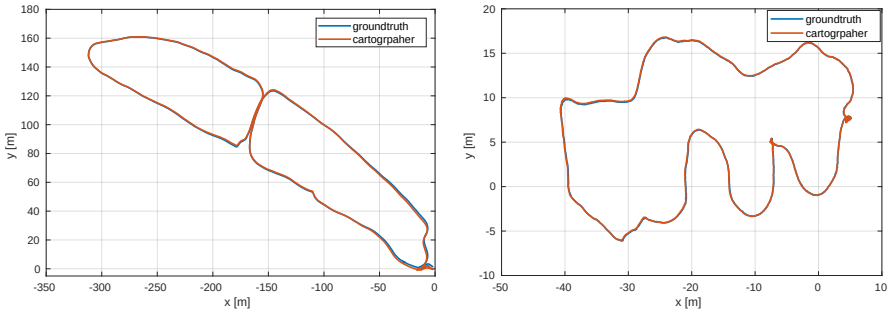


(a) Registered maps **(b)** Close up of the alignment

Figure 2.37: Aligned forest maps. Ground truth map shown in black color where the cartographer map has been shown in red color. ***Left:*** Top view of the aligned maps. ***Right:*** Close up image of the aligned maps. Note how tree trunks nicely fit together.

are smaller, which also leads to better accuracy. We verified the obtained error in another forest patch with younger trees (map size 100 x 96 x 17 meters). Comparing the map with the ground truth map, we obtained similar numbers: the RMS registration error was 0.078 m, the distance between the registered maps was 0.12 ± 0.1 meters. The error between the ground truth trajectory and pseudo-ground truth from the cartographer was 0.04 ± 0.03 meters over the 100.34 m distance traveled (images and a map of the forest patch omitted for brevity).

Both cartographer's localization and the ICP can run in real-time on the robot, however ICP consumes less CPU. The comparisons between cartographer's localization mode and pure ICP based localization suggest the superior performance of ICP. Hence, we would like to conclude that one could also use ICP for localization during harvesting missions. The authors will switch to



(a) Wangen training site

(b) Forest 3 (near Wangen)

Figure 2.38: Top view trajectory estimate from cartographer (brown) aligned with the ground truth trajectory (blue). ***Left:*** Trajectory estimates for data collection at Wangen training site. Note how the error is bigger in the area where maps in Fig. 2.35 are not well aligned. ***Right:*** Trajectory estimates for data set collection in the forest 3.

the ICP based localization in the future. Our ICP implementation is available as open-source for the community[9].

[9] `https://github.com/leggedrobotics/icp_localization`

2.9.5 Fully integrated system

This section presents snapshots of a fully integrated system performing a harvesting mission. We tested our system on a test field where HEAP was commanded to grab "tree trunks". One tree trunk is composed of three wooden logs strapped together such that they can stand straight up (see the video or Fig. 2.39). We show the approaching sequence on our test field in Fig. 2.39 and omit the rest of the maneuver for the sake of brevity since the full maneuver is already shown in Fig. 2.40. In addition to our test field, we have performed the second set of experiments in a small forest alley in a real forest with fully grown adult trees. We show snapshots of 1 operational cycle between grasping two trees in Fig. 2.40. The reader is encouraged to watch a video accompanying this submission since it offers a better insight[10].

Figure 2.39: HEAP approaching wooden logs at our testing field. We do not show the rest of the maneuver for the sake of brevity. The reader is encouraged to watch the video.

It takes about 1-2.5 minutes for the machine to complete the entire cycle, depending on how far the cabin has to turn and how far it has to drive. The cabin turns and driving are tuned conservatively (slow), and the time could be improved drastically by speeding them up. The proposed system was able to run without any changes in both the test field and the forest. Furthermore, we were able to successfully detect both tree trunks and wooden logs and perform a grab. A human operator was often inside the cabin for safety.

[10] `https://youtu.be/1FLD0djPFgU`

Figure 2.40: Snapshots of an operational cycle between grabbing two trees. HEAP repositioning the base close to the tree (snapshots 1-6) followed by scanning the environment and building a local map (snapshots 7-9). Finally, HEAP grabs the tree by extending the arm (snapshots 10-12) and retracting it (snapshots 13-15). After retracting the arm, the harvester is ready to grab the next tree and whole cycle repeats.

2.10 Summary

We present the first (to the best of our knowledge) demonstration of a full-sized harvester performing autonomous precision tree grabbing. We do not investigate tree cutting, which could be done with an appropriate harvesting tool. Components off our system have been evaluated individually and integrated into a complete autonomous system. We show evaluation of the control system, approach-pose planner, mapping accuracy, localization accuracy, point cloud to elevation map conversion algorithm, and the tree detector.

A lightweight and versatile sensor module is presented; it can be used for mapping and later mounted on the harvester for localization. The sensor module and SLAM system together generate an *a priori* map of the mission area. Subsequently, the HEAP harvester can localize itself at mission time, which enables it to navigate under the forest canopy without relying on a GPS signal.

Our system can plan approach-poses in confined spaces, and it can negotiate challenging terrain by combining the chassis-balancing and path-following controllers. Our planner relies on traversability maps that we estimate from elevation maps, which, in turn, we calculate directly from point clouds using our conversion algorithm. We rely on a human expert to specify target trees for harvesting. To combat localization error and enable precision harvesting, we plan grasping poses in the local frame, using a geometric detection algorithm that operates on the laser point-cloud. Lastly, we make parts of our planning[11], mapping[12], localizing[13] and tree-detecting[14] software stack open source for the community.

2.11 Discussion & Outlook

Each of the modules (e.g., planning, mapping) has been first tested and benchmarked individually before integrating them into a complete precision harvesting mission. For example, we first tested planning and control with RTK GPS before adding the SLAM system. Stepwise integration was essential to isolate problems. We also note that having a simulation environment was beneficial, since we could first test the functionality in simulation and then focus on

[11] `https://github.com/leggedrobotics/se2_navigation`
[12] `https://github.com/ANYbotics/grid_map/tree/master/grid_map_pcl`
[13] `https://github.com/leggedrobotics/icp_localization`
[14] `https://github.com/leggedrobotics/tree_detection`

tuning on the machine. State-machine behavior, for example, can be almost completely verified in simulation. Another lesson learned from this work was that state machines get complicated when they must include recovery behavior (e.g. re-plan if no solution is found). In the future, it would be beneficial to look into using behavior trees Winter, Hayes, and Colvin, 2010, which are typically more compact and implicitly integrate recovery behaviors.

Experimental verification of our approach during the wet season was unfortunate, and the biggest problem occurring was that the machine would occasionally get stuck in the mud. This problem was somewhat mitigated by using chains and putting wooden logs and branches in the mud to improve traction. We do not see this as a limitation of our system since forestry operations typically find a place during dry seasons.

We point out that pure, geometric localization (or detection) can fail in a real forest. For example, a crowd of humans observing the machine can look very similar to tree trunks in a point cloud, a situation that, on one occasion, caused the SLAM module to generate spurious pose estimates. Furthermore, we had a few cases of sun rays directly shining on our camera, which condition caused the visual odometry to diverge. Subsequently, the whole state estimation pipeline diverged despite the pure LIDAR odometry working well. This problem seeks an approach that can adequately detect sensor degradation to ensure robust mapping and localizing in all cases. We also believe that it is important to perform a quantitative evaluation of different SLAM systems under similar conditions in forest areas. This procedure would involve testing different approaches (e.g. LOAM variants, Cartographer, ICP) in varied forest styles with a high-accuracy ground-truth map (e.g. from the 3D scanner). The research community could thus focus the development efforts on the most promising approach.

An immediate improvement to the localization system wold be to use ICP as opposed to Cartographer's localization mode (see Section 2.9.4). Other improvements we are planning will focus on utilizing complementary sensors where applicable and implementing health-checking systems to mitigate state-estimating problems. For example, one could run both vision and LIDAR-based SLAM to increase robustness since these two modalities complement each other Khattak et al., 2020. Furthermore, traversability estimation should use visual information, since pure geometry can be misleading in natural environments.

Robust handling of heavy clutter would be especially beneficial, since such situations constitute the primary conditions of tree-detection failure. Detecting

trees in the local map could also benefit from using visual information. In case of thick vegetation or heavy branch clutter, one could also look into using radar sensors for determining a grabbing pose on a tree stem.

We plan to improve the approach-pose planner by incorporating footprint adaptation and adjusting the number of generated approach-poses based on the environment, as noted in Section 2.8. Furthermore, it would be interesting to benchmark different planners (e.g., Probabilistic Roadmap (PRM) or hybrid A* algorithm Dolgov et al., 2010). Additionally, the implementation of approach-pose generation can be parallelized. For tree felling, the arm planner can be enhanced to include tree geometry, once the gripper is holding a tree trunk. Lastly, the tree trunk weight could be included in the arm controller to improve the arm-plan tracking.

Besides improving the mission-execution and motion planners, an exciting area to pursue is to develop a more intelligent mission planner. Instead of just relying on an operator-provided order of execution, one could optimize some objective (e.g., minimal time). Such a planner becomes especially relevant if one wants to maximize efficiency in a scenario involving multiple harvesters.

Apart from algorithmic improvements, one crucial aspect for future forestry missions is to have rugged sensors. We used an umbrella to protect our sensor module from water during the experiments (see Fig. 2.3, in the back of the machine); however, this is not a viable solution for a fully autonomous harvester operating in a harsh environment. Furthermore, the sensors must be protected from branches, vegetation clutter, and branch debris falling onto the machine. Since harvester machines need to deal with clutter and vegetation, it is advisable to have redundant sensors such that localization can be robust to occlusions.

Incorporating the improvements outlined above will bring us one step closer to fully automated precision forestry that will increase yields and relieve humans from tedious labor.

Acknowledgement

This research was partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme grant agreement No 852044 and through the SNSF National Centre of Competence in Digital Fabrication (NCCR dfab). We would also like to thank Alexander Reske for his help with Leica RTC 360 3D laser scanner during the data collection. We also thank to Silvere and Simo Gröhn for their help with Cartographer.

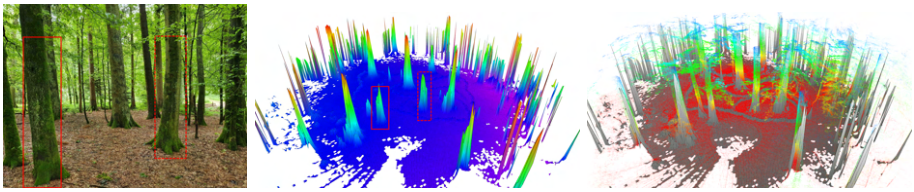
2.12 Appendix

2.12.1 Point Cloud to Elevation Map Conversion

We analyze how the point cloud conversion algorithm presented in Section 2.5.1 reacts to clutter and vegetation. The data was collected with a handheld sensor, and we build small maps of forest patches. The forest patches vary in dimension between 17x18 meters up to 29x32 meters (width x length). We convert the point cloud to the elevation map and show the results below. Unfortunately, we cannot provide rigorous quantitative analysis since the ground truth (actual ground elevation) is extremely hard to collect in the presence of vegetation. Furthermore, it is hard to define the ground truth since there are cases where we want more than just ground information in the elevation map (e.g., trees, stumps). However, looking at the images and maps presented below, one can draw some conclusions about the conversion algorithm's behavior.

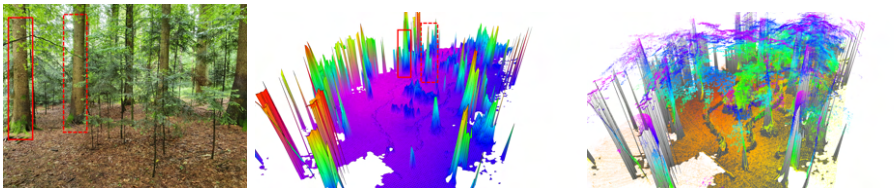
All the elevation maps shown below have a resolution of 10 cm. One can observe (especially from Fig. 2.42) that thin tree trunks present a problem for the elevation map and are often not well captured in it. One could increase the map's resolution to capture them better; however, there is a trade-off between detail and computation time. Furthermore, to avoid holes and artifacts, one needs denser point clouds with the increasing elevation map resolution (smaller cell size). Dense branches and low and medium clutter do not seem to pose a big problem, and the resulting elevation maps have no or low artifacts. Thick tree trunks (30 cm diameter and more) are visible well in the map, and we did not experience any problems with the tree canopy or branches. This holds as long as forest ground is captured in the point cloud. One limitation of the approach is the ability to deal with heavy clutter (e.g., Fig. 2.46). Indeed thick vegetation and branches touching the ground will often appear as blobs in the elevation map.

In Fig. 2.48 we illustrate the influence of a cluster tolerance parameter on the elevation map. Both maps correspond to the cluttered forest patch shown in Fig. 2.46a. One can note that the lower value of cluster tolerance better filters out the clutter and recovers the ground elevation. However, there is a trade-off since it also filters smaller trees out of the map (denoted with a red rectangle). Large tree trunks remain unaffected; in case of a cluttered environment and larger tree trunks, smaller values are recommended. In a case the point cloud is very dense, smaller values usually yield better results.



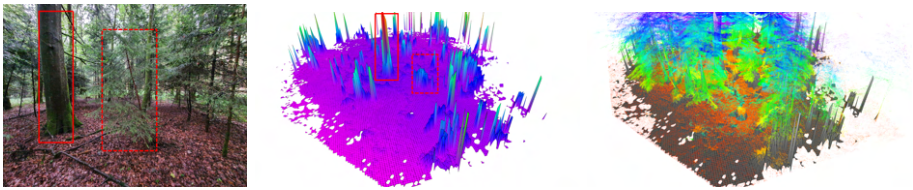
(a) Forest patch **(b)** Elevation map **(c)** Elevation map + point cloud

Figure 2.41: Forest scene with large tree trunks, no clutter and low branch density. The elevation map nicely captures the tree trunks and the forest ground.



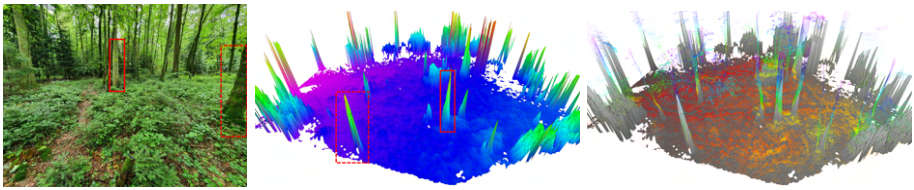
(a) Forest patch **(b)** Elevation map **(c)** Elevation map + point cloud

Figure 2.42: Scene with no clutter, but higher branch density compared to the previous scene. Besides the big tree trunks many smaller trees with trunk diameter less than 10 cm are present. One can clearly see a limitation of our algorithm, since many of the smaller trees are not well captured in the map at 10 cm resolution.



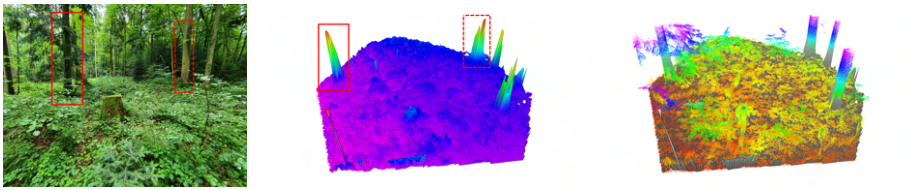
(a) Forest patch **(b)** Elevation map **(c)** Elevation map + point cloud

Figure 2.43: Cluttered scene without vegetation. Compared to the Fig. 2.42, tree trunks in this scene are thicker and better captured in the elevation map. High branch density does not seem to pose the problem for the conversion algorithm as long as there is enough ground data in the point cloud.



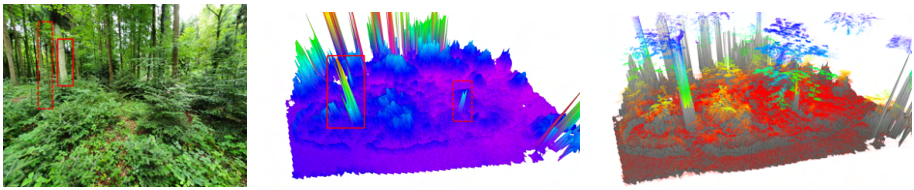
(a) Forest patch **(b)** Elevation map **(c)** Elevation map + point cloud

Figure 2.44: Scene with low vegetation and low clutter. One can see that the elevation map does not contain any artifacts. The vegetation causes the map to be somewhat rougher compared to the forest ground without the vegetation.



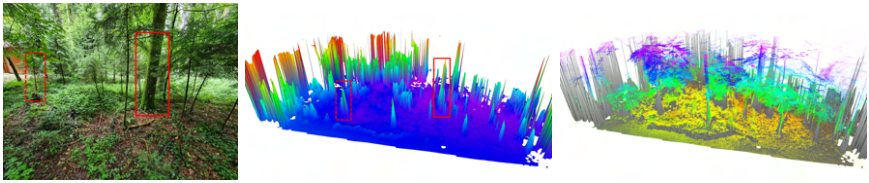
(a) Forest patch **(b)** Elevation map **(c)** Elevation map + point cloud

Figure 2.45: Medium clutter forest patch with ground vegetation. We can see that the surface of the elevation map looks somewhat rougher compared to the Fig. 2.44. The algorithm successfully captures the tree trunks and filters out some thin trees.



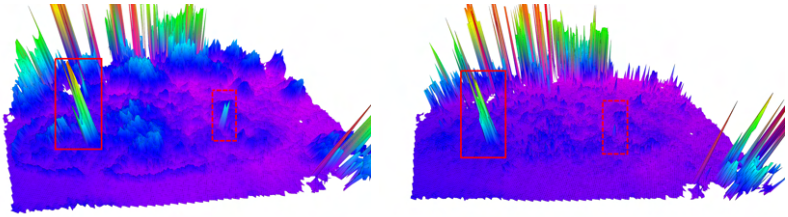
(a) Forest patch **(b)** Elevation map **(c)** Elevation map + point cloud

Figure 2.46: High clutter scene with lots of vegetation. We can see that the thick vegetation corrupts the elevation map and causes bumps and roughness in it. Thick tree trunks are still well visible in the map.



(a) Forest patch **(b)** Elevation map **(c)** Elevation map + point cloud

Figure 2.47: Combined scene with medium vegetation clutter and trees. Branches reaching all the way to the ground effectively inflate the trunk of the tree which can be seen in the middle of the elevation map (big blobs to the left of the tree trunk).



(a) Map computed with higher cluster tolerance **(b)** Map computed with lower cluster tolerance

Figure 2.48: Elevation maps computed using our point cloud to elevation map conversion algorithm with different cluster tolerance parameter. One can observe that the lower cluster tolerance better filters out the vegetation and clutter.

2.12.2 Tree Detection from Local Maps

In this section, we present some tree detection failure cases on the local map patches. Example failure cases are shown in Fig. 2.49. The most common cause of failure is a false negative when no tree is detected, and there is one in the map. False-negative examples can be seen in snapshot 1,2,3,4,5,7,8,9,10,11 and 12. What all these fail cases have in common is the presence of vegetation and clutter in the scene. Some patches are so cluttered (e.g., 1 and 12) that it is extremely challenging for a human to find the trees in the scene. Since our approach extracts Euclidean clusters, any branches or canopy that touches other trees presents a problem. Touching canopies (from two different trees) cause the algorithm to extract two or more trees (together with all the branches) as one cluster. Subsequently, the whole blob of points will be rejected because it is either too wide or the gravity alignment is not vertical enough. Ultimately,

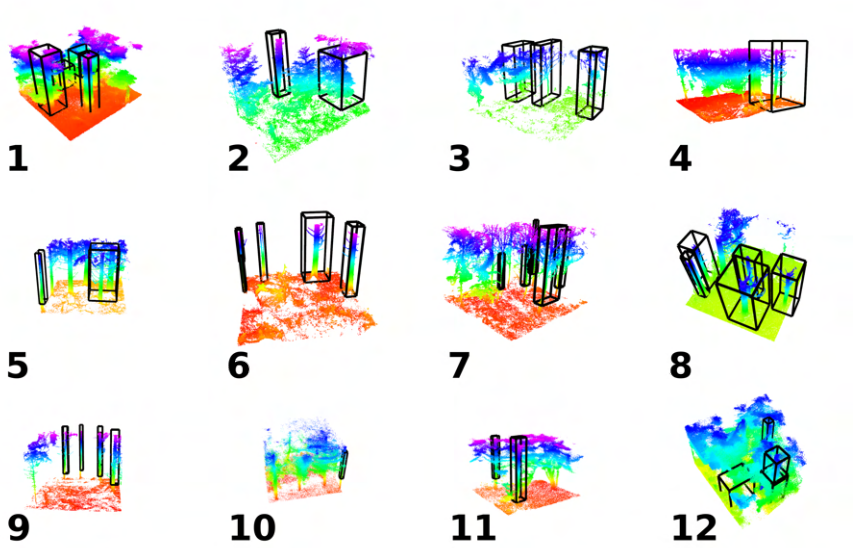


Figure 2.49: 6 m by 6 m point cloud patches where detection algorithm failed. Red corresponds to the lowest and purple to the highest elevation. Tree detections are shown with transparent boxes. The patches come from three different maps that were used for detection accuracy evaluation (see Sec. 2.9.3.

since our detection approach is purely geometric, it runs into limitations in the presence of high clutter and vegetation. The clutter problem could be mitigated by changing the cluster tolerance in the Euclidean cluster extraction

phase. If the point cloud is very dense smaller values (less than 0.1 m) usually yield better results. Values that are too small will result in a single tree being split into multiple clusters, whereas large values will result in multiple trees merged into a single cluster. This calls for a detection procedure involving a tree model, which would help filter out the branch clutter (e.g., RANSAC). A learning-based approach could also be an option. Alternatively, one could use a different sensor modality that will penetrate through the clutter and allow for direct tree trunk detection (e.g., a radar).

The second failure mode is mistaking a branch for a tree or detecting the same tree twice (false positive). Examples of such behavior can be seen in snapshots 6 and 12. This type of failure is not as common as the false negative. One could extend our approach with simple sanity checks (e.g., if two clusters have similar x and y coordinates than most likely they belong to the same tree) to increase the performance. Another option would be to filter out branches and vegetation, which can be mistaken for a tree. A visual sensor would also help in this case since it is relatively easy to distinguish between vegetation and a tree trunk for a human.

2.13 Lessons Learned

We propose a system for autonomous precision tree-grabbing in a natural forest, a milestone towards autonomous precision tree harvesting. The mission starts with a human mapping an area of interest with a handheld sensor module. Subsequently, the module is mounted on HEAP to enable navigation under the forest canopy without GPS signal. Strategies for mapping, localization, control, and planning are proposed. We focus on lessons most relevant for this thesis, i.e., lessons learned in planning and control.

We observed that having terrain-adaptive behavior was essential for completing the mission. The chassis balancing system overcame substantial obstacles, such as ditches and stumps that the motion planner was unaware of. Furthermore, minimizing contact forces on the legs achieved traction control, preventing wheels from falling deep into the mud. Subsequent controllers deployed on HEAP were all terrain-adaptive.

Even the terrain adaptive controller was sometimes not enough, and the machine would get stuck in the mud. In this case, the arm was manually deployed to lift the wheels out of the mud and push itself. This motivated us to investigate whether we could develop a planner using the arm for locomotion.

Although the recovery behavior for the forestry use case could have been realized using heuristics, we wanted to create a general planner that can mimic what good human operators can do (e.g., in MenziMuckCom, 2011).

Lastly, we tried to compute approach poses for tree grabbing heuristically, which was susceptible to many corner cases. However, coming up with a set of candidate poses was relatively easy. Subsequently, the approach pose selection was outsourced to RRT planner, and we optimized for path and approach pose in a single motion planning problem. We followed this approach of giving as much freedom as possible to the planner in the rest of the thesis.

3

Whole-body motion planning for walking excavators

Jelavic, E. and Hutter, M. (2019). "Whole-body motion planning for walking excavators". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2292–2299

DOI: 10.1109/IROS40897.2019.8967631

Video: https://youtu.be/u3XO4PaPc6I

This article presents a trajectory planning framework for all-terrain vehicles with legs and wheels such as walking excavators. Our formulation takes into account the whole body of the robot while computing the plans for locomotion. Hence, we can produce motion plans over the rough terrain that would be hard to plan without considering all Degrees of Freedom (DoF) simultaneously. Our planner can also optimize over the contact schedule for all limbs, thereby finding the feasible motions even for the infeasible initial contact schedule. Furthermore, we introduce a novel formulation of the support area constraint. We generate plans for a Menzi Muck M545, a 31 DoF walking excavator with five limbs: four wheeled legs and an arm. We show motion plans for traversing a variety of terrains that require whole-body planning. To the best of our knowledge, this is the first work that addresses motion planning in rough terrain for vehicles with legs and wheels.

3.1 Introduction

Robots with legs and wheels such as walking excavators offer great potential for operation in uneven terrain. They can be deployed for various tasks such as

landscaping, forestry or earth-moving. This versatility comes with a cost: to perform locomotion in rough terrain the operator is required to coordinate all degrees of freedom for each of the limbs simultaneously. As a result, operating such a system can be quite exhausting and hiring experienced operators is expensive. Furthermore, handcrafting the motion generation for rough terrain traversal is difficult, because legged-wheeled systems have many DoF. To this end, we present a Trajectory Optimization (TO) based planning approach that leverages all DoF of a legged-wheeled robot. We do this by planning over the full joint space and the base pose at the same time which allows us to find a set of motions that is richer than if we planned motion for parts of the robot separately. Using TO for planning is attractive because it is general enough so that any high-level task can be formulated as a TO problem. With enough computing power (an ideal case), the solver produces a valid plan or reports that a solution does not exist which would solve the planning problem on a general level. We demonstrate our approach on a walking excavator. Planning the motion for the whole body, allows us to compute plans for different terrains using the same planner.

3.1.1 Related Work

Path planning techniques for wheeled robots and legged robots with point feet have received a lot of attention. However, there is a significant research gap in the field of planning for legged-wheeled robots and the literature available is rather scarce. In the following, we categorize the existing approaches in terms of how they solve the planning problem (e.g., static or dynamic motion plans, contact schedule optimization). We also discuss the missing components to address the motion planning problem in rough terrain for a legged-wheeled robot.

1) Dynamic trajectory planning Trajectory optimization for legged robots has been well studied, and so far impressive results have been shown for legged robots with point feet Bellicoso et al., 2018b; Farshidian et al., 2017a; Winkler et al., 2018, 2017. Typically, planners for legged locomotion produce dynamic motion plans that are computed as a solution to a complex nonlinear program. Although the application of those planners is limited to the robots with point feet, there is still some connection to our problem. In both cases, one needs to enforce a similar set of constraints, ensure that the robot remains stable or even optimize over the contact schedule (duration and order of contact phases for each End-Effector (EE)). Recently, there have also been advances in the locomotion planning for legged-wheeled robots such as Viragh

et al., 2019, Bjelonic et al., 2019a and Geilinger et al., 2018. All of these authors show dynamic locomotion with wheels, although without taking the terrain into account while planning. Furthermore, the optimization horizon is rather short (usually 1-2 seconds) and may not be suitable for slow robots. While the approaches used for dynamic legged-wheeled locomotion planning are closer related to our problem, there is still a major difference. The difference stems from the fact that our platform is a customized, 12000 kg heavy, walking excavator *HEAP (Hydraulic Excavator for an Autonomous Purpose)* 2018. Therefore we restrict ourselves to computing the kinematically feasible motions only. Kinematic planning is justified by the fact that our system is rather slow, Coriolis forces can be neglected and planning dynamic motions becomes unnecessary. Moreover, some dynamic motions could even be dangerous. We require that all states are statically stable. As a benefit, omitting the complex rigid-body dynamics in the problem formulation allows us to extend the optimization horizon. This is because we do not have to enforce dynamics constraint along the trajectory which can be expensive to compute.

2) Statically stable trajectory planning Wheeled-legged systems were successfully deployed for extraterrestrial missions in rough terrain where the system is required to be statically stable at all times. However, little work has been done on computing motion plans that would keep a robot statically stable. Typically, the robot is teleoperated and a purely reactive controller is trying to keep the base in the desired pose; examples can be found in Cordes, Babu, and Kirchner, 2017; Giordano et al., 2009; Hashimoto et al., 2005; Reid et al., 2016; Smith, Sharf, and Trentini, 2006; Wilcox, 2012. Computing motion plans for legged-wheeled robots in an indoor environment has been done in Hashimoto et al., 2005, Lim et al., 2017 and Klamt and Behnke, 2017, 2018; Klamt et al., 2018. None of these approaches considers the whole body planning problem. Driving and stepping phase are treated as two separate behaviors and the robot switches between them. Typically, heuristics are used to deduce when to switch and which action to do next. In Klamt et al., 2018, the switching command comes from the human operator. The switching behavior is in contrast with our formulation, where we formulate a whole body motion planning problem. Driving and stepping motions are considered together and the optimizer may choose to perform both at the same time. One interesting approach to planning for legged-wheeled locomotion can be found in Tanaka and Hirose, 2008, where wheels are used for locomotion and legs to jump over the obstacles. This approach, although interesting from a system design point of view, is not suitable for a heavy excavating machine. Kinematically feasible plans have been computed in Giftthaler et al., 2017 for the In Situ Fabricator

2 (IF2), a robot with a similar level of complexity as our excavator. Although their planner considers the planning problem for the whole body, they do not consider any stepping motions. Furthermore, they do not consider the terrain in the planning algorithm.

3) Stability Criterion Formulation The support polygon stability criterion has been known in the legged robotics community for a long time Wieber, 2002. Usually, it has been embedded into the TO as an intersection of half-spaces. For the sake of robustness, the support area is often shrunk which leads to a tedious computation of half-spaces and error-prone geometric manipulations. Recent work Del Prete, Tonneau, and Mansard, 2016; Winkler et al., 2017 has proposed to formulate the support area stability criterion as a convex hull of the support points. To drive Center of Pressure (CoP) away from the edge of the support polygon, they add a cost term in the optimization. However, because a cost term is used to enforce the stability, CoP can still reach the edge of the support polygon (and the robot could fall). To this end, we bring a novel formulation of the support polygon constraint that can shrink the support area without computing any half-spaces. This way we retain a guarantee that CoP (or Center of Mass (CoM) in our case) never reaches an edge of the support polygon.

4) Contact Schedule Optimization To compute the motion plans for legged systems, one needs to reason about which EE's are in contact with the environment at all times. The discrete nature of contacts makes the planning a hard combinatorial problem. For optimizing the contact schedule, previous approaches have used Mixed-Integer optimization formulations which become hard to compute for longer planning horizons Deits and Tedrake, 2014 or complementary constraints that require special treatment from the solver Mordatch, Todorov, and Popović, 2012. Most recently, contact schedule has been computed by incorporating timings as continuous variables in the Nonlinear Program (NLP) Winkler et al., 2018. The latter approach can use a general NLP solver, and we extend it for legged-wheeled robots.

3.1.2 Contribution

To the best of our knowledge this is the first work that addresses a whole body motion planning for legged-wheeled systems in rough terrain. Our approach is targeted for use on the autonomous walking excavators however, it could be used on other robots as well. With respect to the related work, our contributions can be summarized as follows: We introduce the complete TO formulation for legged-wheeled robots in rough terrain. Furthermore, our

formulation allows for the use of different types of EE's for locomotion. A walking excavator is a good example since it can use both wheeled legs and the shovel for locomotion. In addition, we introduce a novel and intuitive stability constraint formulation. We allow for shrinkage of the support area without tedious computation of half-spaces. Finally, we present a formulation that can optimize the contact schedule for legged-wheeled robots. Our formulation uses continuous variables and can be embedded directly into an NLP.

3.2 Problem Formulation

We solve a continuous time TO problem that can be described as

$$\begin{aligned} &\text{find} && \boldsymbol{x}(t), \dot{\boldsymbol{x}}(t) \\ &\text{s.t.} && \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \qquad \boldsymbol{x}(t_F) = \boldsymbol{x}_F \\ & && \boldsymbol{g}(\boldsymbol{x}) \leq 0 \quad \boldsymbol{g}_h(\boldsymbol{x}) = 0 \quad \boldsymbol{g}_{nh}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = 0 \end{aligned}$$

Where $\boldsymbol{g}(\boldsymbol{x})$ set of inequality constraints, $\boldsymbol{g}_h$ and $\mathbf{g}_{nh}$ represent the set of holonomic and non-holonomic constraints, respectively, the vector $\mathbf{x}(t)$ is a vector of decision variables, and t_0 and t_F are the initial and final time, respectively. In essence, what we need to solve is an Optimal Control Problem (OCP). There are different techniques to solve an OCP such as Dynamic Programming, Indirect Methods and Direct Methods that transcribe the continuous time problem into an NLP Diehl et al., 2006. We choose to transcribe an optimal control problem into an NLP. Solving an optimal control problem as an NLP is attractive because one does not need to provide a stable (or feasible) initial solution to the problem; this is all outsourced to the optimizer. For us, this means that the challenge of computing an initial feasible trajectory over long time horizons and complex terrains is eliminated. Besides, Nonlinear Optimization is a well-developed field with the mathematical apparatus capable of dealing with very general equality and non-equality constraints. The full NLP that we solve to generate motion plans is summarized in Eq. 3.2a below. Constraint specific to our excavator, (Eq. 3.2b) is shown in green color.

$$
\begin{aligned}
\text{find:}\quad & {}^{I}\boldsymbol{p}_B(t) && \textit{base linear position} && \text{(3.2a)}\\
& \boldsymbol{\theta}_B(t) && \textit{base euler angles}\\
& \forall\, i, \quad i \in \mathcal{L}: \\
& \quad {}^{I}\boldsymbol{p}_i(t) && i^{th}\ \textit{limb position}\\
& \quad \boldsymbol{q}_i(t) && i^{th}\ \textit{limb joint angles}\\
& \quad T_{i,j} && i^{th}\ \textit{limb gait timings}\\
\text{s.t.}\quad & {}^{I}\boldsymbol{p}_B(t=0) = \boldsymbol{p}_0 && \textit{initial position}\\
& {}^{I}\boldsymbol{\theta}_B(t=0) = \boldsymbol{\theta}_0 && \textit{initial orientation}\\
& {}^{I}\boldsymbol{p}_B(t=T) = \boldsymbol{p}_g && \textit{goal position}\\
& {}^{I}\boldsymbol{\theta}_B(t=T) = \boldsymbol{\theta}_g && \textit{goal orientation}\\
& {}^{B}\boldsymbol{p}_{COM}(t) \in \mathcal{S}(t) && \textit{support polygon}\\
& \forall\, i,\ \ i \in \mathcal{L}: \\
& \quad {}^{I}\boldsymbol{p}_i(t) \in \mathcal{W}_i(t) && i^{th}\ \textit{limb workspace}\\
& \quad \boldsymbol{b}_L \le \boldsymbol{q}_i(t) \le \boldsymbol{b}_U && i^{th}\ \textit{limb joint bounds}\\
& \quad \sum_j T_{i,j} = T && i^{th}\ \textit{limb total duration}\\
& \quad \text{if } i \in \mathcal{C}(t): \\
& \quad\quad {}^{I}\boldsymbol{p}_i^{z} = h_{terrain}({}^{I}\boldsymbol{p}_i^{x,y}) && \textit{terrain height}\\
& \quad\quad \text{if } i \in (\mathcal{C}(t) \cap \mathcal{L}_w): \\
& \quad\quad\quad {}^{W_i}\dot{\boldsymbol{p}}_i \cdot \boldsymbol{a}^{y} = 0 && \textit{no lateral slip}\\
& \quad\quad \text{if } i \in (\mathcal{C}(t) \cap (\mathcal{L} \setminus \mathcal{L}_w)): \\
& \quad\quad\quad {}^{I}\dot{\boldsymbol{p}} = 0 && \textit{zero contact velocity}\\
& \quad\quad\quad {}^{I}\boldsymbol{a}^{x} \times {}^{I}\boldsymbol{n}({}^{I}\boldsymbol{p}_i^{x,y}) = \boldsymbol{0} && \textit{shovel orientation} && \text{(3.2b)}
\end{aligned}
$$

The problem formulation is general enough for legged-wheeled systems except for the shovel orientation constraint that is specifically tailored to our excavator called Hydraulic Excavator for Autonomous Purpose (HEAP) Jud et al., 2021b. $\mathcal{L}(t)$ is the set of all limbs (HEAP has five limbs, four legs + an arm). The set $\mathcal{C}(t)$ denotes all the limbs that are in contact at time t, $\mathcal{S}(t)$ is the support polygon at time t, and the set $\mathcal{L}_w$ denotes all the limbs that have a wheel at the end. Left superscript is a coordinate frame; I denotes the iner-

tial (world) frame, B denotes the base frame, and W_i denotes the i^{th} wheel frame. Right superscript denotes a component of the vector e.g. ${}^{W_i}\dot{\boldsymbol{p}}_i^z$ is the z component of the velocity of the i^{th} limb expressed in the i^{th} wheel frame. Coordinate frames used for problem formulation are shown in Fig. 3.3a, 3.3b and 3.3c.

3.2.1 Joint and Cartesian Space Formulation

Decision variables for the NLP shown in Eq. 3.2a are the base pose, joint positions and EE positions. Note that the EE variables are redundant since we already know the EE position through the forward kinematics. We chose to keep them since they lead to a more natural formulation of the terrain-related constraints. Furthermore, the addition of EE positions as optimization variables facilitates the NLP initialization; it is easier to find points that satisfy the terrain constraints in Euclidean space than to compute them directly in the joint space.

Using box constraints to enforce the kinematically feasible range of motion for EE's can be found in Winkler et al., 2018, Bellicoso et al., 2018b). This is desirable since it leads to linear constraints and speeds up the optimization. However, box constraints are not suitable for our platform. The workspace of each wheeled EE is a non-convex shaped volume that is difficult to describe analytically in Euclidean space (as shown in Fig. 3.1). Hence, specifying the range of motion constraint becomes laborious, error-prone and tedious. To ensure that our plans are kinematically feasible, we add the joint variables into the optimization and enforce the forward kinematics constraint. In Fig. 3.1, one can see that HEAP's legs do not have knee joints, so the position of the EE always stays equally far away from the hip. The absence of knee joints restricts the EE range of motion and makes it hard to come up with good heuristics for step planning for individual limbs (approach presented in Klamt and Behnke, 2018, Klamt and Behnke, 2017). We would also like to point out that by considering the system as a whole, we can discover a richer set of motions and ultimately traverse more challenging obstacles.

3.2.2 Spline Formulation

NLP decision variables are represented in the time domain as splines composed of third order polynomials. Instead of directly optimizing over the polynomial coefficients, we optimize over the states at polynomial junctions and the time duration of the polynomial. This is the Hermite Parametrization (HP); if two

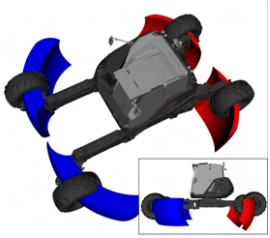


Figure 3.1: Model of the excavator shown without the arm for better visibility. Workspace of the EE's is shown; in blue color for the front legs, in red color for the hind legs. These shapes depict the volume where HEAP can place the center of the wheels without moving the base. Workspace of the arm is essentially a cylinder that encompasses the whole base with an inner radius of 1.5 m and with the maximal outer radius of 8 m.

nodes $(x_k, \dot{x}_k)$, $(x_{k+1}, \dot{x}_{k+1})$ are connected by a third order polynomial with a duration ΔT then:

$$x(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0$$
$$a_i = f(x_k, \dot{x}_k, x_{k+1}, \dot{x}_{k+1}, \Delta T) \quad \forall\, t \in [t_k, t_{k+1}]$$

HP leads to a more natural constraint formulation since we do not have to transcribe states into the polynomial coefficients and we can directly optimize over the states and velocities; which is why we chose HP for collocation. Derivation of the relationship between the polynomial coefficients and node values at junctions can be found in Winkler et al., 2018.

3.2.3 Contact Schedule Optimization for Wheeled robots

To optimize over the contact schedule, we extend the idea from Winkler et al., 2018 such that it can handle limbs with wheels. Same as for a pointed foot, a limb with wheel can either be in contact or not, which means that contact and swing phase alternate. The main difference w.r.t to phase-based parametrization introduced in Winkler et al., 2018 is that one cannot fix the value of a certain spline in one of the phases. E.g., For a robot with point feet the position of a foot does not change while the foot is in contact; however, a position of a wheeled EE can change in both contact and swing phase. The difficulty with wheeled EE's is that different constraints have to be active for contact and swing phase. In phase based parametrization, only the timings of the polynomials change, the number of junctions between polynomials per phase stays constant. Hence, one could try to enforce the phase-specific constraints directly at the polynomial junctions, because the junctions always remain in

the same phase. The latter approach, however, doesn't work for constraints such as support area constraint which depend on the contact state of more than one limb. This is because the polynomial junctions can be shifted in time and hence, the contact state of the other limbs might change. To make the contact schedule optimization for wheels possible, we propose to add the indicator variables $c_i(t), \forall\, i \in \mathcal{L}$ (similar to Neunert, Farshidian, and Buchli, 2016) which can take values between 0 and 1. These indicator variables are splines shown in Fig. 3.2. They are parametrized using the phase based parametrization. The splines are forced to be constant, except at the transition phase (darker red regions in Fig. 3.2) which should be kept as short as possible. Using the contact indicators, we can enforce alternating sets of constraints in alternating phases. When the constraint should not be active, the corresponding $c_i(t)$ becomes 0, and product of the constraint and the indicator evaluates to zero. This way, we can still optimize the contact schedule for limbs with wheels.

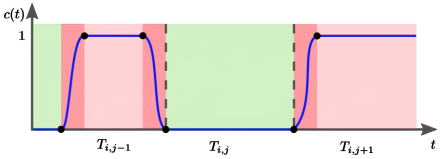


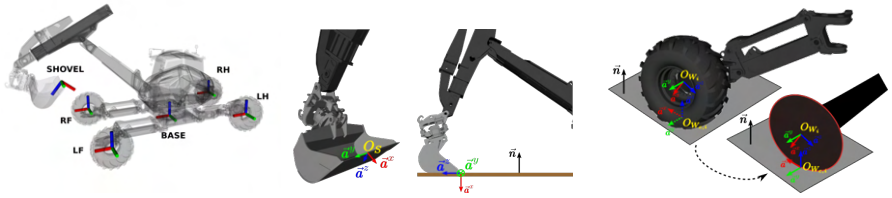
Figure 3.2: Contact indicator $c(t)$ variables used for optimizing over the gait. Each contact indicator variables takes the value 1 (red area) if the foot is in contact and the value 0 if it is not (green area). Polynomials in splines are kept constant except at the transitions between the swing and stance phase (darker red area). Splines are connected at the nodes (black dots); note that the number of nodes per phase is always constant and not dependent on the duration of the polynomials in between.

3.3 Implementation

We describe some constraints shown in Eq. 3.2a in more detail. We do not use any cost in our formulation since this allows for faster computation times and less tuning. However, one might want to add the cost to gain more control over the planned motion.

3.3.1 Limb Workspace Constraint

This constraint ensures that the position of the EE's stays in the allowed range of motion with respect to the base. We formulate this constraint for each limb,



(a) Cordinate frames on HEAP **(b)** Shovel coordinate system. **(c)** Wheel coordinate systems.

Figure 3.3: ***Left:*** Coordinate frames used for planning with HEAP. Each limb has a coordinate frame attached to the EE. Legs of the excavator are denoted with abbreviations, LF stands for Left Front, RH for Right Hind etc. x axis is shown in red color, y in green and z in blue. ***Middle:*** O_S denotes the origin of shovel contact coordinate system. The x, y, z axis of coordinate systems are denoted with $\boldsymbol{a}^x, \boldsymbol{a}^y, \boldsymbol{a}^z$ respectively. The shovel orientation constraint enforces that the $\boldsymbol{a}^x$ vector is parallel with the normal vector $\boldsymbol{n}$. ***Right:*** O_{W_i} is in the center of the wheel, $\boldsymbol{a}^y$ is axis of rotation. $O_{W_{c,i}}$ is the contact point of the i^{th} wheel. For modeling purposes, we use a thin disc wheel model.

regardless of whether it has a wheel or not. The constraint is formulated in the world frame as follows:

$$\boldsymbol{R}_{BI}(\boldsymbol{\theta}(t))({}^{I}\boldsymbol{p}_i - {}^{I}\boldsymbol{p}_B) = {}^{B}\boldsymbol{T}(\boldsymbol{q}_i) \ \in \mathbb{R}^3 \quad \forall i \in \mathcal{L}$$
$$\boldsymbol{b}_{L,i} \leq \boldsymbol{q}_i \leq \boldsymbol{b}_{U,i} \ \in \mathbb{R}^{DoF(i)} \quad \forall i \in \mathcal{L}$$

Where ${}^{I}\boldsymbol{p}_i, \forall i \in \mathcal{L}$ denotes the position of the EE in the world frame, ${}^{B}\boldsymbol{T}(\boldsymbol{q}_i)$ is the forward kinematics computed for the i^{th} limb in the base frame, $\boldsymbol{R}_{BI}(\boldsymbol{\theta}(t))$ is the rotation matrix from the inertial frame into the base frame, ${}^{I}\boldsymbol{p}_B$ and ${}^{I}\boldsymbol{p}_i$ are positions of the base and i^{th} EE in the world frame, respectively. $\boldsymbol{b}_{L,i}$ and $\boldsymbol{b}_{U,i}$ are the lower and the upper bound for the joint positions $\boldsymbol{q}_i$ of the i^{th} limb and $DoF(i)$ is the number of joints that the i^{th} limb has.

3.3.2 Robust Support Polygon Constraint

Support polygon constraint enforces stability along the trajectory. In general, for contact points that are not on the same height, the admissible region for the CoM such that the robot is in the static equilibrium is not equivalent to the support polygon spanned by those contact points (projected in the plane perpendicular to the gravity vector). However, in Del Prete, Tonneau, and Mansard, 2016 it has been shown that as long as each contact point's friction cone contains a direction opposite to the gravity, CoM being in the support polygon spanned by the contact points is sufficient for the static stability.

Hence it follows that one of the limitations of our approach is that the contact points should not be on the surfaces that are "too steep." We use a convex hull formulation for the support polygon constraint, thus avoiding the need to compute any half-spaces. Computing half-spaces is tedious since HEAP can have up to five contact points at the same time. Expressing the constraint as a convex hull enables us to use the same constraint formulation even though the number of the EE's that are in contact changes over time. We enforce the following constraint at discrete time intervals:

$$\sum_{i \in \mathcal{C}(t)} \left(\alpha_i + \frac{\epsilon}{n}\right) {}^{B}\boldsymbol{T}(\boldsymbol{q}_i) = {}^{B}\boldsymbol{p}_{COM} \ \in \mathbb{R}^2$$

$$\sum_{i \in \mathcal{C}(t)} \alpha_i = 1 - \epsilon \ \in \mathbb{R},\ 0 \leq \epsilon \leq 1,\ \alpha_i \geq 0$$

Where α_i and ϵ are real numbers, ${}^{B}\boldsymbol{p}_{COM}$ is the center of mass (CoM) expressed in the base frame, and n is the number of EE's that are in contact at time t. Note that α_i are optimization variables while ϵ is a parameter. The support polygon constraint keeps the robot from falling. However, the optimizer might find motion plans where CoM of the robot is on the boundary of the support polygon. To avoid this situation, we shrink the support polygon. Thanks to our formulation of the stability constraint, this is done very elegantly without the need to compute any half-spaces. The bigger the ϵ, the more conservative we are; i.e., the minimum permitted distance of CoM to the edge of the support polygon becomes larger. The reader is referred to the Appendix for the derivation of the constraint.

3.3.3 Shovel Orientation Constraint

To prevent damage to the terrain, we impose that shovel bucket has to touch the ground with the flat end. The constraint is implemented as follows:

$$\boldsymbol{R}_{SB}(\boldsymbol{q}_i)\boldsymbol{R}_{BI}(\boldsymbol{\theta}(t))\, {}^{I}\boldsymbol{n}({}^{I}\boldsymbol{p}_i^{x,y}) \times {}^{S}\boldsymbol{a}^{x} = \boldsymbol{0} \ \in \mathbb{R}^3$$
$$\forall i \in (\mathcal{C}(t) \cap (\mathcal{L} \setminus \mathcal{L}_w))$$

$\boldsymbol{R}_{SB}(\boldsymbol{q}_i)$ is the rotation matrix from the base frame into the shovel frame (frames shown in Fig. 3.3), $\boldsymbol{R}_{BI}(\boldsymbol{\theta}(t))$ is rotation from the inertial frame into the base frame and ${}^{I}\boldsymbol{n}({}^{I}\boldsymbol{p}_i^{x,y})$ is the terrain normal in the inertial frame at the point where shovel (see Fig. 3.3b) touches the ground. ${}^{S}\boldsymbol{a}^{x}$ is x unit vector expressed in the shovel frame and $\times$ is cross product.

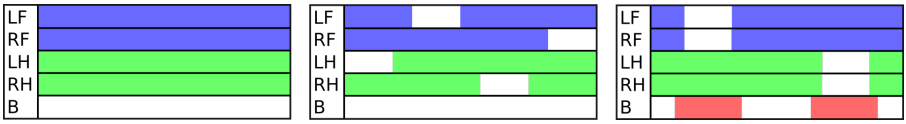


Figure 3.4: Gaits used on HEAP. Color denotes limb that is in contact, and white space denotes that the limb is in the swing phase. All timings are normalized. ***Left:*** Driving gait. ***Middle:*** Walking without the arm. ***Right:*** Walking with legs and the arm.

3.3.4 Lateral Slip Constraint

For every limb that has a wheel at the end and is in contact, we forbid moving in the lateral direction (no moving sideways). To avoid the lateral slip, the y component of the linear velocity of the center of the wheel expressed in the frame W_i has to be zero (see Fig. 3.3c). Strictly speaking, one should constrain the lateral velocity of the contact point to be zero, and that velocity also depends on the angular velocity of the center of the wheel and the radius of the wheel. For our case, the angular velocity is so small that it can be neglected. The constraint without simplifications can be found in Giftthaler et al., 2017.

$$\boldsymbol{R}_{W_iB}(\boldsymbol{q}_i)\boldsymbol{R}_{BI}(\boldsymbol{\theta}(t))\,{}^{I}\dot{\boldsymbol{p}}_i(t)\cdot{}^{W_i}\boldsymbol{a}^{y}=0\ \in\mathbb{R}$$
$$\forall i\in(\mathcal{C}(t)\cap\mathcal{L}_w)$$

$\boldsymbol{R}_{W_iB}(\boldsymbol{q}_i)$ is the rotation from base frame to the i^{th} wheel frame., ${}^{I}\dot{\boldsymbol{p}}_i(t)$ is velocity of the center of the wheel in the world frame and ${}^{W_i}\boldsymbol{a}^{y}$ is the y axis of the coordinate system in the center of the wheel (see Fig. 3.3c).

3.3.5 Gait Design

Although our formulation allows for optimization of the contact schedule and has the ability to change the gait pattern, we need to provide an initial contact schedule to the optimizer. We have designed three different gait patterns, purely by observing the sequences that human operators perform. Gaits that we use are shown in Fig. 3.4. The driving gait with all wheels in contact and the arm in the air is used most often (left part of Fig. 3.4). Next, we design a walking gait with an overlap (all four wheels on the ground) and the arm always in the air (middle). This gait is useful if one needs to change the robot's pose in place (i.e., without driving). On the right, the stepping gait is shown. The arm establishes ground contact to help to overcome obstacles such as steps or gaps.

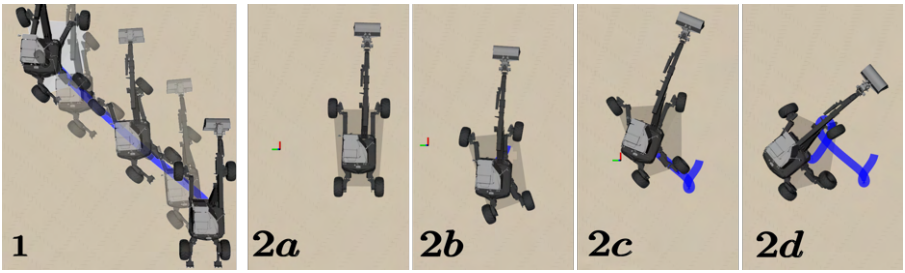


Figure 3.5: ***Subfig. 1*** shows HEAP repositioning itself diagonally from $(0, 0)$ m to $(10, 10)$ m. ***Subfig. 2a-2d*** shows HEAP performing cusps to arrive at a goal pose 5 m to the left. We do not impose any constraints on the final EE configuration. Blue line shows the CoM trajectory.

Figure 3.6: ***Left to Right***: HEAP crossing a 3m wide gap using the stepping gait. The coordinate frame marks the goal position. The arm is facing forward initially, but it has to be turned back to step over the gap successfully. This behavior can be seen in the middle figure; HEAP is turning the arm 180 degrees to establish contact on the left side of the gap and thus be able to step to the right side.

3.4 Results and Discussion

In this section we present different motions generated for HEAP using our planner. We implemented our trajectory optimization problem in C++, using the Ipopt as an NLP solver Wächter and Biegler, 2006 and Hsl subroutines as linear solvers Science and Council, 2007. We interface the Ipopt using Ifopt Winkler, 2018a. All derivatives are computed analytically for better performance. Computing the forward kinematics and the corresponding jacobians was done using the RBDL library Felis, 2016. We show different behaviours that emerge from our TO formulation. We don't change the problem formulation to obtain different motions, we merely provide a different initial contact schedule. Our problem setup only allows for changing the timings of the contact schedule, it does not allow for adding or removing contact phases.

3.4.1 Driving Motions

We show some motions that emerge from our planner when suing driving gait introduced in section 3.3. The motion plans are shown in Fig. 3.5. In Sub-

Figure 3.7: ***Left to Right***: HEAP climbing on a 1m block using the stepping gait. The machine exhibits the same behavior as for the gap.

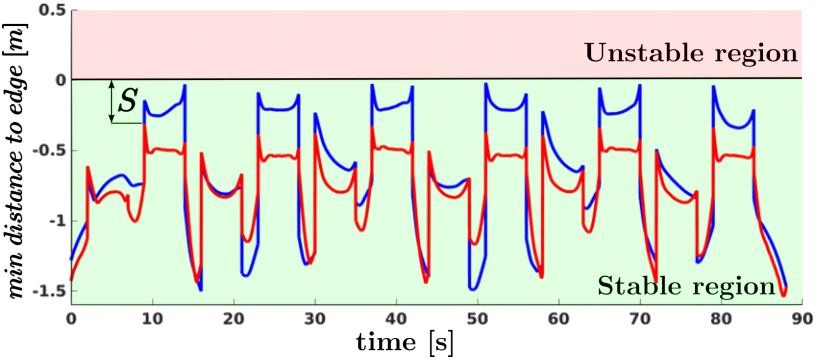


Figure 3.8: Minimal distance of the CoM to the edge of the support polygon while performing the maneuver shown in Fig. 3.10. Negative distance means that CoM is inside the polygon. Robot remains stable as long as the CoM stays inside the green region. Blue CoM trajectory was obtained with $\epsilon = 0.0$ and the red one with $\epsilon = 0.5$. We define S to be the stability margin.

figure 1, HEAP is given a diagonally shifted goal position. Note how the optimizer automatically discovers crab steering mode without specifying any further details. In Sub-figure 2a - 2d HEAP is driving to a goal position located on the side. It cannot just drive directly to it, nor it can crab steer to it. To reach the goal, HEAP has to make cusps and switch between the forward and reverse driving, which is the solution that our planner finds. Note that because of the kinematic configuration of the legs, it is not possible to turn the wheels in place without violating the lateral slip constraint. For planning the driving maneuvers, the optimization takes ten times shorter than the time horizon of the plan.

3.4.2 Stepping and Walking Motions

We tested the effectiveness of the stepping gait from section 3.3 for overcoming obstacles. Stepping gait uses the arm on the excavator for locomotion. We show two maneuvers that we tested in Fig. 3.6 and Fig.3.7. In Fig. 3.6,

HEAP is asked to cross a gap which is modeled as a parabola (because of the gradient for NLP). Planning such a maneuver is a tough non-convex problem, especially since the robot cannot just jump over the obstacle. It has to slowly coordinate placement of its limbs while standing directly above the gap. This involves turning the arm of the excavator 180 degrees such that it can be used to traverse on the right side of the gap. The NLP solver has to find a feasible solution within a set that is non-convex and not connected. We also show a motion plan where HEAP has to step a 1m high block (see Fig. 3.7). This motion uses the stepping gait as well. For both gap crossing and stepping on a block, the optimization takes about 2.5 times shorter than the plan length. This is due to the terrain constraints that create a non-convex and not connected feasible set, and it is hard for the optimizer to find a feasible solution. We also show the stepping motion where we command a goal pose that is rotated 90 degrees. Snapshots of the motion are shown in Fig. 3.10. This motion was planned using the walking gait. Note how HEAP turns the arm to the side opposite of the swing leg, to keep balance. The arm motion emerges naturally from our formulation because we consider the whole-body planning problem.

3.4.3 Robust Support Polygon Constraint

We test the effectiveness of the newly proposed stability constraint formulation. We ask the planner to compute motion plans for a 90 degrees stepping turn (see Fig. 3.10). We measure the minimal distance of the CoM the edge of the support polygon. The plot of minimal distance over time is shown in Fig. 3.8. Green shaded region denotes stable states, whereas CoM crossing into the red region will cause the robot to fall. It can be seen that our formulation successfully enforces greater minimum distances of CoM to the edge of the support polygon (greater stability margin $\mathcal{S}$).

3.4.4 Contact Schedule Optimization

We show the contact schedule optimization on a case where it is not possible to achieve the given goal without optimizing over the timing. The motion is shown in Fig. 3.9. On the left, the base is positioned 5m away from the goal position with the arm almost fully extended and the shovel in contact. The total duration of the planned motion is 7 s. The initial timings are set such that the wheels are in contact for all 7 s, and the arm is in contact for 6s. HEAP's kinematic limits prevent it from reaching the goal with the shovel that is in contact and the speed limits (1 m/s max) prevent it from reaching

Figure 3.9: ***Left to Right***: HEAP moving from $(x, y) = (5, 0)$ m to $(x, y) = (0, 0)$ m while simultaneously optimizing over the gait timings. *Left*: The base has to move where the coordinate system axis are. *Middle*: Even with the fully extended arm and tilted base, it is still too far from the goal position to reach it without changing the gait timings. *Right*: The contact timings change and the contact is broken.

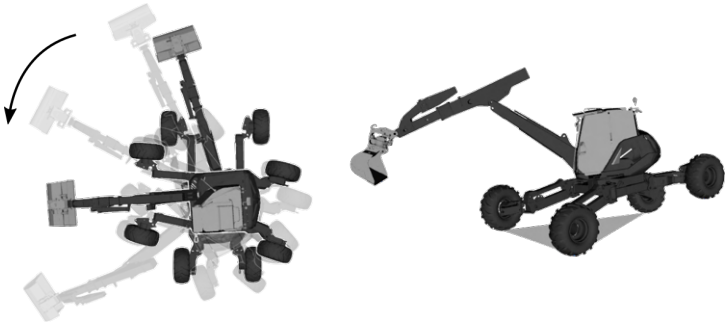


Figure 3.10: ***Left:*** HEAP using the walking gait to turn in place 90 degrees. The initial and the final pose are shown opaque, while the poses in between are shown transparent. The arrow shows the turning direction. ***Right:*** Snapshot from a turning motion. Note how the machine uses the arm as a counterweight to keep balance.

the goal if the pure driving phase stays 1s long. Thus the optimization must figure out that it has to extend the swing phase for the arm and shorten the contact phase (total duration stays the same). The machine extends itself as much as kinematic limits allow, (middle Fig.) but ultimately the contact schedule timings change, and the contact has to be broken (right Fig.) After the optimization, the arm driving phase is 2.2 seconds long. Time to compute the trajectory was 15 s (two times longer than the planning horizon).

3.5 Conclusion and Outlook

We have shown a TO based path planning in rough terrain for legged-wheeled systems. Our formulation allows for planning with both wheeled and non-wheeled limbs, as well as combinations of thereof. We demonstrate the effectiveness of our approach on a walking excavator traversing different types of

terrains. Ability to traverse different terrains, using the same planner shows the advantage of whole-body planning. Our formulation is not robot specific. We show capabilities of the planner to optimize over the gait timings for all EE's, including the ones with wheels. Finally, we introduce an intuitive and straightforward formulation of the support polygon constraint that avoids computation of half-spaces and allows for shrinkage of the support area.

We plan to bring this approach on the real platform and show driving and stepping motions on HEAP. Another possible improvement could be embedding signed distance fields and collision checking. Collision checking would allow us to execute tasks such as manipulation, or stepping over convex obstacles. At the moment we observe that HEAP's legs can collide with the terrain while stepping on a block. Furthermore, since we use a gradient-based method, our approach suffers from all the problems inherent to gradient methods, most hindering one being trapped in the local minima. In the future, one could use sampling-based methods to find a good initial guess for the contact schedule that negotiates the non-convex terrain. A good initial guess for the contact schedule speeds up the optimization drastically. One could also extend the gait generation such that there is a phase where no limbs are in contact, and the base is touching the ground. Such a maneuver is executed sometimes by human operators to reposition all legs at once.

3.6 Appendix

3.6.1 Robust Support Polygon Constraint Derivation

To derive our formulation of the robust support polygon constraint, first, we show how to shrink a support polygon defined with vertices that are centered around the origin of the coordinate system. Subsequently, we show how to reduce the problem to the previous case to derive the form of the constraint that we presented. The derivation is valid for a support area with any positive number of support points. A convex hull for a set of vertices $\boldsymbol{p}_1, .., \boldsymbol{p}_N$ is set of all points $\boldsymbol{x}$ such that:

$$\boldsymbol{x} = \sum_i \alpha_i \boldsymbol{p}_i, \quad \sum_i \alpha_i = 1, \quad \alpha_i \geq 0$$

Assuming that all the vertices defining the convex hull (blue points in Fig. 3.11, left) are centered around the origin, elements $\boldsymbol{x}$ inside the shrunk convex hull can be described as:

$$\boldsymbol{x} = \sum_i \alpha_i \boldsymbol{p}_i, \quad \sum_i \alpha_i = 1 - \epsilon,\ 0 \leq \epsilon \leq 1,\ \alpha_i \geq 0$$

To see this, we define $k = 1 - \epsilon$, and then the constraint becomes: $\sum_i \alpha_i = k$. Dividing the expressions with k and substituting $\tilde{\alpha}_i = \alpha_i / k$ we get:

$$\boldsymbol{x} = \sum_i \tilde{\alpha}_i k \boldsymbol{p}_i = \sum_i \tilde{\alpha}_i \tilde{\boldsymbol{p}}_i, \quad \sum_i \tilde{\alpha}_i = 1,\ \tilde{\alpha}_i \geq 0$$

Here we have defined new points $\tilde{\boldsymbol{p}}_i = k\boldsymbol{p}_i$; Since $k \leq 1$ it is easy to see that we have scaled the radius vectors of all points such that they shift towards the origin. Since the points were centered (origin was in the centroid of the polygon by assumption) the shape of the polygon stays the same and the origin remains the centroid of the polygon. Now we show how to formulate the constraint with points that are not necessarily centered and thus obtain the robust support polygon constraint. The problem formulation is shown in Fig. 3.11, middle and right drawing. We start by defining the centered points $\hat{\boldsymbol{p}}_{com} = \boldsymbol{p}_{com} - \boldsymbol{c}$ and $\hat{\boldsymbol{p}}_i = \boldsymbol{p}_i - \boldsymbol{c}$ where $\boldsymbol{c}$ is the centroid of the polygon $\boldsymbol{c} = \sum_i \boldsymbol{p}_i / n$, and n is the number of vertices of the polygon. Now we can formulate the constraint for the centered set of points:

$$\hat{\boldsymbol{p}}_{com} = \sum_i \alpha_i \hat{\boldsymbol{p}}_i, \quad \sum_i \alpha_i = 1 - \epsilon,\ 0 \leq \epsilon \leq 1,\ \alpha_i \geq 0$$

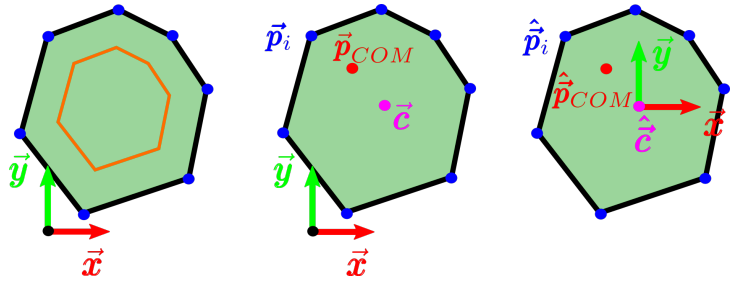


Figure 3.11: ***Left***: Convex polygon and the same polygon in orange color shrunk by a factor of 0.6, ***Middle***: Not centered convex polygon, ***Right***: Centered convex polygon. $\boldsymbol{c}$ denotes the centroid of the polygon, $\boldsymbol{p}_{COM}$ the center of mass, $\boldsymbol{p}_i$ is the i^{th} support point.

Then we substitute back the original variables:

$$\boldsymbol{p}_{com} - \boldsymbol{c} = \sum_i \alpha_i (\boldsymbol{p}_i - \boldsymbol{c})$$

... Steps omitted for the sake of space. We arrive at:

$$\sum_i (\alpha_i + \frac{\epsilon}{n}) \boldsymbol{p}_i = \boldsymbol{p}_{com},\ \sum_i \alpha_i = 1 - \epsilon,\ 0 \leq \epsilon \leq 1,\ \alpha_i \geq 0$$

This is precisely the constraint presented in section 3.2. Note that for $\epsilon = 0$ we recover the standard convex hull constraint and that for $\epsilon = 1$ the support polygon degenerates to a point which is the centroid of the support points.

3.7 Lessons Learned

This paper presented a TO-based motion planner for legged wheeled systems in the presence of complex terrain. The planner transcribes the Motion Planning Problem (MPP) using the collocation method and computes whole-body motion plans. The MPP formulation permits robots with wheels, legs, and combinations thereof (such as HEAP, used as a testing platform).

We learned that optimization is a very powerful tool extremely good at dealing with complex non-linear kinematics and constraints. The proposed planner could compute smooth motions even with a very naive initial guess. However, introducing obstacles slowed down the optimization significantly. Furthermore, we noticed that allowing the optimization to optimize Contact Schedule (CS) often led to long computing times and sometimes loss of convergence. It was clear that the gait should be optimized with a different method.

Explicitly computing half spaces for a support polygon is easy when limb and base planning are decoupled; however, it becomes tough for whole-body planning since the motion of the limbs and base are coupled. Furthermore, it needs to be clarified how to compute gradients for such an operation (the optimizer needs gradients). To address this problem, we developed the novel robust support polygon constraint, which seamlessly integrates inside the planner and it was used inside the optimization in every subsequent paper.

4

Terrain-Adaptive Planning and Control of Complex Motions for Walking Excavators

Jelavic, E., Berdou, Y., Jud, D., Kerscher, S., and Hutter, M. (2020). "Terrain-adaptive planning and control of complex motions for walking excavators". In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, pp. 2684–2691

DOI: 10.1109/IROS45743.2020.9341655

Video: https://youtu.be/7QU1UmnMy1Q

This article presents a planning and control pipeline for legged-wheeled (hybrid) machines. It consists of a Trajectory Optimization based planner that computes references for end-effectors and joints. The references are tracked using a whole-body controller based on a hierarchical optimization approach. Our controller is capable of performing terrain adaptive whole-body control. Furthermore, it computes both torque and position/velocity references, depending on the actuator capabilities. We perform experiments on a Menzi Muck M545, a full size 31 Degrees of Freedom (DoF) walking excavator with five limbs: four wheeled legs and an arm. We show motions that require full-body coordination executed in realistic conditions. To the best of our knowledge, this is the first work that shows the execution of whole-body motions on a full size walking excavator, using all DoFs for locomotion.

4.1 Introduction

Robotic platforms are becoming more sophisticated and required to operate in increasingly challenging environments. One example are hybrid legged-wheeled[1] systems that offer versatility in a variety of terrain. They combine the agility of legged systems with the speed of wheeled systems in flat terrain. However, planning and control algorithms suitable for hybrid systems remain somewhat underdeveloped compared to their purely legged or wheeled counterparts. To this end, we present a motion planner capable of producing robust, terrain-aware motion plans, suitable for execution on real platforms. Planning with entirely accurate terrain information is not possible, due to the imperfect sensor data and spatial map discretization. Furthermore, too many details and exact geometry (e.g., vegetation roughness, mud) can still render Trajectory Optimization (TO) very hard. Hence, it is desirable to be able to plan on simplified geometry and retain robustness at the execution time. We tackle the robustness problem by designing the terrain-adaptive controller suitable for machines that are not fully torque-controllable. Our framework has been validated on a hydraulic walking excavator *HEAP (Hydraulic Excavator for an Autonomous Purpose)* 2018 in both simulations and experimentally[2].

4.1.1 Related Work

Motion planning and control for legged systems, and for hybrid systems in particular, are nowadays typically done using TO based approaches due to their favorable scaling with the system dimension. Often, planning algorithms run in a receding horizon fashion to maximize robustness, which is beneficial for deployment on the real hardware. Such an architecture has been able to produce various motions for legged robots with point feet (e.g. Grandia et al., 2019b, Bellicoso et al., 2018b, Farshidian et al., 2017a). Despite similarities with legged systems, hybrid systems have received less attention from the robotic community. So far, algorithms for hybrid systems have been developed mainly by the aerospace community. Typically, in extraterrestrial missions, a robot is required to be statically stable while traversing over uneven terrain; an example of such a work can be found in Jarrault, Grand, and Bidaud, 2011, Cordes, Babu, and Kirchner, 2017 or Wilcox, 2012. However, these systems are usually teleoperated in purely reactive mode. There is no planning or whole-body control deployed on the robot.

[1] Legged-wheeled systems are referred to as *hybrid* systems in further text

[2] `https://youtu.be/7QU1UmnMy1Q`

Figure 4.1: ***Top:*** A walking excavator has five limbs; an arm, and four legs with a wheel at the end of each leg. They usually operate in muddy and slippery conditions, which requires motion plans with large stability margin. ***Bottom:*** Hydraulic Excavator for Autonomous Purpose (HEAP) balancing on three legs while performing a stepping maneuver. A video demonstrating all the motions accompanies this submission.

Recently, several authors have investigated the application of TO to whole-body planning and control for hybrid robots. In Bjelonic et al., 2019a, Bjelonic et al., 2019b, *ANYmal* solves an Nonlinear Program (NLP) with box constraints on the end-effector position over a prediction horizon of 0.85 s - 2 s. While motions on the real system look quite impressive, the planning update rates achieved (about 200 Hz) are chiefly thanks to the short prediction horizon and linear inequalities (box constraint) that can be used to enforce end-effector range of motion. In contrast, the kinematics of a walking excavator (absence of the knee joint) does not permit such a simplification, and one has to solve for the joint positions as well (see Jelavic and Hutter, 2019). This renders the NLP considerably harder, which then results in a decreased Real-time (RT) factor. In Viragh et al., 2019, *ANYmal* performs hybrid stepping and driving motions generated by solving a Quadratic program (QP) with a prediction horizon of 2 s. The authors introduce a simplified Zero Moment Point (ZMP) criterion that allows for planning at 50 Hz while the torque computation runs at 400 Hz. Similar to Viragh et al., 2019, *Robosimian* Bellegarda and Byl, 2019 shows impressive dynamic driving motions computed for a 2 s planning horizon. However, both robots perform motions only in simulation

and are typically deployed in less harsh environments compared to HEAP, see Fig. 4.1.

Momaro Klamt and Behnke, 2018 and more recently, *Centauro* Klamt et al., 2018 both perform stepping and driving maneuvers over uneven terrain. Contrary to the approaches outlined above, *Momaro* and *Centauro* do not blindly locomote, but rather maintain a surroundings map that is used to decide when to drive and when to step. However, neither robot performs whole-body motion planning, and transitions between driving and motions are handcrafted. In contrast, our planning and control pipeline computes and executes whole-body plans, which is a more scalable approach for robots with many DoFs.

We propose an approach that decouples the problem into sequential offline motion planning and online tracking phases. Few previous works have approached the problem in this way. In Winkler et al., 2018, a plan involving dynamic motions is computed for a robot with point feet. The motion is tracked using a hierarchical Whole Body Controller (WBC) Bellicoso et al., 2016. In Medeiros et al., 2019, whole body plans are computed in a simplified 2D scenario and then extended to 3D with results only being shown in simulation. *Skaterbots* Geilinger et al., 2018, use a general framework that allows for generation of different hybrid motions. While demonstrated motions are quite challenging and look smooth, they have only been demonstrated on small robots in laboratory conditions. In its present form, it remains an open question of how the planning and control algorithm would transfer to a large scale robot with a high degree of modeling and environment uncertainty.

Typically, whole-body control is associated with torque-controlled robots and there is rich literature covering the topic Bellicoso et al., 2019, Fahmi et al., 2019, Grandia et al., 2019b, Bjelonic et al., 2019a, Bjelonic et al., 2019b. However, many existing machines do not feature actuators for high accuracy torque control (e.g. large friction, delays) or do not have all actuators torque-controllable. Furthermore, models of the system dynamics are only available with limited accuracy. In contrast, we develop a control framework that can handle a mixture of torque-controlled and position/velocity controlled DoFs.

4.1.2 Contribution

We present a planning and control pipeline for legged wheeled machines. In particular, we focus on large scale robots with many DoFs such as walking excavators. Our contributions can be summarized as follows: We extend the motion planner introduced in our previous work Jelavic and Hutter, 2019 for

execution on the real hardware. Besides, we design a tracking controller for executing challenging whole-body motions on a real machine. All of the motions shown are computed and executed using the same pipeline, i.e., the same planner and the same controller without any changes. We use an extended terrain-adaptive whole-body controller based on Hierarchical Optimization (HO) framework Bellicoso et al., 2016. The controller handles both torque and position/velocity controllable DoFs in a single structure. Lastly, our framework has been demonstrated on a full-scale hydraulically-actuated excavator in a realistic environment. We show that the proposed approach can execute challenging motions despite the mud, high actuator friction, and limited model accuracy available for HEAP. To the best of our knowledge, this is the first time whole-body motions have been shown on a full size walking excavator.

4.2 Planning

In our previous work Jelavic and Hutter, 2019, we introduced a collocation Hargraves and Paris, 1987 based planner that solves an optimal control problem. It produces kinematically consistent plans while respecting the non-holonomic rolling constraint for the wheels. In this article our planner has been extended with additional constraints and analytical costs, to make plans executable on real hardware.

4.2.1 Notation

Before introducing any equations, we introduce the notation used. Legs are denoted with LF (Left Front), RF, LH, RH (Right Hind). The inertial frame is denoted with I and B denotes the floating base of the robot. The left subscript indicates the frame in which the quantity is expressed, e.g., ${}_I\boldsymbol{r}_{IB}$ denotes the position of the base with respect to the inertial frame expressed in the inertial frame. For rotation, we use quaternions or rotation matrices, where $\boldsymbol{R}_{IB}$ ($\boldsymbol{q}_{IB}$) is a rotation of the base with respect to the inertial frame. We use $\boldsymbol{v}$ for linear velocities; ${}_I\boldsymbol{v}_{ee}$ is a linear velocity of the end-effector expressed in the inertial frame. Angular velocity is denoted with $\boldsymbol{\omega}$, ${}_I\boldsymbol{\omega}_{IB}$ is the angular velocity of the base frame as seen from the inertial frame expressed in the inertial frame. Further examples using the same convention can be found in Furgale, 2014. We denote desired quantities with the right superscript d, e.g., ${}_I\boldsymbol{v}_B^d$ is a desired base linear velocity, expressed in the inertial frame. Right subscript denotes a vector component, e.g. $\boldsymbol{v}_x$ is the x component of vector

$\boldsymbol{v}$. Finally, the generalized coordinate vector $\boldsymbol{q}$ and the generalized velocity vector $\boldsymbol{u}$ are given with:

$$\boldsymbol{q} = \begin{bmatrix} {}_I\boldsymbol{r}_{IB} \\ \boldsymbol{q}_{IB} \\ \boldsymbol{q}_j \end{bmatrix} \in SE(3) \times \mathbb{R}^{n_j}, \boldsymbol{u} = \begin{bmatrix} {}_I\boldsymbol{v}_B \\ {}_B\boldsymbol{\omega}_{IB} \\ \dot{\boldsymbol{q}}_j \end{bmatrix} \in \mathbb{R}^{n_u} \tag{4.1}$$

where $n_u = 6 + n_j$ and n_j is the number of joints. $\boldsymbol{q}_j \in \mathbb{R}^{n_j}$ is the vector of joint coordinates.

4.2.2 Extensions

The planner uses a set of constraints, introduced in our previous work Jelavic and Hutter, 2019. We describe the newly added planner components with the equations given below.

Base motion constraint: Box constraint on the base roll angle encourages the optimization to primarily use the arm for balancing and not to tilt the base. Large roll angles θ of the base shrink the support polygon and could lead to catastrophic falls. In addition, a limit on the angular velocity of the base discourages fast motions.

$$|\theta| \leq \theta_{max}, \ {}_B\boldsymbol{\omega}_{IB} \leq \boldsymbol{\omega}_{max} \tag{4.2}$$

End-effector motion constraint: We do not allow fast movement of the limbs. Fast movement of the limbs may cause the controller to request too much actuation, rendering the tracking problem infeasible (e.g. not enough oil flow from the hydraulic pump). The second issue that the planner does not account for inertial forces when limbs are swinging. Hence, high velocities can lead to poor base tracking performance.

$$|\boldsymbol{v}_{EE}| \leq \boldsymbol{v}_{EE,max} \tag{4.3}$$

Nominal posture terminal cost: The planner is encouraged to finish the motion in the nominal posture. This makes the planning the next maneuver easier since the nominal posture is stable and joint positions are far away from the bounds. We denote cost functions or cost terms with J.

$$J_{nominal} = (\boldsymbol{q}_T - \boldsymbol{q}_n)^T \boldsymbol{S} (\boldsymbol{q}_T - \boldsymbol{q}_n) \tag{4.4}$$

where $\boldsymbol{S}$ is the cost matrix and $\boldsymbol{q}_T$ and $\boldsymbol{q}_n$ are terminal and nominal joint positions, respectively.

Wheel velocity difference cost: Since HEAP cannot control each wheel's velocity individually, we encourage the planner to find the solutions where all wheels have similar velocities. In this way, it easier for controller to track the motion and the amount of slip is reduced.

$$J_{wheel,i,j} = \sum_{t=0}^{T} \|v_i^2 - v_j^2\|_2^2 \tag{4.5}$$

where v_i is the linear velocity magnitude of the i^{th} wheel and $(i,j) \in \{(LF,RF),(LH,RH)\}$. The effect of the cost on the wheel magnitude difference is illustrated in Figure 4.2.

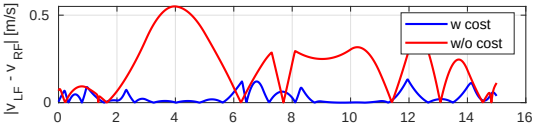


Figure 4.2: Difference in velocity magnitude between Left Front (LF) and Right Front (RF) wheel during the sideways driving maneuver. The base command was to drive left. It can be seen that the introduction of the cost reduces speed difference by an order of magnitude over the trajectory duration. Motions that require less velocity difference are easier for the controller to track.

Similar to our findings, Melon et al., 2020 also reports introducing costs in the optimization improves the quality of the planned motion. Unfortunately, improved quality comes with more computation time (about 2x decrease of the RT factor).

4.3 Control

The proposed control system is based on HO Bellicoso et al., 2016. HO controller essentially implements an inverse dynamics algorithm Siciliano et al., 2010. Such a control scheme allows HEAP to be force controlled. Despite operating in purely tracking mode, force control can be beneficial since the machine can adapt to unperceived terrain changes (e.g., bumps or puddles not captured in the planning phase). The schematic of the proposed whole-body controller is shown in Fig. 4.3. Fig. 4.4 shows the excavator and all the DoFs.

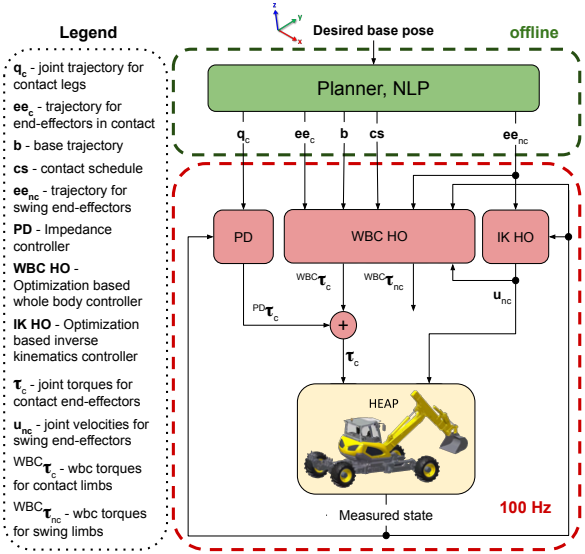


Figure 4.3: Structure of the proposed planning and control system. The planner (green color) sends the desired references to the controller (red color). The WBC computes desired torques for both contact and swing end-effectors. In addition to the WBC, we run an impedance controller for the stance end-effectors and the Inverse Kinematics (IK) controller for the swing end-effectors. Impedance controller torques are added to the WBC torques and sent to the contact limbs. Velocities computed from IK are sent to the low-level controller. Note how the WBC torques computed for swing limbs ${}^{WBC}\boldsymbol{\tau}_{nc}$ are not used. While the planner runs offline, the proposed controller runs at 100 Hz. The limiting factor in achieved frequency is the bandwidth of the Controller Area Network (CAN) bus.

4.3.1 Whole-Body Control

The WBC is the core controller of our framework. In addition, we extended it with two more controllers (see Fig. 4.3) in order to compute references for both dynamically (torque) controlled joints and kinematically (position/velocity) controlled joints. The control system receives operational space references from the planner and computes optimal generalized accelerations $\dot{\boldsymbol{u}}^*$ and contact forces $\boldsymbol{\lambda}^*$, i.e. the solution vector $\boldsymbol{x}^*$ looks like: $\boldsymbol{x}^* = [\dot{\boldsymbol{u}}^{*T}\boldsymbol{\lambda}^{*T}]^T \in \mathbb{R}^{n_u+3n_c}$, where n_c is the number of end-effectors that are in contact.

WBC solves a series of QP problems in a prioritized order. A solution of a QP with lower priority is computed in the nullspace of the QP with higher priority. For each QP, inequality constraints $\boldsymbol{A}_i\boldsymbol{x} \leq \boldsymbol{b}_i$ and equality constraints $\boldsymbol{C}_i\boldsymbol{x} = \boldsymbol{d}_i$ define a task T_i where index i denotes the task priority. Matrices

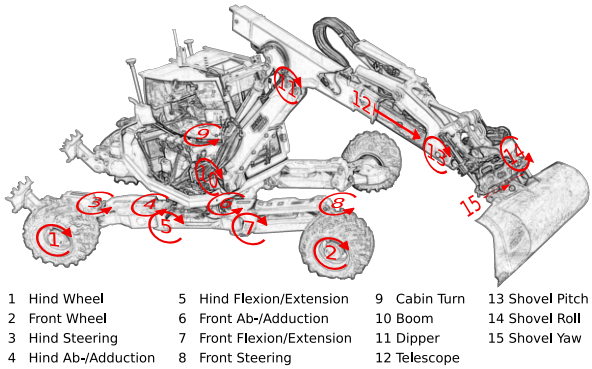


Figure 4.4: Schematic of HEAP with all joints and their axes. Joint notations on RF and Right Hind (RH) legs are omitted for the sake of clarity. They have the same DoF's as the LF and Left Hind (LH) leg that are shown. Joints 1, 2, 9, 14 and 15 cannot be torque controlled. Joint 12 has a very high and nonlinear static friction which makes the torque control hard. The rest of the joints can be torque controlled. Figure adapted from Jud et al., 2021b.

$\boldsymbol{A}$ and $\boldsymbol{C}$, together with vectors $\boldsymbol{b}$ and $\boldsymbol{d}$ are task-dependent and have to be specified by the user. By adding tasks T_i and assigning their priorities in a meaningful way, one can shape the behavior of the robot (see Bellicoso et al., 2016).

4.3.2 Tasks

The tasks used in WBC with their respective priorities are specified in Table 4.1. Before providing the formulation details, we explain how the priorities for the tracking tasks are chosen.

The base tracking task is split into sub-tasks with different priorities to reduce the number of DoFs in the optimization. Unconstrained optimization variables can have a detrimental effect on the quality of the motion (bang-bang solutions) and may produce unnecessary motions. Looking at the kinematics of the excavator in Fig. 4.4, it can be seen that roll, pitch and height of the base can be controlled using Flexion/Extension (FE) joints (joints number 5 and 7 in Fig. 4.4); other joints have a minor contribution. This holds even for the case when one leg is in the air. Hence, the base tracking task is split into terrain adaptive posture tracking (roll, pitch and height that are influenced by FE joints) and 2D pose tracking (x, y and yaw influenced by Abduction/Adduction (AA) and steer joints). Exploiting the hierarchical task

setup to achieve posture adaptation has been reported in Bellicoso et al., 2019. We exploit hierarchical task setup to realize the terrain adaptive behavior. Prioritizing the adaptive posture tracking tasks in the HO over the 2D pose tracking tasks achieves the desired goal and the machine adapts to the terrain (see Section 4.4.1). In addition, the HO is more constrained which helps finding steadier motions.

Wheel and base 2D pose (x, y and yaw) are most influenced by the same set of joints. These are steering (numbers 1 and 2 in Fig. 4.4) and AA joints (numbers 4, 5, 6 and 8). For this reason, the base and wheel 2D pose tracking tasks should have the same priority. Otherwise, the DoFs may be used to satisfy one task perfectly, and then the lower priority task could be rendered infeasible. It is worth observing that given the fixed positions of the wheels, the position of the base is largely determined. There is little space to move the base without violating the wheel rolling constraint, which means that accurate wheel tracking implies accurate base tracking. For this reason, we tune the wheel tracking gains to be higher than the base for the 2D pose tracking tasks.

Table 4.1: WBC task setup. Smaller number indicates higher priority.

Priority	Task
1	Floating base equations of motion
1	Joint limits
1	Friction cone
1	Wheel rolling constraint
2	Base orientation (*pitch, roll*)
2	Base translation (*height*)
3	Base orientation (*yaw*)
3	Base translation (*lateral, longitudinal*)
3	Ground leg orientation (*yaw*)
3	Ground leg translation (*lateral, longitudinal*)
4	Swing limb orientation
4	Swing limb translation
5	Joint acceleration minimization
5	Contact force minimization

Floating base equations of motion: We require that computed solution $\boldsymbol{x}^*$ satisfies rigid body Equations of Motion (EoM). For details on the EoM see Bjelonic et al., 2019a, Siciliano et al., 2010 or Featherstone, 2014.

Joint limits: This task implements box constraints on position, velocity and torque joint limits, i.e.

$$\boldsymbol{q}_{j,lower} \leq \frac{1}{2}\ddot{\boldsymbol{q}}_j \Delta t^2 + \dot{\boldsymbol{q}}_{j,t}\Delta t + \boldsymbol{q}_{j,t} \leq \boldsymbol{q}_{j,upper} \tag{4.6}$$

$$\dot{\boldsymbol{q}}_{j,lower} \leq \ddot{\boldsymbol{q}}_j \Delta t + \dot{\boldsymbol{q}}_{j,t} \leq \dot{\boldsymbol{q}}_{j,upper} \tag{4.7}$$

$$\boldsymbol{\tau}_{j,lower} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{upper} \tag{4.8}$$

Since the optimization has accelerations as decision variables, we need to integrate one time step to enforce the position and bounds limits. Δt denotes time step, $\boldsymbol{q}_{j,t}$ are the joint positions and $\dot{\boldsymbol{q}}_{j,t}$ joint velocities at time step t, respectively.

Friction Cone: The friction cone is approximated by a friction pyramid (to have linear constraints). Details about the implementation of the constraint can be found in Bellicoso et al., 2017.

Wheel rolling constraint: Wheel rolling constraint ensures that there is no lateral movement of the wheel, i.e., the wheel can only move in the longitudinal direction. The derivation of the rolling constraint can be found in Bjelonic et al., 2019a.

Base tracking: We use a standard formulation that be commonly found for floating base robots, e.g., Bellicoso et al., 2017, Bellicoso et al., 2016. The equations are given below:

$$\begin{aligned}[\boldsymbol{J}_{B,r} \;\; \boldsymbol{0}_{3\times 3n_c}]\boldsymbol{x} = \boldsymbol{R}_{WI}\big({}_I\ddot{\boldsymbol{r}}^d_{IB} + \boldsymbol{K}_p({}_I\boldsymbol{r}^d_{IB} - {}_I\,\boldsymbol{r}_{IB}) \\ + \boldsymbol{K}_d({}_I\dot{\boldsymbol{r}}^d_{IB} - {}_I\,\dot{\boldsymbol{r}}_{IB})\big) - \dot{\boldsymbol{J}}_{B,r}\boldsymbol{u}\end{aligned} \tag{4.9}$$

$$\begin{aligned}[\boldsymbol{J}_{B,q} \;\; \boldsymbol{0}_{3\times 3n_c}]\boldsymbol{x} = \boldsymbol{R}_{WI}\big({}_I\boldsymbol{\alpha}^d_{IB} + \boldsymbol{K}_p(\boldsymbol{q}^d_{BI} \boxminus \boldsymbol{q}_{BI}) \\ + \boldsymbol{K}_d({}_I\boldsymbol{\omega}^d_{IB} - {}_I\,\boldsymbol{\omega}_{IB})\big) - \dot{\boldsymbol{J}}_{B,q}\boldsymbol{u}\end{aligned} \tag{4.10}$$

where $\boldsymbol{J}_{B,r}$ is a translational Jacobian and $\boldsymbol{J}_{B,q}$ si a rotational Jacobian. The desired angular acceleration of the base is denoted with ${}_I\boldsymbol{\alpha}^d_{IB}$. Operator $\boxminus : SO(3) \to \mathbb{R}^3$ is defined in Bloesch et al., 2016.

Ground leg translation: With steerable wheels, one can control both lateral and longitudinal translation of the wheel. We close the control loop over the longitudinal error (distance to W^d along the e_x axis in Fig. 4.5a) and lateral

error (distance to W^d along the e_y in the same Figure) resulting in the equality constraint:

$$\pi_j\big([\boldsymbol{J}_{W,r}\ \ \boldsymbol{0}_{3\times 3n_c}]\boldsymbol{x}\big) = \pi_j\big(\boldsymbol{R}_{WI}\big({}_I\ddot{\boldsymbol{r}}^d_{IW} + \boldsymbol{K}_p({}_I\boldsymbol{r}^d_{IW} - {}_I\,\boldsymbol{r}_{IW}) + \boldsymbol{K}_d({}_I\dot{\boldsymbol{r}}^d_{IW} - {}_I\,\dot{\boldsymbol{r}}_{IW})\big) - \dot{\boldsymbol{J}}_{W,r}\boldsymbol{u}\big) \tag{4.11}$$

where $\boldsymbol{J}_{W,r}$ is a translational Jacobian in wheel frame W (see Figure 4.5b). The operator $\pi_j(\cdot), j \in \{x, y\}$ is the projection onto the wheel longitudinal axis e_x (red color in Figure 4.5a) and lateral axis e_y (green color).

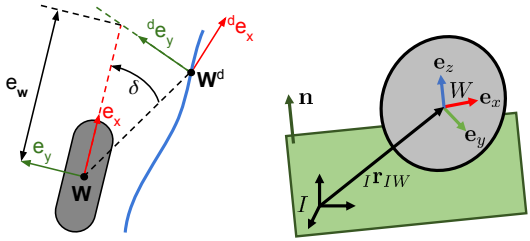
Ground leg orientation: Since wheels introduce non-holonomic constraints, one has to control the yaw angle of the wheel to track the position. Equalities describing the ground leg orientation task are given by:

$$\pi_z\big([\boldsymbol{J}_{W,q}\ \ \boldsymbol{0}_{3\times 3n_c}]\boldsymbol{x}\big) = \pi_z\big(\boldsymbol{K}_p\boldsymbol{R}_{WI}(\boldsymbol{q}^d_{WI} \boxminus \boldsymbol{q}_{WI}) + \boldsymbol{K}_d\boldsymbol{R}_{WI}({}_I\boldsymbol{\omega}^d_{IW} - {}_I\,\boldsymbol{\omega}_{IW}) + \boldsymbol{K}_{p,e}(\dot{q}^{ref}_{j,w})\boldsymbol{e} - \dot{\boldsymbol{J}}_{W,q}\boldsymbol{u}\big) \tag{4.12}$$

$$\boldsymbol{e} = [0\ 0\ sign(\dot{q}^{ref}_{j,w})\delta]^T \tag{4.13}$$

$$\delta = atan2({}_W\boldsymbol{r}_{WW^d,x},\ {}_W\boldsymbol{r}_{WW^d,y}) \tag{4.14}$$

The matrices $\boldsymbol{K}_P, \boldsymbol{K}_{p,e}$, denote proportional gains, and $\boldsymbol{K}_D$, derivative gains. All the matrices are diagonal and belong to a set of positive definite matrices $\mathbb{S}^3$. The orientational Jacobian expressed in the wheel frame W is given with $J_{W,q}$. Wheel joint velocity reference (joints number 1 and 2 in Figure 4.4) is denoted with $\dot{q}^{ref}_{j,w}$. We give more details about $\dot{q}^{ref}_{j,w}$ computation in Section 4.3.5. Besides tracking the orientation given from the plan, we introduce an additional feedback term $\boldsymbol{K}_{p,e}\boldsymbol{e}$ over the desired position in the Equation 4.12. This term drives the wheel directly to the desired position (origin of $\boldsymbol{W}^d$ in Fig. 4.5a). The $sign(\cdot)$ function in Equation (4.13) is used to correctly handle the reverse driving. Such a control law is similar to the pure pursuit tracking algorithm Coulter, 1992, with the main difference that we do not impose moving along a circular arc towards the goal point. The pure pursuit algorithm computes a circular arc trajectory since it has been designed for wheeled robots with front steering when both front and rear wheel pairs have to point to the same center of rotation. However, because of the additional degree of freedom in the leg (the AA joints), HEAP's wheels do not have to point to the same center of rotation. The gain $\boldsymbol{K}_{p,e}(\dot{q}^{ref}_{j,w})$ is an adaptive gain on the $[q_{w,min},\ q_{w,max}]$ interval. Such a practice is common in the vehicle control community Kuwata et al., 2008.



(a) Wheel top view **(b)** Wheel frame

Figure 4.5: ***Left:*** Top view of a wheel tracking the desired trajectory (shown in blue color). The desired position and orientation of the wheel center at time t are denoted with a coordinate system $\boldsymbol{W}^d$. Coordinate system $\boldsymbol{W}$ denotes the current position and orientation. Wheel axes are shown in red and green colors. The yaw error δ from Equation (4.14) is denoted with δ in the Figure. Position error along the rolling direction is shown with a full black line and denoted with symbol e_w. ***Right:*** The wheel frame W is defined to lie on the origin of the wheel. The axis $\boldsymbol{e}_y$ is defined as the wheels rotation axis, and the axis $\boldsymbol{e}_x$ is defined as the direction perpendicular to $\boldsymbol{e}_y$ and the terrain normal $\boldsymbol{n}$. The wheel moves in the direction of the axis $\boldsymbol{e}_x$. Note that the wheel frame W does not rotate with the wheel.

Swing limb tracking: For swing limb motion tracking, we use the same formulation as for the base (see Equatinos (4.9) and (4.10). The base Jacobians J_B are replaced with limb end-effectors Jacobians J_i, $i \in \{W, A\}$, depending on whether we control the wheel frame W or the arm end-effector frame A. Desired and measured coordinates for the base are replaced with desired and measured coordinates for the limb end-effector.

Joint acceleration and contact force minimization: The task minimizes the joint accelerations by setting $\ddot{\boldsymbol{q}}_j = \boldsymbol{0}$. It also tries to minimize the contact forces to get rid of internal forces acting on the robot Bellicoso et al., 2017.

4.3.3 Impedance control

For the end-effectors that are in contact, we add torques computed using the Proportional-Derivative (PD) controller ${}^{PD}\boldsymbol{\tau}_c = \boldsymbol{K}_p(\boldsymbol{q}_j^d - \boldsymbol{q}_j) + \boldsymbol{K}_d(\dot{\boldsymbol{q}}_j^d - \dot{\boldsymbol{q}}_j)$. $\boldsymbol{q}_j^d$ and $\dot{\boldsymbol{q}}_j^d$ are known from the planner (denoted $\boldsymbol{q}_c$ in Fig. 4.3). The PD action helps to combat the friction effects and model inaccuracies, which become prominent at low torques. E.g., if the arm is extended far in the front, contact forces at the hind legs become low, and there is less torque in the joints. At low torques, Coulomb friction can cause unwanted chattering. This issue is ameliorated by adding some torque computed from the PD controller. The net effect is that the legs are "stiffened up", i.e., the actuators operate in a

higher frequency range where the friction effects are not as strong. Instead of PD control, a lead-lag controller could also be used.

4.3.4 Inverse Kinematics Control

The IK controller is also based on HO and computes the desired joint velocities $\dot{\boldsymbol{q}}_{nc}$. To make the swing limbs (including the arm) accurately follow a planned motion trajectory, we use joint velocity control as this can much better compensate for modeling errors and highly-dominant friction effects. Furthermore, some joints on our machine are simply not torque-controllable. One way to make the WBC and IK work together is to let the IK compute desired joint velocities and then add them as a constraint in the WBC. This constraint is merely another task T_i in the WBC whose priority should be lower than the joint limits task. The task can be formulated as an equality constraint:

$$\ddot{\boldsymbol{q}}_j \Delta t + \dot{\boldsymbol{q}}_{j,t} = \dot{\boldsymbol{q}}_{j,IK} \tag{4.15}$$

Such a task should only be added for non-contact limbs. Incorporating the IK into the WBC in this way ensures that solution $\boldsymbol{x}^*$ satisfies the EoM (highest priority task in the WBC). However, Equation 4.15 can introduce unwanted noise in the WBC which is why we do not add such a task.

We merely add the swing limb tracking task in the WBC controller (see Table 4.1). The torques computed for the swing limb cannot be sent to the actuators (some are not torque-controllable, furthermore joint friction is prominent without any load). However, by enforcing the swing limb tracking task, the WBC becomes aware of the swing limb motion and accounts for inertial effects on the base. As a consequence, the IK controller does not deteriorate the base tracking because the WBC has already accounted for the swing limb motion. In this way, the two controllers (WBC and IK) can work together without conflicts. In addition, the inertial effects are minimized in the planning phase since the motion planner does not request any dynamic motions. In Fig. 4.3, note how the torques ${}^{WBC}\boldsymbol{\tau}_{nc}$ for swing limbs computed by the WBC are neglected and velocities from IK are used for control. Validation of our approach is shown in Fig. 4.6.

4.3.5 Wheel Speed Control

The turning speed of the wheels cannot be measured directly since wheel actuators do not have encoders or any sensors. Therefore it is not possible to close the control loop over the wheel joint (see Fig. 4.4) velocities. To

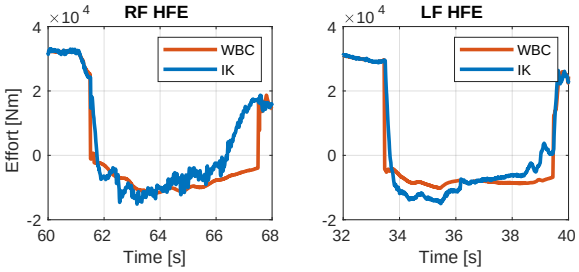


Figure 4.6: Torques in LF and RF flexion joints during the swing phases while performing stepping experiment. It can be seen the torques for swing limb joints computed by the WBC do not substantially differ from the actual torques when tracking the velocities from the IK controller.

address this problem, we close the loop over the wheel position. Furthermore, the wheel joints are not torque controllable either. Nonetheless, we still let the WBC include them in the optimization problem as if they were torque controllable. The same reasoning as for the swing limbs applies (see Section 4.3.4).

The reference wheel speed $\dot{q}_{j,w}^{ref}$ is calculated using Equation 4.16. K_{ff} denotes the feed-forward gain multiplying the desired speed obtained from the planner, and K_p and K_i are proportional and integral gain, respectively. The reference current for the valves, is then calculated as $i_v^{ref} = l(\dot{q}_{j,w}^{ref})$. Note that function l is unknown and hard to identify.

$$\dot{q}_{j,w}^{ref} = K_{ff}\dot{q}_{j,w}^{d} + K_p e_w + K_i \int_0^t e_w(t)dt \tag{4.16}$$

Where position error e_w (see Fig. 4.5a) is calculated as the distance from the measured wheel position along its rolling direction $\boldsymbol{e}_x$ to the plane spanned by the vectors $\boldsymbol{e}_y^d$ and $\boldsymbol{e}_z^d$ of the desired wheel frame.

$$e_w = \frac{{}_I\boldsymbol{e}_x^d \cdot \left({}_I\boldsymbol{r}_{IW}^d - {}_I\boldsymbol{r}_{IW}\right)}{{}_I\boldsymbol{e}_x^d \cdot {}_I\boldsymbol{e}_x}, \tag{4.17}$$

where ${}_I\mathbf{r}_{IW}$ is the position of the wheel origin in the world frame and ${}_I\mathbf{e}_x$ is the rolling direction of the wheel in the world frame.

4.4 Results and Discussion

We perform experiments on HEAP, a full-size 12-ton walking excavator. HEAP is equipped with 12 torque-controllable and 4 position/velocity controllable joints in the chassis. The arm has 4 torque-controllable joints and 3 position/velocity controllable joints . The wheel joints in the chassis (numbers 1 and 2 in Figure 4.4) are not torque controllable, together with the cabin turn, shovel roll, and shovel yaw (numbers 9, 14 and 15). The planner is run offline on a laptop with Intel Xeon E3-1535M v5 2.90GHz processor. It is implemented using Ipopt NLP solver Wächter and Biegler, 2006 which implements the primal-dual barrier method. Ipopt is interfaced from C++ using Ifopt Winkler, 2018b. Both the planner and the controller use Rigid Body Dynamics Library (RBDL) for rigid body algorithms Felis, 2017. For state estimation we rely on a two-state information filter Bloesch et al., 2017 that fuses Real-time Kinematic (RTK) Global Navigation Satellite System (GNSS) measurements with Inertial Measurement Unit (IMU) measurements from chassis and the cabin. The reader is encouraged to watch the video accompanying this article. In all of our experiments, we keep a human driver inside the cabin for safety.

Figure 4.7: ***Left to Right***: HEAP driving forward over a small hill. The front leg adapts its height to the ground height, despite motion planner producing plans under the flat ground assumption.

4.4.1 Reactive controller behavior

Fig. 4.7 shows the terrain adaptive control behavior induced by our controller. Fig. 4.8 shows the tracking performance. The planner thinks that the ground is flat and comes up with a plan to move forward. However, there is a small hill in the way, and the controller adjusts the front leg position to keep the base upright. This behavior naturally emerges from the task prioritization given in our controller (see Section 4.3.1). Without the torque control in the FE joints, such adaptation would be hard to achieve.

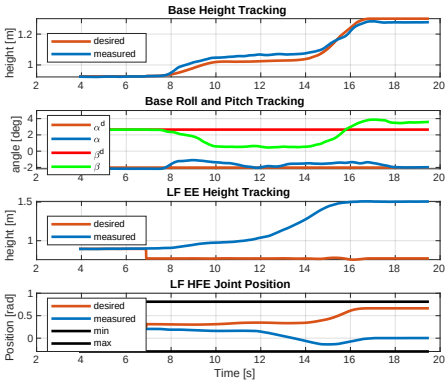


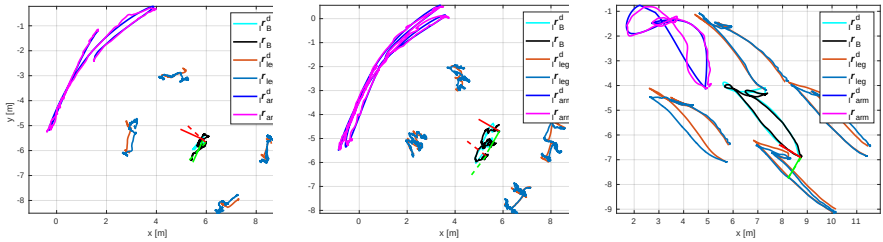
Figure 4.8: Adaptive terrain behavior. Note how the base height, roll α and pitch β are tracked accurately over uneven terrain (see Fig. 4.7). To achieve the base tracking, the controller abandons tracking for the LF end-effector height and the FE joint position. Such a behavior is achieved through task prioritization.

Figure 4.9: ***Left to Right***: HEAP performs a driving maneuver. The base is commanded to move forward and left (frames 1-3). Subsequently, we command the machine to drive back to the starting point (frames 4-6). The planning and control pipeline naturally discovers the crab steering mode of the machine.

4.4.2 Driving motions

We show the ability of our proposed pipeline to generate driving motions in Fig. 4.9. The base is commanded to go forward and left. The optimization discovers the crab steering mode (frame 2), where all wheels point in the same direction. Upon reaching the goal pose (frame 3), we command the machine to drive back to the place where it started. The same crab steering behavior emerges once more (frame 4), this time when driving backward. The motion is completed in frame 6. Note that the machine ends up in a configuration where the base is shifted to the right. This is because the planner receives only the base pose as a goal; the optimization discovers the actual configuration. Computing the plan for driving maneuvers takes about 1 s. Tracking performance is shown in Fig. 4.11c.

Figure 4.10: ***Left to Right***: HEAP performs a stepping maneuver and reorients its base by 30°. The gait pattern requires it to lift the legs in the following order { LH, LF, RH, RF }. The machine cannot complete the maneuver without using the arm for balancing (e.g., in snapshot 3).



(a) Stepping turn **(b)** Stepping sideways **(c)** Driving

Figure 4.11: ***Left:*** Tracking in $x-y$ plane for the base and end-effectors while performing turning maneuver shown in Figure 4.10. The coordinate frame (full lines) denotes starting pose while frame shown in dashed lines shows the end pose. The desired base pose is shown in cyan color, while the desired trajectory of each leg is shown in brown color. The desired trajectory for the arm is shown in blue color, while the actual trajectory is shown in the magenta color. ***Middle:*** Tracking performance for the stepping sideways maneuver (shown in video). The machine is commanded to step sideways two times by 0.5 m ***Right:*** Driving maneuver, shown in Figure 4.9. HEAP was commanded to drive front and right and then to return to the starting point.

4.4.3 Stepping motions

To illustrate the whole-body capabilities of the controller, HEAP executes the stepping maneuver shown in Figure 4.10. In this experiment, we plan a stepping motion to turn the whole machine by 30° in the yaw direction. Motion is then tracked using our whole-body controller proposed in Section 4.3. One can observe that the controller simultaneously coordinates the behavior of the boom, base, and all the legs. The boom is used as a counterweight and turns to the side opposite of the swing limb. Furthermore, the controller shifts the base to keep the Center of Mass (CoM) inside the stable region. In the end, one can observe a successfully reoriented base (snapshot 6). Thanks to the cost introduced in section 4.2, the machine ends the motion in a stable configuration close to the nominal position. Hence, it is ready for re-planning and execution of another turn maneuver if necessary. Computing the plan for

stepping motions takes about 10 s-15 s. Tracking performance in $x-y$ is shown in Fig. 4.11b and 4.11a.

4.5 Conclusion and Outlook

We extend the planning approach introduced in our previous work to make motion plans more robust and executable on a real platform. We design a terrain-adaptive controller that is capable of tracking the motion plans produced by our planner. The controller is based on a whole-body control framework and tailored for robots where not all joints may be torque-controllable, or the torque control may be difficult. Our framework has been experimentally validated through executing challenging motions on a full size walking excavator with 31 DoF. We hope to have advanced large machines towards the mobility level displayed by the best human operators. In the future, we would like to use the arm for locomotion through creating contacts with the ground and not merely balancing. The use of the fifth limb would enable stepping over obstacles, a maneuver commonly executed by humans. Furthermore, we would like to investigate what level of robustness is achievable by controlling as many joints as possible in position mode. Torque control is highly beneficial, however expensive since more cylinders have to be retrofitted with high-performance hydraulic valves.

4.6 Lessons Learned

This paper brings the optimization planner from Paper II to hardware. We extend the planner proposed in Paper II with additional cost terms to facilitate sim to real transfer. We design the terrain-adaptive controller for tracking whole body plans, including driving and stepping motions. Motion plans computed by our planner have been executed on HEAP, a 12-tonne legged excavator.

An important observation from this paper is that hydraulic actuators are highly non-linear and precise torque control is hard to achieve. The situation is alleviated if the joint under consideration is highly loaded. An example of such a joint is Hip Flexion-Extension (HFE) joint. In high load conditions, the friction force is much smaller than the load force, and the relative error in force control decreases. Hence, load-bearing joints can be controlled in the torque mode. For non-load bearing joints, e.g., steering joints, precise torque

control is tough to achieve. Lastly, tuning the torque control loops is very laborious. These observations made us revisit the control system design (Chapter 6).

Planning once offline and executing the plans in purely tracking mode limits the motion duration we can execute. Over time the robot deviates too much from the plan, and it is hard to recover since the controller developed in this work has no look-ahead horizon. The drift is especially prominent for driving motions since HEAP cannot control individual wheel speeds, and there is always some slip during tracking. Limitations on types of maneuvers we could accurately execute spurred us to investigate re-planning, i.e., running the planner in the Model Predictive Control (MPC) fashion.

5

Combined Sampling and Optimization Based Planning for Legged-Wheeled Robots

Jelavic, E., Farshidian, F., and Hutter, M. (2021). "Combined sampling and optimization based planning for legged-wheeled robots". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 8366–8372

DOI: 10.1109/ICRA48506.2021.9560731

Video: https://youtu.be/B-NHY4xwgwY

Planning for legged-wheeled machines is typically done using trajectory optimization because of many degrees of freedom, thus rendering legged-wheeled planners prone to falling prey to bad local minima. We present a combined sampling and optimization-based planning approach that can cope with challenging terrain. The sampling-based stage computes whole-body configurations and contact schedule, which speeds up the optimization convergence. The optimization-based stage ensures that all the system constraints, such as non-holonomic rolling constraints, are satisfied. The evaluations show the importance of good initial guesses for optimization. Furthermore, they suggest that terrain/collision (avoidance) constraints are more challenging than the robot model's constraints. Lastly, we extend the optimization to handle general terrain representations in the form of elevation maps.

5.1 Introduction

Equipping robots with legs enables mobility over unstructured terrains, the same as the animals can traverse them. On the other hand, wheeled robots are not as mobile as their legged counterparts, but they are far more energy-efficient. By combining legs and wheels into a *hybrid system*, roboticists have tried to keep the best characteristics of legged and wheeled systems Bjelonic et al., 2020b; Klamt and Behnke, 2017; Sun et al., 2020. Unfortunately, increased flexibility comes with an increase in complexity, especially true for motion planning since the combinatorial aspect of legged robots (contact schedule) is combined with wheeled robots' non-holonomic nature (rolling constraints). A powerful tool used to cope with the increased complexity is Trajectory Optimization (TO). TO has been shown to work well on hybrid systems with many Degrees of Freedoms (DoFs) (Bjelonic et al., 2020b; Du, Fnadi, and Benamar, 2020; Jelavic and Hutter, 2019). However, optimization is prone to local minima. While some control over solution quality can be achieved through cost function shaping (e.g., Medeiros et al., 2020; Melon et al., 2020), the optimization can ultimately fail due to the non-convexity of the planning problem. Failures happen especially often in challenging terrain where collision (avoidance) constraints become particularly difficult for optimizers. In this letter, we overcome this problem by providing good initializations (*initialization step*) for the optimization-based planner (*refinement step*); i.e we combine Sampling Based Planning (SBP) with TO. Our *initialization step* computes base poses, joint positions, and contact schedule, which are all then fed to the optimization. The gain is that our method can handle systems with many DoF while still being able to cope with challenging terrain.

5.1.1 Related Work

Early research on locomotion for hybrid systems treats the whole system as a driving robot and uses legs as an active suspension. Such a practice is widespread in the aerospace community Cordes, Babu, and Kirchner, 2017; Giordano et al., 2009 and has more recently been applied to wheeled quadrupeds Bjelonic et al., 2019a. The main disadvantage of such an approach is that it does not fully leverage the hybrid system's legged nature, i.e., the robot cannot negotiate obstacles.

To simplify the combinatorial nature of contact schedule planning for legged robots, researchers often use cyclic gaits to restrict the solution space size. Planning with cyclic gaits has been widely used for quadrupedal robots with

Figure 5.1: Hydraulic Excavator for Autonomous Purpose (HEAP) navigating a variety of different terrains (refer to accompanying video for more examples). ***Left-up:*** Driving up a 2 m service ramp. ***Right-up:*** *hole* terrain, traversing over a hole via 1 m wide passage which is too narrow for both legs. ***Left-middle:*** *rough* terrain, navigating a terrain with roughness of ± 1.5 m. ***Right-middle:*** *step* terrain, stepping on a 1 m high block. ***Left-down:*** Stepping over 0.5 m high wall. ***Right-down:*** *gap* terrain, crossing a 2 m wide gap.

point feet (Bellicoso et al., 2018b; Farshidian et al., 2017a; Rebula et al., 2007; Winkler et al., 2017) and more recently, it has been applied to hybrid robots as well (Bjelonic et al., 2020b; Du, Fnadi, and Benamar, 2020; Viragh et al., 2019). Presented controllers run under the flat ground assumption in an Model Predictive Control (MPC) fashion with a prediction horizon of about 2 s. Such a strategy relies on reactive control behavior from MPC, and while it can traverse small irregularities in the terrain, large obstacles still pose an issue. Furthermore, these approaches are not suitable for temporally global plans because of the relatively short prediction horizon.

More recently, terrain-aware planning has been proposed for hybrid robots (Medeiros et al., 2020; Sun et al., 2020), which demonstrate the ability to traverse challenging terrain and plan motions in a whole-body fashion. However, the presented methods solely rely on trajectory optimization and often fall prey to the local minima. Furthermore, the environment is known *excatly*, and it remains unclear how the planner would handle maps generated from real sensory data since discretization and noise can lead to discontinuous gradients in the optimization.

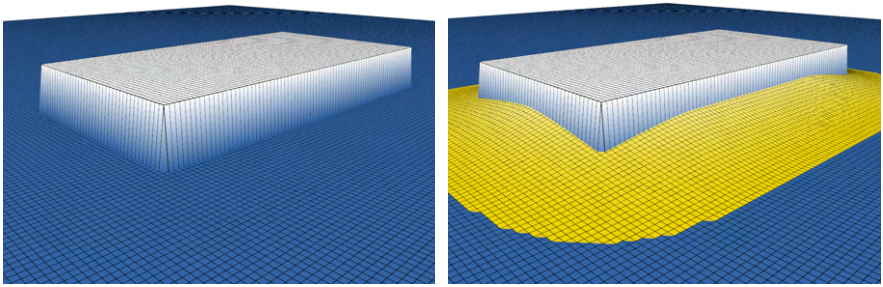
Unlike the optimization, SBP cope well with the non-convex environment. Attempts to use SBPs can be found in Geisert et al., 2019; Tonneau et al., 2018a, where the proposed approach samples base poses in $SE(3)$ space and computes a guiding path for the base of the robot. The footholds are computed in the next stage using the guiding path. However, the proposed approach does not use any optimization, which would make the planning for robots with non-holonomic constraints difficult. Unlike Tonneau et al., 2018a, our approach does not accept a guiding path before having computed the footholds.

Recently, Short and Bandyopadhyay, 2017 has introduced a Contact Dynamic Roadmap (CDRM) data structure for rapid collision checking and foothold generation at runtime. The crux of the approach is computing a Probabilistic Roadmap (PRM) of collision-free configurations offline and then using it for planning online. A similar idea is employed for self-collision avoidance during the initialization phase in our work. We extend the PRM with additional data, so that the approach is applicable to robots with heavy limbs such as walking excavators.

Finally, Klamt et al. Klamt and Behnke, 2017, Klamt and Behnke, 2018 use a graph-search based approach to plan hybrid motions for the Momaro robot. The robot decides when to drive and when to step based on a carefully crafted cost function. The transition sequences between driving and stepping are not computed in a whole-body fashion (as they are handcrafted). Lastly, the graph-search algorithm choice is motivated by the fact that the robot can turn in place using its wheels only. Hence, the approach is unsuitable for robots with a minimal turning radius greater than zero in its current form.

5.1.2 Contribution

We present a combined sampling and optimization based planner for legged-wheeled machines with many DoFs. We use terrain representation to generate a wide variety of locomotion behaviors for navigating complex terrain. The planner is divided into two stages: *Initialization Step* based on a sampling-based planner and *Refinement Step* through nonlinear optimization. These two stages produce kinematically feasible and statically stable plans, motivated by our use case on a walking excavator *HEAP (Hydraulic Excavator for an Autonomous Purpose)* 2018. Given the base's initial and goal pose, our formulation computes base trajectory in $SE(3)$, joint trajectories, and contact schedule. To the best of the author's knowledge, existing approaches provide only high-level waypoints (e.g., Bellicoso et al., 2018a; Klamt and Behnke, 2018; Wermelinger et al., 2016). Finally, our approach is the first (to



(a) Raw terrain **(b)** Filtered terrain

Figure 5.2: ***Left:*** Elevation map of a block with a height jump. The color gradually changes from blue to white, where blue areas are the lowest. ***Right:*** Filtered elevation map shown in yellow color. Areas with dark yellow color have the lowest height. The filtered elevation can be used to compute base pose, as described in Sec. 5.3

the best of our knowledge) to include general terrain representations into an optimization-based planner for hybrid systems.

5.2 Problem Statement

A legged-wheeled robot comprises N limbs with wheels and M limbs without wheels, e.g., a walking excavator has four limbs with wheels, and one non wheeled limb (see *HEAP (Hydraulic Excavator for an Autonomous Purpose)* 2018). The base of the robot can move in $SE(3)$ space. Robot's knowledge about the environment is contained in a map, which is a mapping $h : \mathbb{R}^2 \to \mathbb{R}$ that maps coordinates to various functions describing the environment (e.g., height, traversability). In this letter, a multilayered grid map Fankhauser and Hutter, 2016b data structure is used. Robot's i^{th} limb is in contact with the environment if it is close enough to the surface, i.e.

$$|\boldsymbol{p}_i^z - h(\boldsymbol{p}_i^x, \boldsymbol{p}_i^y)| \leq \epsilon,\ \epsilon \in \mathbb{R}_+ \tag{5.1}$$

where $\boldsymbol{p}^z$ denotes z component of the contact position $\boldsymbol{p}$ and $h(\boldsymbol{p}^x, \boldsymbol{p}^y)$ is height at the contact position. We assume that the contact point is below the center of the wheel (negative gravity direction). For a 30° slope (which is the limit beyond which we consider terrain untraversable), the approximation error is about 15 % of the wheel radius, which is well within the adaptation capabilities of the tracking controller, as demonstrated in Jelavic et al., 2020.

The environment is divided into *traversable* part denoted with $\mathcal{T}$ and *untraversable* part denoted with $\neg\mathcal{T}$. A contact is valid if the contact point lies in the traversable set $\mathcal{T}$, i.e.

$$(\boldsymbol{p}^x, \boldsymbol{p}^y) \in \mathcal{T} \equiv sdf(\boldsymbol{p}^x, \boldsymbol{p}^y) > \delta > 0 \tag{5.2}$$

where $sdf(\cdot,\cdot)$ represents a 2D Signed Distance Function (SDF) with the positive distance meaning the point lies in $\mathcal{T}$. We require all the contact points to stay at least δ away from the $\neg\mathcal{T}$. The 2D SDF is stored as a grid map layer and calculated using marching parabolas Felzenszwalb and Huttenlocher, 2012. Note that, unlike the limbs, robot's base is not required to stay in $\mathcal{T}$ since the base is not in contact with the terrain.

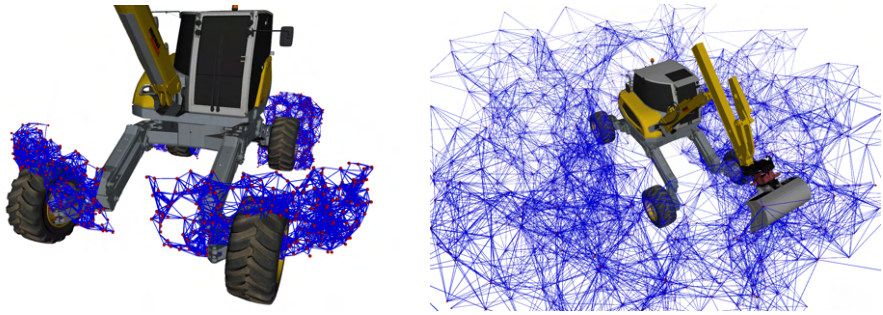
Our goal is to find a trajectory τ with a length of T seconds such that $\boldsymbol{p}_B(t=0) = \boldsymbol{p}_{B,start}$ and $\boldsymbol{p}_B(t=T) = \boldsymbol{p}_{B,goal}$ where $\boldsymbol{p}_{B,start}, \boldsymbol{p}_{B,goal}$ are given starting and goal position for the base of the robot. Apart from the base, we do not enforce any other constraints on robot's pose or joint angles although this is not a hard requirement of our approach.

5.3 Initialization Step

The backbone of the initialization step is a sampling-based planner that samples base poses. We use Rapidly-Exploring Random Trees (RRTs) Karaman and Frazzoli, 2011; LaValle, 2006 although the problem formulation also permits the use of multi-query planners such as PRMs Kavraki et al., 1996. Results presented are mostly generated using and RRT# Arslan and Tsiotras, 2013, whose implementation is taken from Sucan, Moll, and Kavraki, 2012. In addition to RRT, a PRM of limb end-effector positions is precomputed for efficient online planning.

5.3.1 Offline Computation

We use the CDRM data structure introduced in Short and Bandyopadhyay, 2017. CDRM can be used at runtime to generate collision-free movements of the robot's limbs. In our case, CDRM helps us avoid collision between arm and legs while using the arm as a supporting limb. For each limb, the mapping between joint angles $\boldsymbol{q}_i$ and end-effector position ${}^B\boldsymbol{p_i}$ is stored. This mapping does not change over time since ${}^B\boldsymbol{p_i}$ is expressed in the base frame of the robot. Furthermore, we store the mapping between i^{th} limb's Center of Mass (CoM) $\boldsymbol{p}_{i,com}$ and joint configuration for the same limb. This allows us to



(a) Roadmap legs **(b)** Roadmap arm

Figure 5.3: Roadmap vertices (red spheres) and edges (blue lines). Each vertex represents an end-effector position in $\mathbb{R}^3$. ***Left:*** Roadmap shown for the legs of a walking excavator. Shown are 300 vertices and about 2000 edges. ***Right:*** Roadmap for the arm. Shown are 1000 vertices and about 8000 edges.

evaluate the stability criterion during the online planning phase rapidly. The offline roadmap is created using the PRM algorithm; it is shown in Fig. 5.3. The roadmap does not change over time which motivates the use of PRM. In addition to the roadmap, we compute terrain normals and filtered height. This computation can be performed by fitting a tangent plane at each point (x, y) locally. Local fitting is done using least squares, and as a result, we get the normal of the plane and fitted height at point (x, y). We filter once with a local radius R of 0.3 m and once with 2.5 m, which is roughly the robot's footprint radius. Filtering result with large R is shown in Fig. 5.2. The idea behind this is to use the filtered height for computing base poses. Terrain discontinuities (e.g., steps) should not be reflected in the base movement since the base moves above the terrain. On the other hand, smoothed terrain (shown in yellow) is a good approximation for base movement, assuming that it stays at some (roughly) constant height above the smoothed terrain.

5.3.2 Online Planning

Having computed the roadmap shown in Fig. 5.3, an RRT planner finds a plan between base poses $\boldsymbol{p}_{B,start}$ and $\boldsymbol{p}_{B,goal}$. We use RRT framework to enable re-planning in potentially changing maps. Similar to previous work Tonneau et al., 2018a, our planner proposes a base pose before computing the limb contacts. Subsequently, we check whether contacts can be established and whether the robot is stable. In general, a sampling-based planner typically

has three main components: sampling, connecting a new sample to the tree, and feasibility checking. In this section, we describe how each step works.

5.3.2.1 Sampling

Candidate base poses are sampled in $SE(2)$ space instead of full-fledged $SE(3)$. The idea behind this decision is straightforward: since limbs interact with the environment, the sampler uses terrain information to constrain some DoFs of the base pose, which reduces the dimension of the search space. Hence our planner samples x, y position of the base, and yaw angle γ from a uniform distribution. The remaining DoFs are computed based on local terrain features: roll angle α, pitch angle β, and z coordinate. Roll and pitch are computed from terrain normal $\boldsymbol{n}$ such that the base remains roughly parallel to the terrain underneath. Finally the z coordinate can be computed as $z = h(x, y) + h_{desired}$ where $h(x, y)$ is the terrain elevation at sampled point (x, y) and $h_{desired}$ is user defined desired height above the terrain.

Selecting $h(x, y)$ and $\boldsymbol{n}$ can be tricky. E.g., when crossing a deep gap, terrain height can be so low that the planner cannot generate any valid poses (despite the heavy filtering). Luckily, one can leverage a simple observation to chose good h and $\boldsymbol{n}$. When moving over untraversable terrain $\neg\mathcal{T}$, the robot only cares about the nearest $\mathcal{T}$ (traversable area). The rationale is that contacts should only be made with $\mathcal{T}$, and the base pose should be selected such that limbs can reach the nearest $\mathcal{T}$. Alg. 3 implements this proposition; it selects h and $\boldsymbol{n}$ such that contacts with $\mathcal{T}$ can be established. The Alg. 3 enumerates all candidate normals and heights (lines 1-6) and then does a small brute force search to select a pair $(h_{best}, \boldsymbol{n}_{best})$ that minimizes some criterion. The *computePoseCost* function in line 9, gives low cost to poses where all legs are grounded and penalizes big roll and pitch angles. Full base pose is then determined from $(h_{best}, \boldsymbol{n}_{best})$.

5.3.2.2 Connection to Tree

Upon drawing a random base pose (x, y, γ), the planner tries to connect it to the tree (using the weighted cost of euclidean distance and angular distance). The connection is done in $SE(2)$ space using Reeds-Shepp (RS) curves Reeds and Shepp, 1990. RS curves give an optimal path between two poses while respecting the minimum turning radius constraint. For a robot that can turn in place, one could use a very small turning radius. Attempting straight-line connections between base poses would make the subsequent refinement step (see Sec. 5.4) very hard since the satisfaction of the non-holonomic rolling

Algorithm 3 Select $\boldsymbol{n}$ and h at position (x, y)

▷ Input: base position (x, y), grid map
$(\hat{x}, \hat{y})$ = nearestTraversablePosition(x, y)
h = height(x, y), $\boldsymbol{n}$ = normal(x, y)
h_f = heightFiltered(x, y), $\boldsymbol{n}_f$ = normalFiltered(x, y)
$\hat{h}$ = height$(\hat{x}, \hat{y})$, $\hat{\boldsymbol{n}}$ = normal$(\hat{x}, \hat{y})$
$\hat{h}_f$ = heightFiltered$(\hat{x}, \hat{y})$, $\hat{\boldsymbol{n}}_f$ = normalFiltered$(\hat{x}, \hat{y})$
Hs = { h, h_f, $\hat{h}$, $\hat{h}_f$}, Ns = { $\boldsymbol{n}$, $\boldsymbol{n}_f$, $\hat{\boldsymbol{n}}$, $\hat{\boldsymbol{n}}_f$}
for $(\boldsymbol{n}_c, h_c)$ in $\{Ns \times Hs\}$ **do**
    $cost$ = computePoseCost$(h_c, \boldsymbol{n}_c)$
    if $cost < c_{best}$ **then**
        $h_{best} = h_c$, $\boldsymbol{n}_{best} = \boldsymbol{n}_c$, $c_{best} = cost$
    end if
end for
return $(h_{best}, \boldsymbol{n}_{best})$

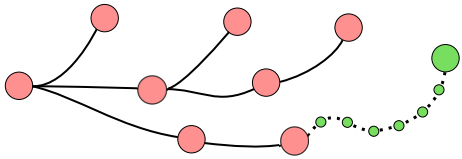


Figure 5.4: Addition of a new node into the RRT tree. Red circles and full black lines are nodes and paths that make the current RRT tree. The newly sampled node (green) has to be connected to the rest of the tree. For a successful connection, all subnodes on the connecting path (small green circles) have to be valid.

constraint cannot be guaranteed. By using RS curves, this is implicitly ensured; however, the resulting trajectory might be longer. Computing the RS connection is done terrain agnostic completely.

5.3.2.3 Feasibility Checking

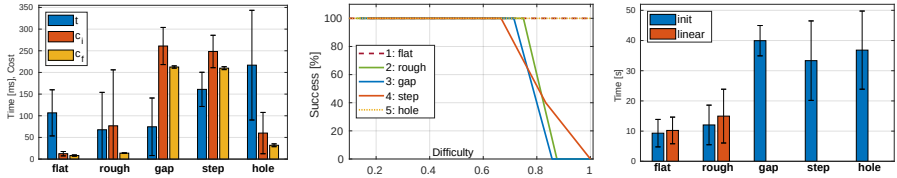
Fig. 5.4 depicts the feasibility checking. Upon drawing a new sample (large green node) as described in step 1, we compute a RS connection (dotted line) to the new node, as described in step 2. Subsequently, the dotted line is discretized into subnodes (small green circles) using an RS interpolation method Sucan, Moll, and Kavraki, 2012. Next, for each subnode we generate full 6 DoF pose using Alg. 3. Finally, each subnode undergoes feasibility checking, ensuring that the robot is statically stable and can establish enough contacts with the ground. In case a feasibility check passes for every subnode, the RRT adds the new state and the connecting path to the tree. It is crucial to

Algorithm 4 Check feasibility of base pose $\boldsymbol{T}$

```
1: (α, β) = rollAndPitch(T)
2: if |α| > α_max or |β| > β_max then
3:     return FALSE
4: end if
5: nContacts = selectContactLegsConfiguration(T)
6: switch nContacts do
7:     case 4
8:         return TRUE
9:     case 3
10:        selectSwingLegsCofiguration(T)
11:        selectSwingArmCofiguration(T)
12:        return isStable()
13:    case 2
14:        selectContactArmConfiguration(T)
15:        selectSwingLegsCofiguration(T)
16:        return isStable()
17:    case 1, 0
18:        return FALSE
```

discretize the RS path with high resolution since straight-line connections are assumed between two subsequent subnodes; in our implementation, we allow for a maximal distance of 20 cm. The length of the whole path is a tuning parameter (in our case, 15 m). Alg. 4 summarizes feasibility checking.

The feasibility check shown in Alg. 4 does not allow poses with a large roll or pitch angle (line 2). We then ground all the legs (line 5). A leg is grounded if a sufficient number of configurations in PRM are in contact with the surface and the contact location lies in $\mathcal{T}$ (see Sec. 5.2). Among grounded legs configurations the algorithm picks the one closest to the default configuration. In case of four contact legs, the pose is deemed to be stable. In case three legs are in contact, the algorithm selects good joint configuration for the swing leg according to some criterion (e.g. ground clearance or proximity to the default configuration). The swing arm configuration is selected such that the CoM is as centralized as possible. In case only two legs are in contact, the algorithm checks whether the arm can be grounded (line 14) and then proceeds with selecting swing leg configurations. The *isStable()* function computes the CoM of the whole robot and verifies that it lies in the support polygon. Aside from performing feasibility checking, Alg. 4 computes the full joint state of the



(a) Costs, Time until first solution **(b)** Success rates **(c)** Optimization times

Figure 5.5: Evaluation metrics ***Left:*** Costs and computation times of the RRT (optimization excluded). t-time until first solution (blue), c_i-initial solution cost (red), c_f-final solution cost (yellow). ***Middle:*** Success rates for different terrains versus difficulty. Maximal difficulty of 1, corresponds to roughness of ± 2 m, gap width of 3.5 m and step height of 1.5 m. Minimal difficulty corresponds to roughness of ± 0.25 m, gap width of 1 m and step height of 0.5 m. HEAP wheel radius is 0.6 m for comaprison. For the *hole* terrain and flat terrain, the difficulty remains unchanged and the curves are shown for the sake of completeness. ***Right*** Computation time until optimization convergence when initializing using planner's first stage versus using linear interpolation. For terrains with obstacles, *gap*, *step* and *hole*, linear interpolation cannot find a solution.

robot $\boldsymbol{q}$ for feasible poses. The CoM of the full joint configuration $\boldsymbol{q}$ can be computed using:

$$\boldsymbol{p}_{com}(\boldsymbol{q}) = \frac{1}{M}\left(m_B \boldsymbol{p}_{B,com} + \sum_{i=1}^{N} m_i \boldsymbol{p}_{i,com}(\boldsymbol{q}_i)\right) \tag{5.3}$$

where M is the mass of the whole robot, $\boldsymbol{p}_{i,com}$ and m_i are the position of CoM and mass of the i^{th} limb, respectively. Thanks to the mapping between $\boldsymbol{q}_i$ and $\boldsymbol{p}_{i,com}$ computed offline, the sum in Eq. 5.3 can be evaluated rapidly. Note that unlike Tonneau et al., 2018a, we do not require all limbs to contact the environment while generating base poses, thus allowing for more flexibility. Once the RRT has reached $\boldsymbol{p}_{B,goal}$, the final path is post-processed. In the first step, the contact schedule is modified to ensure stability. We do not allow establishing/breaking more than one contact between two different successive nodes. If the robot wants to change more than one contact state at any point, we insert a node in between. Those situations happen only when the arm contact is established/broken. The robot tries to change the contact state of the arm and leg(s) simultaneously. We add a short full contact phase (legs + the arm) in between to ensure static stability. Secondly, we compute Inverse Kinematics (IK) for the non-wheeled limbs in contact. Any of those limbs has to satisfy the contact constraint $\dot{\boldsymbol{p}}_i = \boldsymbol{0}$. For each non-wheeled limb in contact,

we find base poses at the beginning and the end of its respective contact phase. The reference position $\boldsymbol{p}^*$ for the IK is found by solving:

$$\min ||\boldsymbol{p}_s - \boldsymbol{p}_e|| \tag{5.4}$$

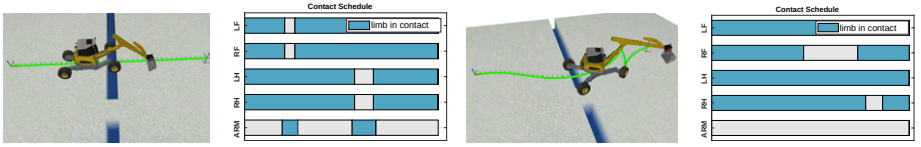
and setting $\boldsymbol{p}^* = (\boldsymbol{p}_s^* + \boldsymbol{p}_e^*)/2.0$, where $\boldsymbol{p}_s$ are all positions in the i^{th} limb roadmap at the beginning of the contact phase and $\boldsymbol{p}_e$ at the end of the contact phase. We then compute i^{th} limb's joint angles for every contact node as $\boldsymbol{q}_i = IK(\boldsymbol{p}^*)$.

Alg. 4 ensures that the robot is stable and that limbs are not in a collision. Nevertheless, it assumes that straight line connections in joint space are collision-free. The assumption might be invalid, especially for the arm, which moves around the base and is used as a counterweight. To overcome this problem, we use the precomputed roadmap in which we, similar to Short and Bandyopadhyay, 2017, invalidate all vertices and edges that are in a collision with other limbs or the environment. Once the graph is updated, each limb's path is found using a graph search algorithm (A* in our case). In practice, legs always end up being collision-free, but the arm often collides with legs. Hence we run the graph search only between nodes where the arm moves.

5.4 Refinement Step

The refinement step uses TO to satisfy all system constraints. TO methods scale well with system dimension and can handle nonlinear constraints such as forward kinematics or non-holonomic rolling constraints. However, computing the correct contact schedule and dealing with obstacles remains challenging for the gradient-based methods, so we initialize optimization with trajectory computed in the *initialization step*. The optimization receives contact schedule, base position/velocity (6 DoF) and joint position/velocity (25 DoF in our case) and solves a feasibility problem. Adding an optimization objective allows for fine motion tuning, however it typically results in increased computation times. TO planner used in this paper is based upon our previous work Jelavic and Hutter, 2019, and below we present modifications that enable us to cope with more challenging scenarios.

Terrain maps To the best of the author's knowledge, there is no optimization-based planner for hybrid systems that can handle general terrain representations. Compared to their legged counterparts, hybrid robots keep their limbs in contact over long distances. Hence, map errors influence more variables



(a) Prefer Shortest Path **(b)** Contact Schedule Stepping **(c)** Prefer Driving **(d)** Contact Schedule Driving

Figure 5.6: Traversing terrain in different manners. Limb names: *LF* stands for Left Front, whereas *RH* stands for Right Hind etc. ***Left:*** HEAP reaches goal pose via shortest path ***Middle Left:*** Contact schedule traversing over the gap. Total duration: 114.98 s ***Middle Right:*** HEAP trying to minimize stepping while reaching the goal. Not that two legs still have to be lifted since the bridge is too narrow for both legs. ***Right:*** Contact schedule for driving whenever possible. Total duration: 156.33 s

and constraints, which makes the optimization more sensitive. So far, proposed terrain-aware optimization planners have used analytical descriptions of the environment Medeiros et al., 2020, Winkler et al., 2018, Sun et al., 2020. Our planner integrates grid maps Fankhauser and Hutter, 2016b into the optimization, thus allowing planning in any environment where a 2.5D map is a suitable representation. The elevation map comes into the optimization in the form of height constraint for all limbs in contact $\mathcal{C}$.

$$\boldsymbol{p}_i^z = h(\boldsymbol{p}_i^x, \boldsymbol{p}_i^y), \ \forall i \in \mathcal{C} \tag{5.5}$$

The constraint (5.5) is problematic since height mapping $h(\cdot, \cdot)$ is discrete and discontinuous (e.g. gaps or steps) which can cause optimization to diverge. Far away from the cell center, Nearest Neighbor (NN) search or linear interpolation are poor approximations of the true elevation. Hence, approximation of partial derivatives $[\frac{\partial h}{\partial x} \ \frac{\partial h}{\partial y}]$ using central finite differences renders them non smooth. For large grid cells (0.1 m), the key to improving the optimization convergence is using a higher-order approximation of the $h(\cdot, \cdot)$ function. We found that bicubic interpolation and bicubic convolution algorithms Keys, 1981 work well. Implementations of both algorithms are integrated into the open-source package *Grid Map* and made available for the community[1]. Unfortunately, neither filtering nor higher-order approximation can help if the terrain is discontinuous (steps or gaps). To handle discontinuities, we use *gradient clipping*, a technique known from the machine learning community. Smaller clipping thresholds prevent getting stuck in bad minima. However, they usually require a few more iterations for convergence.

[1] `https://github.com/ANYbotics/grid_map`

Traversability Constraint We require that all contacts stay in the traversable area; the constraint implements Eq. 5.2. The gradient is also computed using a central finite difference with bicubic interpolation.

Collision Avoidance Constraint imposes a minimum distance between the collision geometries of the robot. We use shape primitives such as spheres or cylinders. We impose that the minimal distance between collision geometries has to be greater than d_{min}. The minimal distance between collision geometries is calculated using the *Bullet* physics engine.

5.5 Results

We tested our planner on HEAP *HEAP (Hydraulic Excavator for an Autonomous Purpose)* 2018, which is a customized Menzi Muck M545 walking excavator; five limbs, 25 joints, and a floating base make it a challenging test bench. The whole planning pipeline is implemented in C++ programming language, and tests are performed on the Intel Xeon E3-1535M processor with 32 GB of RAM.

5.5.0.1 Roadmap Generation

We use a roadmap size of 300 vertices per leg and about 3000 vertices for the arm since it has a much larger workspace (see Fig. 5.3). More vertices yield better workspace approximation; consequently, finding stable configurations is more likely. However, with more vertices, more computation is required to find stable configurations. We found the proposed number of vertices to be enough for finding solutions for the scenarios tested. For each vertex in the roadmap, we attempt the connection to its ten nearest neighbors (in operational space). If the distance to the nearest neighbor is bigger than d_{max}, the connection is rejected. For the legs d_{max}=0.3 m and for the arm d_{max} =1 m. PRM generation takes about 30 minutes, with more than 99 % of the computation required for the arm roadmap generation. Looking for connections that are shorter than d_{max} is the most computationally intensive operation.

5.5.0.2 Terrains

The proposed planning pipeline can compute plans in various terrains (see Fig. 5.1, also see the video attached[2]). The planner uses the same set of parameters for all scenarios. Traversing such challenging terrains would not be

[2] `https://youtu.be/B-NHY4xwgwY`

possible with a purely optimization-based planner and handcrafting a contact schedule for those scenarios would be difficult. We show success rates on complex terrain features in Fig. 5.5b, by averaging five trials for each difficulty. All terrains except flat ground are shown in Fig. 5.1: *rough* (middle left image), *gap* (bottom right), *step* (middle right), *hole* (top right). For terrains *flat*, *rough*, *gap* and *step*, the planner was asked to find a path of about 15 m in length. For terrain *hole*, the length was about 30 m such that the planner has to navigate around the hole to get to the other side (see Fig. 5.1, top right). The maximal distance parameter in the RRT was set to 10 m. We consider planning successful if both initialization and refinement step find a solution. Times and costs in Fig. 5.5a have been obtained by averaging ten successful trials across all terrain difficulties. All the plans are found using the same RRT optimizing planner with planning time of 4 s. Fig. 5.5a shows that the first stage finds initial solutions quickly and that they are close to the optimal ones. Short planning times suggest that the *initialization step* could be used in a receding horizon fashion.

5.5.0.3 Importance of Initialization

To quantify effect of good initialization on the convergence, we use linear interpolation (between start and goal pose) as a baseline strategy. The other strategy is using a whole-body plan from the first stage to initialize all the variables. Linear interpolation uses the contact schedule from the first stage since it cannot compute one alone. The computation times until convergence when initialized with linear interpolation are shown in Fig. 5.5c (red color). The times were obtained by averaging ten successful trials with different initializations over all terrains. The computation times variance is caused by different durations of initialization trajectories from the first stage (longer durations require more computation). A good initialization makes a small difference in continuous terrain (*flat* terrain and *rough* terrain with roughness ± 0.5). However, it becomes essential for harder terrains (*hole*, *gap*, *step*) since linear interpolation fails to produce a solution. This result corroborates our hypothesis from the Sec. 5.1 that TO can easily handle non-holonomic and nonlinear constraints from the robot model, whereas terrain constraints and contact schedule discovery present challenges for the optimization. Unlike agile quadrupeds in Winkler et al., 2018, HEAP cannot execute full flight phases, which makes the terrain constraints especially challenging.

5.5.0.4 Contact Schedule Discovery

The way of terrain traversing can be influenced by tuning the cost function inside SBP. This is illustrated in Fig. 5.6. HEAP is commanded to reach the other side of the gap. Trajectory of the base is shown in green color. In the first scenario (Fig. 5.6a), HEAP incurs no cost for lifting the legs off. Upon introducing the stepping penalty, HEAP realizes that it can use a small bridge to avoid breaking contact on all legs, Fig. 5.6c. This behavior emerges merely by introducing the stepping penalty and without any other modifications. Such flexibility is made possible by optimizing over contact schedule and full-body poses simultaneously. The resulting contact schedules are shown in Fig. 5.6b and Fig. 5.6d.

5.6 Conclusion and Outlook

We present a combined sampling and TO based planner for legged-wheeled robots. The sampling-based stage computes whole body configurations and contact schedule; it is based on RRT planner and a roadmap that is pre-computed offline. Compared to existing work, the roadmap is extended to store the mapping between joint angles and CoM, which allows for quick stability checks in the presence of heavy limbs. Our SBP planner achieves fast planning times and could be used interactively. In the second planning stage, TO satisfies all system constraints, such as non-holonomic rolling constraint. We integrate elevation maps into TO and demonstrate planning on general map representations. Evaluations of the proposed approach suggest that the main difficulty for TO stems from terrain/collision (avoidance) constraints and contact schedule planning, problems that are mitigated using the proposed two-stage approach.

Future developments will verify the approach on real machines. Besides, we plan to investigate using TO for tracking plans from the first stage in an MPC fashion.

5.7 Lessons Learned

This publication investigated combining the merits of sampling and optimization-based planning. We use a two-stage approach where the SBP computes an initial guess for the optimization-based planner from Paper II. The computationally intensive foothold computation and stability checking

inside SBP is sped up by utilizing roadmaps computed offline. General purpose elevation maps are integrated into the optimization-based planner from Paper II. The success rate of the proposed two-stage planner was evaluated on a variety of challenging terrains.

The most important takeaway from this publication is that optimization needs to be carefully initialized in the presence of obstacles or difficult terrain. We quantitatively evaluated how important the initial guess is. We found that it is not enough to provide a feasible Contact Schedule (CS), but it is also essential to initialize the motion and the footholds. The results confirmed our intuition that the optimization gets stuck in local minima for difficult terrains.

On the other hand, on easy terrain, the TO had no issue computing motion plans that required dealing with nonlinear kinematics—in this case, providing a good initial guess slightly improved the convergence (fewer required iterations). These facts suggest that planning in the presence of obstacles is more challenging than planning with complex kinematics/dynamics.

Lastly, naively including elevation maps in the optimization can lead to a loss of convergence. Typically the maps contain discontinuities either because of sensor noise or terrain features (e.g., steps). Naively computing gradients using finite differences can lead to vast and discontinuous gradients. We found that *gradient clipping* can help retain convergence even with large terrain jumps and sensor noise. Furthermore, it is helpful to use interpolation between the samples for large grid map cells. Higher-order interpolations like cubic or cubic convolution improve convergence compared to linear interpolation.

To aid the foothold planning inside the TO, we included a 2D signed distance field computed from the traversability map. We found 2D SDF to be an excellent representation for foothold planning since it has continuous gradients, it is fast to compute, and it avoids dedicated pipelines for foothold feasibly region computation like in Grandia et al., 2022.

6

LSTP: Long Short-Term Motion Planning for Legged and Legged-Wheeled Systems

Jelavic, E., Qu, K., Farshidian, F., and Hutter, M. (2022c). "LSTP: Long Short-Term Motion Planning for Legged and Legged-Wheeled Systems". *submitted to IEEE Transactions on Robotics*

DOI: not published yet

Video: https://youtu.be/iQiNAy6sLlo

This article presents a hybrid motion planning and control approach applicable to various ground robot types and morphologies. Our two-step approach uses a sampling-based planner to compute an approximate motion which is then fed to numerical optimization for refinement. The sampling-based stage finds a long-term global plan consisting of a contact schedule and sequence of keyframes, i.e., stable whole-body configurations. Subsequently, the optimization refines the solution with a short-term planning horizon to satisfy all nonlinear dynamics constraints. The proposed hybrid planner can compute plans for scenarios that would be difficult for trajectory optimization or sampling planner alone. We present tasks of traversing challenging terrain that requires discovering a contact schedule, navigating non-convex obstacles, and coordinating many degrees of freedom. Our hybrid planner has been applied to three different robots: a quadruped, a wheeled quadruped, and a legged excavator. We validate our hybrid locomotion planner in the real world and simulation, generating behaviors we could not achieve with previous methods.

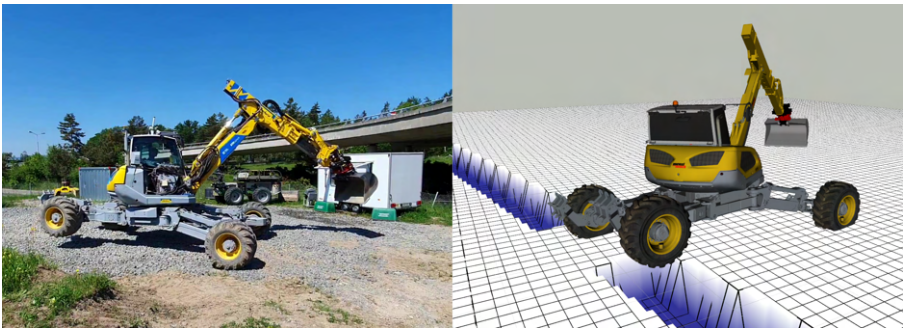
The results show that computing and executing hybrid locomotion plans is possible on hardware in real-time.

6.1 Introduction

Motion planning is one of the fundamental problems in robotics, as it enables mobile robots to safely navigate in uncontrolled environments. Among different strategies for mobility, legs combined with wheels are the most promising solution when it comes to task that require a high level of mobility (legs) and efficiency (wheels) in challenging terrain Bjelonic et al., 2019a; Jud et al., 2021b; Kashiri et al., 2019.

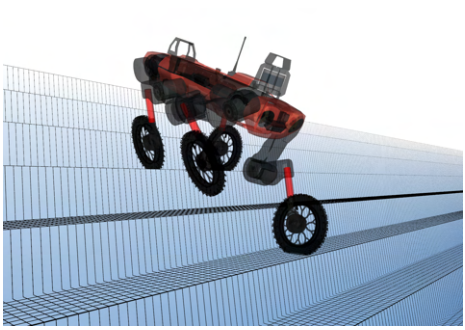
The motion planning for hybrid platforms, i.e. systems with legs and wheels, is particularly challenging since, on one side, legs and wheels increase the number of Degrees of Freedoms (DoFs) the planner has to handle. On the other hand, motion planning for legged-wheeled robots is particularly complex since the combinatorial nature of stepping with legs (contact schedule) is combined with the non-holonomic nature of wheeled robots (rolling constraints). To tackle this challenging problem, we propose a motion planning framework for legged and legged-wheeled platforms based on different levels of model fidelity and different prediction horizons. In particular, a Numerical Optimization (NO) technique is used for short-term planning with high fidelity (e.g. full kinodynamic model) and randomized sampling for long-term planning with lower fidelity (full kinematics in quasistatic conditions).

NO is frequently used for planning and control with legged robots as it handles well high dimensional joint space, non-holonomic constraints, and nonlinear dynamics constraints. Examples can be found in the early work of Bellicoso et al., 2016; Bjelonic et al., 2019a; Jenelten et al., 2020, which follows the decoupled planning approach for the footholds and the base of the robot. With the advance of solvers and more computing power, researchers have started solving whole body planning problems in a receding horizon fashion fully onboard Grandia et al., 2022, whereby discrete foothold locations and continuous body motions are jointly optimized. However, a major limitation is the sensitivity to initialization and local optima, which are often caused by terrain constraints (e.g., collision avoidance) Jelavic and Hutter, 2019. Similarly, researchers tackled the locomotion problem using Reinforcement Learning (RL) methods Lee et al., 2020; Miki et al., 2022a. Despite the unprecedented robustness and impressive field deployments, RL methods still struggle to achieve



(a) Legged excavator

(b) Quadruped



(c) Wheeled quadruped

Figure 6.1: ***a)*** HEAP overcomes a virtual bridge too narrow to drive over it. Hence, the stepping behavior emerges. We did not dig a ditch in our testing field. However, the map supplied to the planner contains it. *Left:* HEAP driving on three wheels over the bridge. *Right:* Visualization of the map the planner sees. ***b)*** ANYmal climbing consecutive steps. ***c)*** ANYmal on wheels stepping up the stairs (plan visualization). All the motions are also shown in the accompanying video `https://youtu.be/4eqq4Ucl92o`.

precise and coordinated foot placement required to negotiate terrains such as stepping stones Brakel et al., 2021; Grandia et al., 2022.

On the other hand, Sampling Based Plannings (SBPs) Karaman and Frazzoli, 2011; Kavraki et al., 1996; LaValle and Kuffner Jr, 2001; LaValle et al., 1998 can handle very non-convex environments since the randomization allows them to escape local minima. The robotic community has been using SBP for planning contacts and whole-body motions on a variety of legged robots Bretl, 2006; Geisert et al., 2019; Hauser et al., 2008; Tonneau et al., 2018a. However, including kinodynamic constraints in a sampling-based planner remains an open research problem. Legged robots have many DoFs which often results

in long planning times as SBP runtimes scale exponentially with configuration space dimension.

This work presents Long Short-Term Motion Planner (LSTP) combining the merit of optimization-based and sampling-based methods while generalizing to different robot types. The method is showcased on a quadruped, a wheeled-quadruped, and a legged excavator. The SBP computes contact schedule and keyframe (whole-body state) sequence quickly ($< 1\,\mathrm{s}$) thanks to offline-computed roadmaps. Subsequently, the NO refines the initial solution. Whole-body states from SBP act as attractors for the optimization preventing it from falling into bad local minima. Thus, we can handle both complex terrains (thanks to SBP), many DoFs, and complex system dynamics (thanks to NO). We run the NO in an Model Predictive Control (MPC) fashion, thus ensuring stability and robustness to disturbances Farshidian et al., 2017a.

6.1.1 Related Work

The robotic community has extensively studied motion planning for legged robots, and NO emerged as one of the most common methods, especially for short-term planning. Examples of optimization-based controllers for quadrupeds can be found in Bellicoso et al., 2016, 2018b where Hierarchical Optimization (HO) satisfies system constraints while tracking base twist reference. Optimization has also been used to implement active suspension behaviour on legged-wheeled robots Cordes, Babu, and Kirchner, 2017; Giordano et al., 2009; Reid et al., 2016. However, these works do not have a look-ahead horizon in the future and hence cannot overcome challenging obstacles.

Planning allows the robot to prepare for the oncoming obstacles and thus increases the chances of overcoming them. In Winkler et al., 2018, the authors use collocation to transcribe the problem into an Nonlinear Program (NLP) and solve it using IPOPT solver Wächter and Biegler, 2006. The optimization discovers whole-body maneuvers using all DoFs to overcome various obstacles. This approach has been extended to work with a legged-wheeled robot Medeiros et al., 2020 and a legged excavator Jelavic and Hutter, 2019. Similar examples of using NO for planning can be found in Geilinger et al., 2018; Jelavic et al., 2020; Melon et al., 2020. The authors above plan offline once and then track the plan using a tracking controller. In Jelavic et al., 2020; Winkler et al., 2018, the tracking controller is based on HO from Bellicoso et al., 2016.

Keeping the contact schedule fixed and allowing the NO to optimize for the whole-body motion is a common practice in the legged-robot community.

While NO can also optimize over the contact schedule Jelavic and Hutter, 2019; Winkler et al., 2018, it often results in a very non-convex NLP that is difficult to solve. This motivates the use of Mixed-Integer Programs (MIPs) and L1 norms for planning Ioan et al., 2021; Tonneau et al., 2020. MIPs have been successfully applied in robotics, e.g., for humanoid robots Deits and Tedrake, 2014. While this is an attractive problem formulation, the computation times scale exponentially with the planning horizon.

Planning in the receding horizon fashion reduces the need for an accurate tracking controller and increases the system's robustness. Many roboticists use NO in this fashion, and this paradigm is often called MPC. In Bellicoso et al., 2018b; Bjelonic et al., 2019a; Jenelten et al., 2020; Winkler et al., 2015 the authors generate motions in a decoupled fashion (decoupled base trajectory and footholds) using Zero Moment Point (ZMP) criterion. The decoupled planning has also been applied to legged-wheeled robots Bjelonic et al., 2019a; Sun et al., 2020. The optimization problem can be solved at high frequencies, up to 200 Hz. However, decoupling the problem reduces the controller's generality.

Increased computing power and new methods based on differential dynamic programming Farshidian et al., 2017c have enabled optimizing for base and limbs simultaneously Farshidian et al., 2017a. Many recent works solve a full optimization problem using a centroidal or single-rigid-body model Grandia et al., 2019b, 2022; Ishihara, Itoh, and Morimoto, 2019; Jenelten et al., 2022; Sleiman et al., 2021. Algorithm in Li, Frei, and Wensing, 2021 solves a hierarchical MPC, where a high-fidelity model (near future predictions) is combined with a lower-fidelity model (predictions further away in time). Optimizing over full dynamics without decoupling makes planned motions stabler Li, Frei, and Wensing, 2021. Moreover, the MPC has progressed to the point where it is possible to incorporate the terrain constraints and solve the optimization in real time Grandia et al., 2021; Melon et al., 2021. Bjelonic et al., 2022 shows a two-stage optimization where a motion library is first computed offline and then executed using online MPC. We also follow the approach of using perceptive MPC as a backbone for our refinement planning stage.

While NO is a great tool for dealing with complicated nonlinear dynamics and many DoFs, it inevitably falls prey to local minima. Some researchers have tackled this problem by employing stochastic optimization Mastalli et al., 2020. While the planner can cope with challenging terrain, the motion plans take up to several minutes to compute. Another stream of research uses different types of planners to cope with local minima; most of them are based

on Rapidly-Exploring Random Trees (RRTs) Karaman and Frazzoli, 2011; LaValle and Kuffner Jr, 2001, Probabilistic Roadmaps (PRMs) Kavraki et al., 1996 or grid-based methods (e.g. A* Hart, Nilsson, and Raphael, 1968).

Grid-based methods discretize the workspace and use a graph search algorithm (A* being a popular choice). Early works Escande, Kheddar, and Miossec, 2013 can generate complex motions by defining a suitable potential field over the workspace. However, motions like crawling narrow passages and ladder climbing come at the cost of high computation times (up to 3 h). More recently, Klamt and Behnke, 2017, 2018 propose motion planning for the legged-wheeled robot Momaro based on A* that runs in real-time by combing handcrafted stepping motions with a carefully designed cost function.

Early works utilizing SBPs generate complex maneuvers, such as free climbing or ladder climbing, yet with very long computation time (e.g., 16 min for 32 steps) Bretl, 2006; Escande, Kheddar, and Miossec, 2013; Hauser et al., 2008. Hornung et al., 2014 shows RRT-based whole-whole body motion planning with a Nao humanoid. The work builds upon ideas presented in (Kuffner et al., 2002; Şucan and Chitta, 2012), where samples are drawn from precomputed sets. While gracefully handling whole-body manipulation on flat ground, the approach is unsuitable for traversing obstacles since the set of stable configurations is strongly dependent on the terrain. Recently, more efficient approaches are proposed for humanoid and quadrupedal robots Geisert et al., 2019; Short and Bandyopadhyay, 2017; Tonneau et al., 2018a,b. SBPs can deal with very complex environments, and even compute limb contact schedule Tonneau et al., 2018a. However, creating kinodynamically consistent motion directly in the SBP remains challenging, possibly leading to complex planning pipelines and long planning times.

Replacing the SBP with an RL policy can be an effective alternative for foothold planning. In Tsounis et al., 2020, RL is used to plan footsteps which are then tracked by the RL based controller. Gangapurwala et al., 2022 also uses an RL footstep planner with a model-based tracking controller executing the plan. Li et al., 2021b uses a similar strategy except that the model-based tracking is happening in the latent space. While promising, these methods still require retraining for different types of terrains and cannot plan acyclic gait patterns.

Global planning typically relies on a sampling-based or grid-search technique with reduced model order. The resulting high-level planner is then combined with a local planner or controller Klamt and Behnke, 2017; Wermelinger et

al., 2016 that are often based on cyclic gait pattern. Tranzatto et al., 2022b; Wellhausen and Hutter, 2021 extends the SBP with neural network to predict the terrain difficulty for the underlying tracking controller. The introduced cost estimator requires re-training of the neural network estimator in case of a controller swap.

Combining sampling and optimization-based planning is a sound strategy to escape the local minima and enforce kinodynamic constraints. One possible combination is to solve a Boundary Value Problem (BVP) for computing connections inside the SBP. BVP can typically be solved either for low DoF systems (e.g., a vehicle) or with an analytical solution (e.g. Ding et al., 2021; Hwan Jeon, Karaman, and Frazzoli, 2011; Kim, Kwon, and Yoon, 2018; Xie et al., 2015). Fernbach et al., 2020; Fernbach, Tonneau, and Taïx, 2018 computes transition feasibility for a humanoid robot and successfully transfers the solution to hardware. Another possibility is using sampling or grid-based search to provide an initial guess for the optimization and refine the motion plan. The two-stage approach has been widely used in the literature, most notably for manipulation problems Dai et al., 2018; Leu, Wang, and Tomizuka, 2022; Leu et al., 2021. In Li et al., 2021a, motion plans for a trailer composition are calculated, and in Li, Long, and Gennert, 2016, the authors plan whole-body motions for a humanoid robot without planning a contact schedule. We followed the idea of using the two-stage approach, and we extended it to be used for whole-body motion and contact planning.

This work computes approximate whole-body plans with an SBP and then refines them with MPC. We follow the idea of using contact dynamic roadmaps introduced in Short and Bandyopadhyay, 2017 to speed up the SBP which are an extension of dynamic roadmaps Leven and Hutchinson, 2002 for legged robot usage. We further extend the contact roadmap to handle robots with heavy limbs. Our planner leverages MPC from Farshidian et al., 2017a for tracking and re-planning on a shorter time horizon. We propose a strategy for using kinematic MPC on legged excavators operating in challenging terrain where terrain adaptation is imperative, but precise torque control is often unavailable.

6.1.2 Contribution

This work presents LSTP, a combined sampling and optimization-based planner for mobile robots with many DoFs. The planner consists of a long horizon *Initialization Step* with SBP computing a contact schedule and a sequence of keyframe whole-body configurations and a short horizon *Refinement Step* with

MPC satisfying dynamics and contact constraints. We introduce the following contributions:

- We extend the dynamic roadmap data structure to be applicable for robots with heavy limbs, such as legged excavators. Compared to Short and Bandyopadhyay, 2017, we achieve faster roadmap lookup. This allows our SBP planner to rapidly find plans under 0.5 s over challenging terrains such as stepping stones.
- The proposed LSTP generalizes to a variety of different legged mobile platforms. We exemplarily demonstrate this on a quadruped, a wheeled quadruped with non-steerable wheels, and a legged excavator with steerable wheels and an arm. So far, only generalizations for different legged robots have been shown in the Tonneau et al., 2018a
- We present and apply for the first time a whole body MPC controller to a legged excavator, enabling it to execute motions on par with skilled human operators. We retain all the qualities from the controller presented in Jelavic et al., 2020 and extend it to execute new motions
- We extensively verified the proposed planner in simulation and experimentally on three different robots, including a quadruped (ANYmal), legged-wheeled quadruped (ANYmal on wheels), and a legged excavator (Hydraulic Excavator for Autonomous Purpose (HEAP)).

With respect to our previous work Jelavic, Farshidian, and Hutter, 2021, we generalize the SBP planner for different robots, show hardware experiments on multiple platforms and design a whole-body control system for HEAP. Lastly, we remove the need for offline optimization-based refinement and introduce a real-time MPC.

6.2 Preliminaries

Notations used for robot's limbs are the following: *LF* stands for Left Front, and similarly we have, *RF*, *LH*, *RH* (Right Hind). *ARM* denotes HEAPs arm. Label *EE* means End-Effector. Left superscript denotes the frame (e.g. ${}^{\mathcal{B}}\boldsymbol{p}$ means a position in the base frame $\mathcal{B}$) and defaults to world frame $\mathcal{W}$ if omitted. Right superscript indicates a component of a vector (e.g. $\boldsymbol{p}^z$ is the z component of $\boldsymbol{p}$). $\boldsymbol{T}$ denotes a homogeneous transform composed of translation $\boldsymbol{t}$ and rotation $\boldsymbol{R}$.

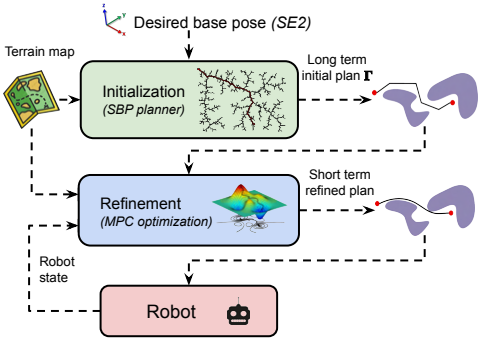


Figure 6.2: Planning pipeline overview

Our planner computes a trajectory that is T seconds long, given the starting base pose $\boldsymbol{T}_B(t=0) = \boldsymbol{T}_{B,start}$ and $\boldsymbol{T}_B(t=T) = \boldsymbol{T}_{B,goal}$ where $\boldsymbol{T}_{B,start}$ and $\boldsymbol{T}_{B,goal}$ are given poses of the robot's base. We always command the goal pose in $SE(2)$ since the height, roll, and pitch are well-defined by the terrain around the goal pose (see Sec. 6.3.3.1). No additional constraints on the joint configurations are imposed at $\boldsymbol{T}_{B,goal}$. Proposed pipeline is visualized in Fig. 6.2. The planner is divided into an *Initialization step* computing an approximate long-term plan $\boldsymbol{\Gamma}$ and a *Refinement step* computing a short-term plan satisfying all the constraints and exhibiting smooth motions.

The planner perceives the world via a multilayered map, which can be defined as mapping $f : \mathbb{R}^2 \to \mathbb{R}$ from (x, y) coordinates to various functions $f(x, y)$. It always receives elevation data $h(\cdot, \cdot)$, then computes and stores Signed Distance Function (SDF) or traversability as layers in the map, relying on the *grid_map* implementation Fankhauser and Hutter, 2016a. Terrain (map) is partitioned into *traversable* part $\mathcal{T}$ and *untraversable* part $\neg\mathcal{T}$. Any contact point needs to be in the traversable area $\mathcal{T}$:

$$(\boldsymbol{p}^x, \boldsymbol{p}^y) \in \mathcal{T} \equiv sdf_2(\boldsymbol{p}^x, \boldsymbol{p}^y) \geq \delta \geq 0, \tag{6.1}$$

where δ is a user-defined parameter and $sdf_2(\cdot, \cdot)$ represents a 2D SDF with the positive distance meaning the point lies in $\mathcal{T}$. All contact points need to be at least δ away from the $\neg\mathcal{T}$. sdf_2 is calculated using Felzenszwalb and Huttenlocher, 2012 and stored as a grid map layer.

A legged robot can in general have both limbs without wheels and limbs with wheels (e.g. HEAP has 4 legs with wheels and an arm). We say that i^{th} limb is in contact when it's close to the surface,

$$|\boldsymbol{p}^z_{ee,i} - h(\boldsymbol{p}^x_{ee,i}, \boldsymbol{p}^y_{ee,i}) - R_w| \leq \epsilon, \quad \epsilon \in \mathbb{R}_+, \tag{6.2}$$

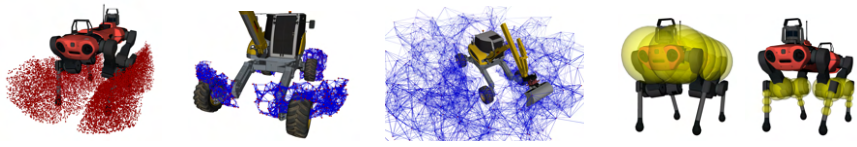
where $\boldsymbol{p}_{ee,i}$ denotes the position of i^{th} limb End-Effector (EE) and $h(\cdot,\cdot)$ gives the terrain height. R_w is the wheel's radius and is set to 0 for limbs without wheels. $\epsilon \in \mathbb{R}_+$ is used to control contact proximity. Eq. 6.2 assumes that the contact point is directly underneath the wheel's center, which is often violated. However, the tracking controller can compensate for this error (see experiment section).

6.3 Long Term Planning: Initialization Step

The *Initialization Step* computes an approximate long-term (global) solution to the planning problem. The approximate solution is a sequence of keyframe configurations with a feasible contact schedule. Each keyframe is a statically stable whole-body configuration with contact flags. Simple interpolation between keyframes can violate system constraints (e.g., rolling constraints); hence, the solution is only approximate. The *Initialization Step* uses RRTs Karaman and Frazzoli, 2011; LaValle et al., 1998 to explore the space, but one could also use PRM planners Kavraki et al., 1996. This chapter first outlines the computation of limb roadmaps that happens offline. Limb roadmaps store the mapping between joint configurations and EE positions and are used to speed up foothold computation during the planning phase. Subsequently, the chapter describes the terrain preprocessing pipeline, triggered whenever the map is updated. The last section combines all the elements in a sampling-based planner.

6.3.1 Roadmap Precomputation

A roadmap is a mapping between i^{th} limb joint angles $\boldsymbol{q}_i$ and the EE position in the base frame ${}^{\mathcal{B}}\boldsymbol{p}_{ee,i}$. Each limb has its own base-pose invariant roadmap (expressed in the base frame). Examples of roadmaps are shown in Fig. 6.3. Generating a roadmap offline and using it online is an idea introduced in Kallman and Mataric, 2004; Short and Bandyopadhyay, 2017. Here we extend the roadmap to store mapping between the i^{th} limb joint angles $\boldsymbol{q}_i$ and the position of its Center of Mass (CoM) ${}^{\mathcal{B}}\boldsymbol{p}_{com,i}$. The CoM location is needed to



(a) Roadmap ANYmal legs **(b)** Roadmap HEAP legs **(c)** Roadmap HEAP arm **(d)** Collision spheres for ANYmal base and limbs

Figure 6.3: Roadmap vertices (red spheres) and edges (blue lines). Each red dot represents an end-effector position in $\mathbb{R}^3$. ***a)*** Roadmap for ANYmal robot (only vertices shown). Each leg has 5000 vertices in the roadmap. ***b)*** HEAP legs have 300 vertices and about 2000 edges in the roadmap. ***c)*** HEAP arm with 1000 vertices and about 8000 edges shown. ***d)*** Collision spheres approximating the geometry of the legged robot. We use ten balls of a radius of 20 cm for the base, two balls of 7 cm radius for each knee, and three balls of 3.5 cm radius for each shank.

test the support polygon stability criteria for keyframes. The CoM position of a robot with total mass M and limbs $\mathcal{I}$ can be computed as:

$$^{\mathcal{B}}\boldsymbol{p}_{com} = \frac{1}{M}\Big({}^{\mathcal{B}}\boldsymbol{p}_{com,B}\, m_B + \sum_{i\in\mathcal{I}} {}^{\mathcal{B}}\boldsymbol{p}_{com,i}\, m_i \Big). \tag{6.3}$$

Evaluating Eq. 6.3 is much faster than computing the CoM from scratch, and it allows us to rapidly check the stability criterion during online planning. In Eq. 6.3, m_i denotes mass of i^{th} limb and m_B is the mass of robot's base. For robots with heavy limbs, such as HEAP or humanoid robots, evaluating Eq. 6.3 is essential to ensure stability.

Besides stability, no self-collision should occur for any given configuration of the legs. A simple way to prevent self-collision is to enforce the roadmap vertices to stay in their respective quadrants. For the HEAP arm, preventing all collisions in the roadmap generation might be too restrictive. Hence, the collision-free arm movement is planned during the online planning phase by invalidating portions of the roadmap colliding with legs or the terrain. Choosing the number of vertices in the roadmap involves a tradeoff. Many vertices approximate the workspace well but require more computation for online planning. We found that 4000-5000 vertices yielded an acceptable tradeoff for a legged robot. For HEAP, we used 300 vertices in the leg roadmap and 3000 vertices in the arm roadmap. The connection between neighboring vertices is rejected if it is longer than d_{max}. We used $d_{max} = 0.3\,\text{m}$ for HEAPs legs and $d_{max} = 1\,\text{m}$ for the arm. For the legged robot we used $d_{max} = 0.05\,\text{m}$. Space around the robot is voxelized, and we store vertices within respective voxels.

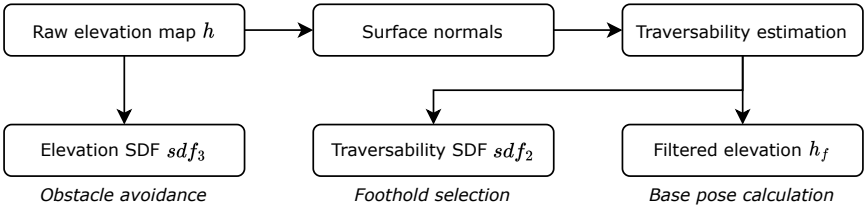


Figure 6.4: Terrain preprocessing module overview.

Voxelization accelerates configuration lookup since a good guess of foothold typically exists, e.g., around the default position. Finding candidate footholds takes $9 \pm 9\,\mu$s for a quadruped robot. We avoid transforming the terrain into the roadmap frame and search vertices only around the best guess instead of searching the entire roadmap and randomly sampling a feasible one Short and Bandyopadhyay, 2017. In implementing their method, just transforming the terrain without any foothold search takes $35 \pm 10\,\mu$s.

6.3.2 Terrain Preprocessing

The only input to the terrain preprocessing module is a raw elevation map (h) implemented using a grid map data structure Fankhauser and Hutter, 2016a. From this elevation, we compute additional layers for surface normals, traversability SDF, elevation SDF, and filtered elevation. The relations between different tasks and their usages are illustrated in Fig. 6.4.

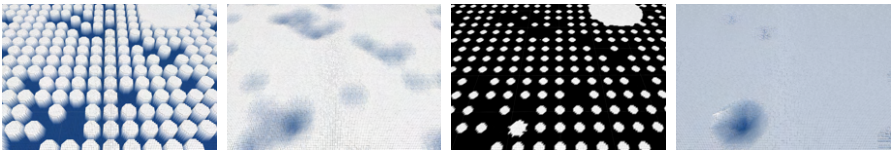
Surface normals are computed by fitting a tangent plane to a neighbourhood within a small radius of each grid cell (see Appendix 6.11.1).

Traversability SDF is the SDF in 2D storing the distance of any points from the nearest untraversable region (see Fig. 6.5c). We refer to this field as sdf_2 in further text. More details about how terrain traversability is classified can be found in Appendix 6.11.1.

Elevation SDF is a signed distance field in 3D which is used for collision avoidance. The value is positive if the point is outside of the terrain and negative if the point lies inside the terrain. We refer to the 3D SDF field as sdf_3 in further text. sdf_3 is computed using the implementation from *grid_map* package which can be found open source[1].

Filtered elevation, denoted by h_f, is a smoothed version of the terrain height visualized in Fig. 6.5d. The smooth elevation is used by the base pose selection module (Sec. 6.3.3.1). Implementation details can be found in Appendix 6.11.1.

[1] `https://github.com/ANYbotics/grid_map`



(a) Raw Elevation **(b)** Elevated Mean **(c)** Untraversable **(d)** Filtered terrain

Figure 6.5: Intermediate steps in terrain preprocessing for stepping stones terrain. ***a)*** Raw elevation map input. ***b)*** Elevated mean (see Appendix 6.11.1). ***c)*** Untraversable terrain shown in black color. ***d)*** Filtered terrain, used by the base pose selector module.

6.3.3 Sampling-Based Planning

With the roadmap computed offline and the terrain pre-processed, we are ready to use the RRT based planner online. The planner attempts to connect $T_{B,start}$ and $T_{B,goal}$ geometrically (only considers the kinematics). The state space $\mathcal{S}$ is defined in Eq. 6.4 with n_J being the number of limb joints (actuated) and $\{0,1\}^{|\mathcal{I}|}$ contact states of all limbs $\mathcal{I}$. SBPs main components are described below.

$$\mathcal{S} \coloneqq \mathbb{R}^3 \times SO(3) \times \mathbb{R}^{n_J} \times \{0,1\}^{|\mathcal{I}|}. \tag{6.4}$$

6.3.3.1 Sampling

Although the planner is a full 3D planner, the sampling module operates in *SE(2)* space which is motivated by the use of legged systems. Since legged robots locomote by interacting with the terrain, their base has to be in the vicinity of the terrain surface. Therefore, we uniformly sample the base's x, y position and the orientation in yaw γ and compute the remaining DoFs from the terrain features. Note that the base pose selection module operates on filtered terrain, which only retains prominent terrain features (see Sec. 6.3.2). Roll α and pitch β are computed from the terrain normal $\boldsymbol{n}$ such that the robot roughly stays parallel to the surface below the base. The z coordinate can then be computed simply as $z = h_{des} + h_f(x, y)$ where h_{des} is a user-defined desired height above the ground and $h_f(x, y)$ is the filtered terrain height.

6.3.3.2 Expansion

Expansion refers to connecting the newly drawn sample to the rest of the tree. The maximum connection length is a tuning parameter (15 m for HEAP and 5 m for the legged robot). The connecting path for the base is computed using Reeds-Shepp (RS) curves Reeds and Shepp, 1990, which give an optimal path between two poses in *SE(2)* while respecting a minimal turning radius

constraint. Upon drawing a random sample (x, y, γ), we use RS curve length d_{rs} to determine its nearest neighbors. The RS path is computed without considering any terrain information; it just helps enforce the rolling constraints for robots with steerable wheels. Using straight-line connections might render the problem infeasible in the *refinement step* (see Sec. 6.5). For legged robots, we also use RS curves with small turning radii since they naturally minimize sideways motions, typically less stable and slower than locomoting forwards.

6.3.3.3 Feasibility Checking

SBP checks the connection feasibility before connecting the newly drawn sample to the rest of the tree. Unlike Tonneau et al., 2018a where a *guiding path* is computed first, we do not add the connection into the tree before calculating the entire trajectory and ensuring its feasibility with four criteria.

- *Kinematic Reachability*: A contact must lie inside reachable workspace which is approximated by the roadmap.
- *Collision Avoidance*: Robot's base and limbs must not be in collision with the environment for each state along the proposed connection (similar to Escande, Kheddar, and Miossec, 2013).
- *Valid Contact Positions*: All contacts need to lie within traversable terrain $\mathcal{T}$.
- *Stability Criteria (relaxed)*: CoM is allowed to deviate at most ϵ_s (parameter) from the support polygon edge. In addition, we impose a lower bound on the support polygon's area. For $\epsilon_s = 0$ the robot is statically stable.

Kinematic reachability can be verified by transforming roadmap vertices into the world frame and then using Eq. 6.2. This can be evaluated quickly using the pre-computed roadmap and the elevation map. The collision check can be easily enforced using sdf_3 from Sec. 6.3.2. It suffices to ensure that collision spheres $B(\boldsymbol{c}, r)$ (ball of radius r centered at $\boldsymbol{c}$) are outside the terrain, i.e. $sdf_3(\boldsymbol{c}^x, \boldsymbol{c}^y, \boldsymbol{c}^z) \geq r$. Fig. 6.3d shows an example of collision geometry for the legged robot. Contact validity can be checked using sdf_2 from Sec. 6.3.2 and Eq. 6.1. Lastly, the algorithm checks the support polygon stability criterion, which does not hold in arbitrary terrain configuration, and care needs to be taken to ensure stability. We empirically found that the *refinement step*

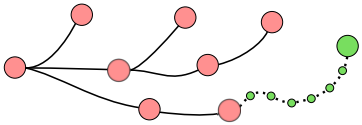


Figure 6.6: Addition of a new node into the RRT tree (expansion step). Red circles and full black lines are nodes and paths that make the current RRT tree. The newly sampled node (green) has to be connected to the rest of the tree. For a successful connection, all subnodes on the connecting path (small green circles) have to be valid. For a legged robot, the subnodes are connected using One-Step Motions (OSMs), which is illustrated in Fig. 6.7.

can stabilize the robot even when the CoM projection is slightly outside the support polygon (controlled by ϵ_s).

Fig. 6.6 depicts the feasibility checking. Upon drawing a new sample (large green node) as described in Sec. 6.3.3.1, we compute a RS connection (dotted line) to the new node, as described in Sec. 6.3.3.2. Subsequently, the dotted line is discretized into subnodes (small green circles) using an RS interpolation method Sucan, Moll, and Kavraki, 2012. The subnodes do not necessarily be equidistant. Next, for each subnode we generate full 6 DoF pose using the filtered elevation layer inside the *grid_map*. Finally, each subnode undergoes feasibility checking, ensuring that all criteria above are satisfied. If every subnode is feasible, the new state and the connecting path are added to the tree. For detailed description, refer to Sec. 6.4.

6.3.3.4 Cost Function

An optimizing SBP in the *Initialization Step* allows us to minimize a user defined cost function. Eq. 6.5 introduces our cost c, composed of RS path length $d_{rs}(\Gamma)$ and sum of robot-specific costs on each state $L(\boldsymbol{s})$ along the trajectory Γ.

$$c = d_{rs}(\Gamma) + \sum_{\boldsymbol{s}\in\Gamma} L(\boldsymbol{s}), \quad \Gamma \subset \mathcal{S}. \tag{6.5}$$

Robot specific costs are given with Eq. 6.6 (HEAP) and Eq. 6.7 (ANYmal and ANYmal on wheels). For HEAP, we penalize the legs going over untraversable terrain $\neg\mathcal{T}$ to prefer driving over stepping (as shown in our previous work Jelavic, Farshidian, and Hutter, 2021). ${}^{\neg\mathcal{T}}\mathbb{1}_{ee,i}$ is an indicator function with a value 1 if the i^{th} limb EE is inside the untraversable area $\neg\mathcal{T}$. w is the

user-defined weight. For the legged robot, we penalize the roll (α) and pitch (β) angles of the base with different weights.

$$L(\boldsymbol{s}) = w \sum_{i \in \mathcal{I}} {}^{\neg\mathcal{T}}\mathbb{1}_{ee,i} \qquad \text{for HEAP,} \tag{6.6}$$

$$L(\boldsymbol{s}) = w_\alpha \operatorname{abs}\alpha + w_\beta \operatorname{abs}\beta \quad \text{for others.} \tag{6.7}$$

6.4 Feasibility Check

We introduce a subroutine called OSM, which does additional discretization between the green subnodes in Fig. 6.6 to ensure connection feasibility. We describe the algorithm for the legged robot first since it is the most general, and then the special cases for other robots.

When all feet are grounded, the robot is in a Full-Support State (FSS). OSM is the transition between two FSSs, during which at most one swing phase (corresponding to one step) happens for each leg. We assume the robot constantly moves between two FSSs which will always result in a short full stance phase. This chapter describes how OSM is used inside RRT and subsequently all the subroutines.

6.4.1 Using One-Step Motion with RRT

Proposed OSM is used inside RRT expansion step. After sampling the base pose, the new vertex ($\boldsymbol{v}_{new}$) needs to be connected to the tree. We find a nearest neighbor in the tree ($\boldsymbol{v}_{nn}$) and use Alg. 8 (see Appendix 6.11.2) to check validity between the two samples. If $\boldsymbol{v}_{nn}$ and $\boldsymbol{v}_{new}$ can be connected, Alg. 8 returns a sequence of OSMs that achieve the connection. The maximum length of the edge between $\boldsymbol{v}_{nn}$ and $\boldsymbol{v}_{new}$ can be tailored to different terrains.

OSMs do not have to be the same length. The planner receives a user-defined list of lengths for OSM, $\mathcal{L} = [l_1, l_2, \ldots, l_n]$ which is sorted in descending order. Empirically we found that the planner makes use of longer connections first, which achieve fast progress in easier terrains and resorts to using shorter connections when the terrain gets harder. The process of adding a new vertex with OSMs of different lengths is illustrated in Fig. 6.7.

6.4.2 One-Step Motion Creation

OSM creation routine (Alg. 6 in Appendix 6.11.2) receives a full state as a starting point and a base pose for the goal. It first generates an FSS for the

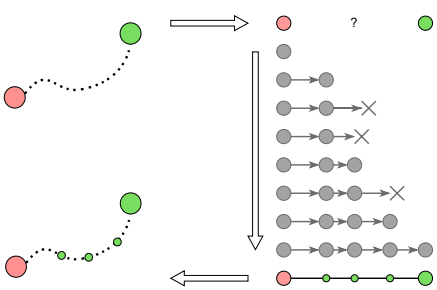


Figure 6.7: Path validation using OSMs. Each gray node and arrow refers to an FSS and an OSM, respectively. Crosses represent failed OSM creations. Top to bottom: we first create an FSS for $\boldsymbol{v}_{nn}$, and then successfully generates the first OSM (the first length defined in lentgh candidates $\mathcal{L}$). The second OSM tries two lengths before the third length succeeds. The third OSM tries one length before the second one succeeds. Finally the last OSM connects to $\boldsymbol{v}_{new}$ in one try.

goal base pose, then discretize the connecting RS path, find a feasible contact schedule, and create valid swing motions. Motion duration is computed in the end. The algorithm returns *true* if a feasible OSM is discovered and *false* otherwise. We illustrate an example OSM trajectory on two different terrains in Fig. 6.10.

6.4.2.1 Full-Support State Computation

FSS creation subroutine receives a base pose and computes joint configuration (one keyframe). We define nominal hip positions[2] which are shown with blue spheres in Fig. 6.8 on the left. A *desired* foothold is computed as a traversable point on the terrain surface closest to the nominal hip position. Subsequently, joint positions $\boldsymbol{q}$ are created by searching *valid* roadmap vertices in a user-defined neighborhood of the *desired* foothold. A vertex is considered *valid* when foot position is in contact with the traversable terrain $\mathcal{T}$ (Eq. 6.1 and Eq. 6.2) and the corresponding leg collision free (see Sec. 6.3.3.3). In case no valid vertex is found, FSS creation for this state returns failure. Fig. 6.8 shows FSS creation on the stairs. One FSS corresponds to a green subnode in Fig. 6.6 and Fig. 6.7.

6.4.2.2 Full-Support State Connection

The base path connecting the OSM start and the goal is discretized into subnodes with the detailed algorithm description found in Appendix 6.11.2. In Short and Bandyopadhyay, 2017, two FSSs connect with a creep gait, i.e. the robot first moves the limbs and then the base with all 4 limbs on the ground. In contrast, our two-stage approach can accommodate for simultaneous motion of the base and swing limbs, which increases the overall motion speed by 35 % and allows overcoming larger obstacles.

[2]computed by projecting nominal footholds onto the base frame's x-y plane

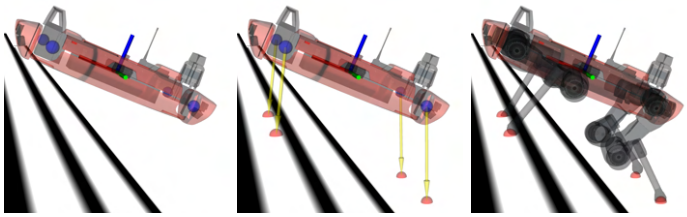


Figure 6.8: FSS computation on stairs (white shows traversable terrain, black untraversable). ***Left:*** Base and nominal hip positions (blue spheres). ***Middle:*** Desired footholds (red spheres) computed using nominal hip positions. ***Right:*** FSS with leg configurations.

6.4.2.3 Contact Schedule Computation

Once the OSM connection has been discretized into subnodes, the algorithm attempts to compute a feasible contact schedule. Generally, the robot has to break contact once the foot is at the edge of its workspace. This observation has been used in Tonneau et al., 2018a where FIFO queue is used to decide which limb to recover first. In contrast, we allow for earlier contact breaks giving the planner more freedom to reshape the support polygon in anticipation of obstacles. First we find the latest contact breaking point and earliest contact establishing point for each limb. We define the Latest Contact Break (LCB) for limb i as the latest subnode when limb's swing phase must start (we must break contact because the limb is in the kinematic limits). The counterpart of LCB is the Earliest Contact Creation (ECC), i.e. the earliest subnode that can establish a valid contact with footholds at OSM goal state. Contact schedule is computed by solving a feasibility problem (see Eq. 6.33) where LCB and ECC define the constraints. Essentially, we seek to compute contact breaks no later than the LCB and contact creations no earlier than ECC. An example of contact schedule computation is shown in Fig. 6.9 with detailed description given in Appendix 6.11.2.

6.4.2.4 Swing Motion Validation

The last step in OSM creation is to validate swing motion. We look for feasible swing trajectories after finding the contact schedule. Swing motion can be verified by invalidating portions of the roadmap and then performing the graph search, as suggested in Short and Bandyopadhyay, 2017. However, in our approach, the roadmap is not stationary during a swing phase because the base moves. We instead search collision-free vertices near the desired swing position that are user-defined distance above the terrain.

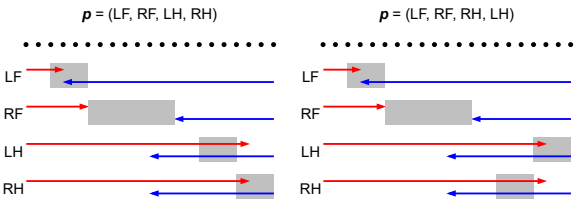


Figure 6.9: Contact schedule computation for swing orders (LF, RF, LH, RH) and (LF, RF, RH, LH). The black dots on the top represents the OSM states. Each red arrow points from the start to LCB, and each blue one from the goal to ECC. Swing phases generated from Alg. 5 (see Appendix 6.11.2) are shown as gray rectangles. There exist many other feasible contact schedules (e.g., move swing phase to the left, increase the size of swing phase).

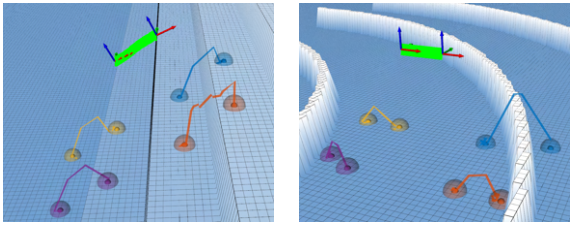


Figure 6.10: OSM with length = 20 cm created on stairs ***(left)*** and obstacles ***(right)***. Terrain color represents the height value (white: maximum, blue: minimum). The base trajectory is shown in green, limb trajectories in separate colors. The spheres on the terrain show liftoff and touchdown locations.

6.4.2.5 Timing Computation

Transition duration between two states along the path is computed as follows:

$$\Delta t_l = t(i+1) - t(i) = \frac{1}{v_J}\Big(\|\boldsymbol{q}_J(i+1) - \boldsymbol{q}_J(i)\|_\infty \Big) \tag{6.8}$$

where $\boldsymbol{q}_J$ is the vector of joint positions and v_J is the expected mean joint velocity (same as Jelavic, Farshidian, and Hutter, 2021).

6.4.3 Legged-Wheeled Robot

By modifying the LCB and ECC search, we can reuse the OSM concept for legged-wheeled robots such as ANYmal on wheels Bjelonic et al., 2019a. A description on how to find LCB and ECC for a legged-wheeled robot is given in Appendix 6.11.3. Compared to legged robot duration Δt_l (see Eq. 6.8), one

needs to consider the average driving speed. Transition duration between two states along the path is thus slightly different:

$$\Delta t_{lw} = \max\left\{\Delta t_l, \frac{1}{v_B}\left(\|\boldsymbol{r}_B(i+1) - \boldsymbol{r}_B(i)\|_2\right)\right\} \quad (6.9)$$

where $\boldsymbol{r}_B$ is the position of the base and v_B is the expected mean driving speed.

6.4.4 Legged Excavator

OSM method is slightly modified to work for systems like legged excavators, whereby the planner has to handle legs with wheels that can move and an arm that cannot move when in contact. We discretize the connection as described in Sec. 6.3.3.3 with maximal distance between the subnodes being 20 cm (green circles in Fig. 6.6).

It is most efficient to keep all four legs in contact and drive, which can be seen as FSS for HEAP. The planner allows configurations with minimum three contact limbs (e.g. 2 legs and an arm). When the contact state switches, e.g., HEAP needs to lift Left Front (LF) leg while Right Hind (RH) is airborne, we insert an FSS in between, which is a special case of OSM where the length is not predefined.

When the arm has to move, HEAPs base stops, and we use the roadmap to find a collision-free path similar to Short and Bandyopadhyay, 2017. If the arm has to establish contact (e.g., for going over a gap), we find the corresponding contact establishing and contact breaking point. The arm contact position $\boldsymbol{p}^*$ is found by solving

$$\min \|\boldsymbol{p}_{cc} - \boldsymbol{p}_{cb}\| \quad (6.10)$$

and setting $\boldsymbol{p}^* = (\boldsymbol{p}^*_{cc} + \boldsymbol{p}^*_{cb})/2.0$, where $\boldsymbol{p}_{cb}$ and $\boldsymbol{p}_{cc}$ are positions of valid vertices in arm roadmap at contact break and creation subnodes. Each contact node's arm joint angles are computed solving Inverse Kinematics (IK) $\boldsymbol{q} = \text{IK}(\boldsymbol{p}^*)$ as in Jelavic, Farshidian, and Hutter, 2021.

6.5 Short Term Planning: Refinement Step

Optimization is the backbone of *Refinement step*. It refines the keyframe trajectory $\boldsymbol{\Gamma}$ found in the *Initialization step* to ensure smooth, stable, and feasible motions. $\boldsymbol{\Gamma}$ is a sequence of full body poses and contact states. Full states

help the optimization convergence instead of just providing base poses (see Sec. 6.7.2). Optimization can modify the whole body states but not the Contact Schedule. As opposed to offline Trajectory Optimization (TO) Jelavic, Farshidian, and Hutter, 2021; Jelavic and Hutter, 2019, we solve the optimization in an MPC fashion, which brings robustness during execution as it can compensate for small deviations from the reference $\mathbf{\Gamma}$. The optimization can deviate from the initial plan if fulfilling systems constraints or rejecting disturbances requires so.

The *Refinement Step* solves an Optimal Control Problem (OCP) given below:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{u}} \quad & \Phi(\boldsymbol{x}(T)) + \int_0^T L(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, dt \\ \text{s.t.} \quad & \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \\ & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \\ & \boldsymbol{g}_1(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = \mathbf{0} \\ & \boldsymbol{g}_2(\boldsymbol{x}(t), t) = \mathbf{0} \\ & \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \geq \mathbf{0}, \end{aligned} \tag{6.11}$$

where L is a time-varying running cost, Φ is the cost at the terminal state, and T is the prediction horizon. The state and input are represented by $\boldsymbol{x}$ and $\boldsymbol{u}$, respectively. A Riccati-based solver solves the optimal control problem in Eq. 6.11. In particular, we use the Differential Dynamic Programming (DDP) method from Grandia et al., 2019b and the Direct Multiple Shooting (DMS) scheme from Grandia et al., 2022. Both methods handle state-input equality constraints $\boldsymbol{g}_1$ using the projection technique Nocedal and Wright, 1999. Pure state equality constraints $\boldsymbol{g}_2$ are enforced using augmented Lagrangian Sleiman, Farshidian, and Hutter, 2021. The state-input inequality constraints $\boldsymbol{h}$ are handled using relaxed barrier functions as proposed in Grandia et al., 2019b. Transcription implementation, as well as optimization solvers, are publicly available in the OCS2 optimal control toolbox Farbod, n.d.

6.5.1 MPC for Legged Robot

OCP given with Eq. 6.11 has been solved using Sequential Linear Quadratic (SLQ) algorithm presented in Grandia et al., 2019b. We used prediction horizon of $T = 1\ s$ with the maximal step size of $0.01\ s$. Below we describe the dynamic model and constraints used in the MPC formulation.

6.5.1.1 System Dynamics

Legged robot state is composed of the position of the CoM $\boldsymbol{p}$ in world frame, Euler angles $\boldsymbol{\theta}$ ($Z - Y' - X''$ sequence) of the base, joint position $\boldsymbol{q}_{J,i}$ for leg $i = \{1, 2, 3, 4\}$, and the linear velocity ${}^{\mathcal{B}}\boldsymbol{v}$ and the angular velocity ${}^{\mathcal{B}}\boldsymbol{w}$ of CoM. The total number of joints for the robot is $n_J = 12$. Since the legs are much lighter than the body, CoM is assumed to be configuration invariant. Robot's state can be written as $\boldsymbol{x} = \left(\boldsymbol{p},\ \boldsymbol{\theta},\ \boldsymbol{q}_J,\ {}^{\mathcal{B}}\boldsymbol{v},\ {}^{\mathcal{B}}\boldsymbol{\omega}\right) \in \mathbb{R}^{24}$ and control input $\boldsymbol{u}$ as $\boldsymbol{u} = \left(\boldsymbol{u}_J,\ {}^{\mathcal{B}}\boldsymbol{\lambda}_{ee}\right) \in \mathbb{R}^{24}$, where $\boldsymbol{u}_J$ is the joint velocity and ${}^{\mathcal{B}}\boldsymbol{\lambda}_{ee,i} \in \mathbb{R}^3$ is the contact force at i^{th} EE. We use a kinodynamic model describing the dynamics of a single rigid free-floating body together with the kinematics for each leg. The Equations of Motion can be found in Grandia et al., 2019b.

6.5.1.2 Cost Formulation

MPC should track $\boldsymbol{\Gamma}$, however it is allowed to deviate. Hence, deviation from the reference state is penalized quadratically. The reference twist is computed using finite differences, and the reference forces are equally distributed on stance feet. High-frequency input is also penalized as proposed in Grandia et al., 2019a.

6.5.1.3 Constraints

We use $\mathcal{C}$ to denote the set of contact limbs.

Zero Force or Velocity Constraint: A contact leg cannot change its position while a swing leg cannot generate a contact force,

$$\begin{cases} \boldsymbol{v}_{ee,i} = \boldsymbol{0}, & i \in \mathcal{C}, \\ \boldsymbol{\lambda}_{ee,i} = \boldsymbol{0}, & i \notin \mathcal{C}. \end{cases} \tag{6.12}$$

Foot Constraint: The foothold of a contact leg must be placed at its reference from $\boldsymbol{\Gamma}$, while for a swing leg, its foot should maintain a certain height to avoid foot scuffing,

$$\begin{cases} \boldsymbol{p}_{ee,i} = \bar{\boldsymbol{p}}_{ee,i}, & i \in \mathcal{C}, \\ \boldsymbol{p}^z_{ee,i} \geq h(\boldsymbol{p}^x_{ee,i}, \boldsymbol{p}^y_{ee,i}) + c(t), & i \notin \mathcal{C}, \end{cases} \tag{6.13}$$

where $\boldsymbol{p}_{ee,i}$ is the foot position of i^{th} limb and $\bar{\boldsymbol{p}}_{ee,i}$ is the reference. $h(\cdot, \cdot)$ evaluates the terrain height. $c(\cdot)$ is the clearance function used to minimize the constraint violation and thus make the problem more numerically stable for the optimizer (see Fig. 6.11).

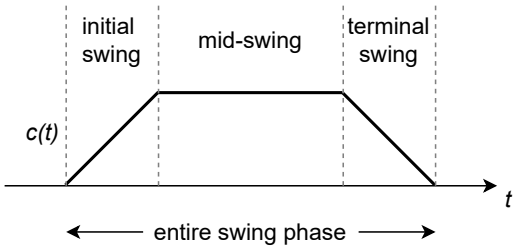


Figure 6.11: The ground clearance function. It gradually increases to a user-defined value and then gradually decreases. Such a shape minimizes violation in the initial and terminal swing phases since feet cannot change state instantly.

Friction Cone Constraint: We use the perturbed cone proposed in Grandia et al., 2019b to avoid differentiability problems around the origin.

$$h_{cone,\epsilon} = \mu_c F_z - \sqrt{F_x^2 + F_y^2 + \epsilon^2} \geq 0, \tag{6.14}$$

where μ_c is the friction coefficient, ϵ is a control parameter, and $\boldsymbol{F} = [F_x, F_y, F_z]$ is the contact force expressed in the local frame of the terrain surface.

Push Force Constraint: To enforce stability margins, the vertical component of contact forces ${}^{\mathcal{W}}\boldsymbol{\lambda}^z_{ee,i}$ for contact legs needs to be larger than a user-defined value,

$${}^{\mathcal{W}}\boldsymbol{\lambda}^z_{ee,i} \geq \lambda_{z,\min}, \quad \forall i \in \mathcal{C}. \tag{6.15}$$

Leg Collision Constraint: sdf_3 is used to prevent collision of knee and shin with the terrain,

$$sdf_3(\boldsymbol{c}) \geq r \quad \text{for all limb collision spheres } B(\boldsymbol{c}, r). \tag{6.16}$$

Limb collision spheres for ANYmal can be found in Fig. 6.3d.

6.5.2 MPC for HEAP

The OCP for HEAP is transcribed into a finite-dimensional NLP using DMS Bock and Plitt, 1984 and solved with Sequential Quadratic Programming (SQP). We used prediction horizon of $T = 15\ s$. Below we describe the differences in OCP formulation compared to the legged robot.

6.5.2.1 System Dynamics

HEAP state is given with $\boldsymbol{x} = \left(\boldsymbol{p},\ \boldsymbol{\theta},\ \boldsymbol{q}_J\right) \in \mathbb{R}^{31}$. Control input can be written as $\boldsymbol{u}$ as $\boldsymbol{u} = \left(\boldsymbol{v},\ \dot{\boldsymbol{\theta}},\ \dot{\boldsymbol{q}}_J,\ \boldsymbol{\alpha}\right) \in \mathbb{R}^{36}$. Euler angles $\boldsymbol{\theta}$ ($Z - Y' - X''$ sequence)

parametrize the rotation, $\boldsymbol{p}$ is the base position, $\boldsymbol{q}_J$ are joint positions and $\boldsymbol{v}$ is the linear velocity of the base. $\boldsymbol{\alpha}$ is a vector of weights used to enforce robust support polygon constraint in Eq. 6.20. System dynamics can then be described as:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{S}\,\boldsymbol{u}(t), \tag{6.17}$$

where $\boldsymbol{S}$ is a selection matrix of the form $\boldsymbol{S} = \begin{bmatrix}\mathbf{I}_{31\times 31}\ \mathbf{0}_{31\times 5}\end{bmatrix}$. Following our previous work Jelavic and Hutter, 2019, HEAP uses a kinematic model since such a heavy machine is operated in quasi-static conditions.

6.5.2.2 Cost Formulation

The cost function follows the same principle as for the legged robot, i.e., the solution should stay close to $\boldsymbol{\Gamma}$ computed by the SBP. Since HEAP has one hydraulic circuit that spins all wheels at approximately the same velocity, we add cost terms that help bridge the simulation to the hardware gap. The cost term below minimizes slip by penalizing the linear wheel velocity difference between the left and the right side.

$$L(\boldsymbol{x}) = \sum_{i,j\in\mathcal{C}_l} \left\|\boldsymbol{v}_{ee,i} - \boldsymbol{v}_{ee,j}\right\|_2^2, \tag{6.18}$$

where $\boldsymbol{v}_{ee,i}$ denotes the velocity of the wheel center. We use $\mathcal{C}$ for contact limbs (legs and arm) and $\mathcal{C}_l$ for contact legs only.

6.5.2.3 Constraints

Rolling Constraint prevents lateral slip, i.e. linear velocity of the wheel center stays perpendicular to wheel joint axis.

$$^{\mathcal{E}_i}\boldsymbol{v}^{y}_{ee,i} = 0, \quad \forall i \in \mathcal{C}_l, \tag{6.19}$$

with $^{\mathcal{E}_i}\boldsymbol{v}^{y}_{ee,i}$ being y component of the wheel center velocity expressed in i^{th} end-effector frame $\mathcal{E}_i$. Optimization is not aware of the wheel joint, it treats the wheel as a moving contact point which reduces number of variables.

Stability Constraint: We use the robust support polygon constraint introduced in Jelavic and Hutter, 2019 which avoids the need to compute half spaces manually:

$$\begin{aligned} &\sum_{i \in \mathcal{C}} \Big(\alpha_i + \frac{\epsilon}{|\mathcal{C}|}\Big) \boldsymbol{p}_{ee,i}^{x,y} = \boldsymbol{p}_{com}^{x,y} \ \in \mathbb{R}^2, \\ &\sum_{i \in \mathcal{C}} \alpha_i = 1 - \epsilon, \quad 0 \leq \epsilon \leq 1, \ \alpha_i \geq 0, \end{aligned} \tag{6.20}$$

where α_i are part of the input vector $\boldsymbol{u}$, ϵ is a user-defined parameter, and $\boldsymbol{p}_{com}$ is the position of CoM. The bigger the ϵ, the more conservative we are, i.e., the minimum permitted distance of CoM to the edge of the support polygon becomes larger. We set $\epsilon = 0.1$ in this work.

Swing Limb Constraint: We follow the same formulation as for the legged robot. The user-defined height threshold was 0.6 m for the legs and 0.9 m for the shovel.

Contact Limb Constraint: We relax the foothold constraint from Eq. 6.13; MPC can optimize contact position as long as the limb stays grounded within the traversable area (Eq. 6.1 and Eq. 6.2).

6.6 Control

6.6.1 Legged Robot Control

The Low-level Whole Body Controller (WBC) for the legged robot is based on the hierarchical optimization from Bellicoso et al., 2016. It considers full nonlinear rigid body dynamics of the robot and it solves a series of Quadratic programs (QPs) tasks in a prioritized order through nullspace projection. Table 6.1 shows tasks and priorities used. WBCs output torques are augmented with a PD term for stance legs. For more details on the specific WBC version, the reader is referred to Grandia et al., 2022.

6.6.2 HEAP Control

Compared to the legged robot and our previous work Jelavic et al., 2020, we omit the WBC based on HO Bellicoso et al., 2016. Instead, through the use of Virtual Model Controller (VMC) we retain the good qualities from our previous work Jelavic et al., 2020 including the terrain adaptation. The new

Table 6.1: WBC task setup. Smaller number indicates higher priority.

Priority	Type	Task
1	=	Floating base equations of motion
1	≤	Joint torque limits
1	≤	Joint position limits
1	≤	Joint velocity limits
1	≤	Friction cone
1	=	Stance legs no contact motion
2	=	Swing leg tracking
3	=	Base orientation tracking
3	=	Base translation tracking
4	=	End-effector force tracking

controller can use the arm as a support limb and it shows higher robustness. Lastly, the presented controller relies on existing tuning of hydraulic cylinders which removes the need for laborious tuning of the low-level torque control loops. Below we describe how VMC is combined with MPC and how each individual joint is controlled.

6.6.2.1 Virtual Model Control

Motion of the CoM is governed by the net resulting force acting on it. Assuming quasi-static conditions ($\ddot{\boldsymbol{p}} = \ddot{\boldsymbol{\theta}} = 0$), virtual force f_v and virtual moment $\boldsymbol{m}_v$ can then be computed by PID control law. This is expressed in Eq. 6.21.

$$\begin{aligned} Mg + \sum_i \boldsymbol{\lambda}^z_{c,i} &= f_v = \mathrm{PID}(z_{des} - z_{meas}), \\ \sum_i \boldsymbol{r}_{c,i} \times \boldsymbol{\lambda}_{c,i} &= \boldsymbol{m}_v = \mathrm{PID}(\boldsymbol{\theta}_{des} - \boldsymbol{\theta}_{meas}). \end{aligned} \tag{6.21}$$

$\boldsymbol{p}$ is the position, g is the gravitational acceleration, and $\boldsymbol{\theta}$ is rotation parametrized by Euler angles. $\boldsymbol{\lambda}_{c,i}$ is a contact force at i^{th} contact point and $\boldsymbol{r}_{c,i}$ is the distance from the contact point to the CoM. We only regulate the height in z direction, roll θ_r and pitch θ_p angle of the base. The contact forces required to produce desired virtual force and moment can then be recovered by solving a QP:

$$\begin{aligned} \min_{\boldsymbol{\Lambda}} \quad & \boldsymbol{\Lambda}^T \boldsymbol{W} \boldsymbol{\Lambda} \\ \text{s.t.} \quad & \boldsymbol{\Lambda}_{lb} \leq \boldsymbol{\Lambda} \leq \boldsymbol{\Lambda}_{ub} \\ & \boldsymbol{A}\boldsymbol{\Lambda} = \boldsymbol{b}, \end{aligned} \tag{6.22}$$

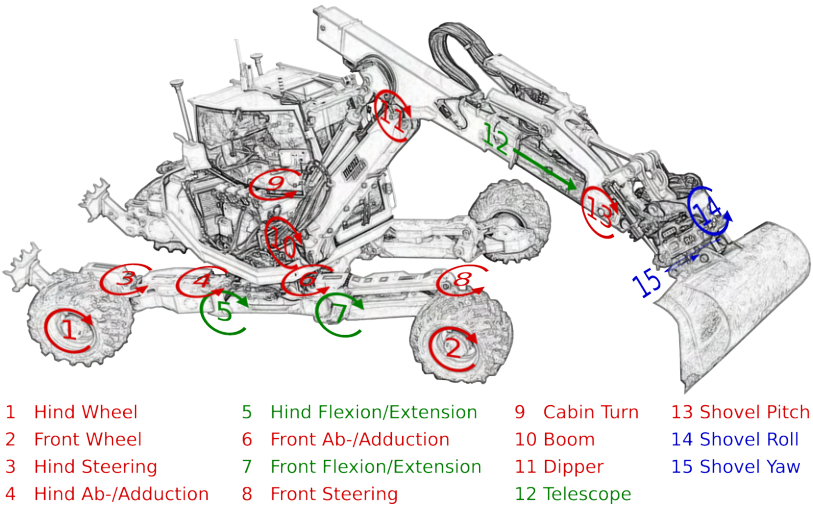


Figure 6.12: HEAP joints. Blue joints are always frozen and red joints are always velocity controlled. Green joints are torque controlled when corresponding limb is in contact, velocity controlled otherwise. Figure adapted from Jud et al., 2021b.

where $\boldsymbol{\Lambda}$ is a stacked vector of contact forces, $\boldsymbol{W}$ is a weighting matrix and the constraint $\boldsymbol{A}\boldsymbol{\Lambda} = \boldsymbol{b}$ arises from the geometry of the contact points w.r.t. to the CoM (Eq. 6.21). The solution of Eq. 6.22 yields minimal contact forces which are distributed as equally as possible. The joint torques can then be recovered with $\boldsymbol{J}^T\boldsymbol{\Lambda} = \boldsymbol{\tau}$ where $\boldsymbol{J}$ is the stacked contact Jacobian matrix. Compared to our previous work Hutter et al., 2016b, we include the arm into the control structure as a contact point.

6.6.2.2 HEAP Whole Body Controller

While MPCs purpose is to track the desired plan, VMC makes the system terrain-adaptive and compliant. In practice we cannot acquire a full accuracy map with on-board sensors. Furthermore, such a map has too many details which make the gradients non-smooth. Hence the system needs to plan using simplified geometry and adapt to the unmodelled terrain at runtime. We achieve this by utilizing a minimal set of torque-controlled joints. An important consideration is that hydraulic cylinders have a lot of friction, and in some cases (e.g. tele joint in Fig. 6.12) the friction is even position dependent. However, if the joint supports lot of weight, the amount of friction torque in the total torque is relatively low and the resulting control performance is bet-

ter compared to non-load bearing joints. We control Hip Flexion-Extension (HFE) and Telescopic (TELE) joints in torque mode given that these joints bear the most load when corresponding limb is in contact. Since Contact Schedule (CS) allows only configurations with at least three contacts, there is always enough DoFs to control roll, pitch and height.

Fig. 6.13 shows the structure of combined MPC and VMC controllers. The MPC receives a trajectory $\mathbf{\Gamma}$ from the SBP. Optimized roll, pitch and height are fed to VMC which then computes HFE and TELE joint torques to achieve desired base pose. Lower level controller selects joint control mode based on the contact schedule CS. All joints are velocity controlled except for HFE and TELE if the corresponding limb is in contact. Both torque and joint velocity commands are filtered using exponential smoothing to remove small jumps. MPC is tuned to accurately follow the plan from the SBP. MPC uses a prediction horizon of 15 s, which is enough to correct for small deviations around the long term plan $\mathbf{\Gamma}$. In contrast, the VMC control loops are tuned for good disturbance rejection since VMC sees the terrain as a disturbance. VMC uses a state machine to handle late touchdown events. A swing leg that

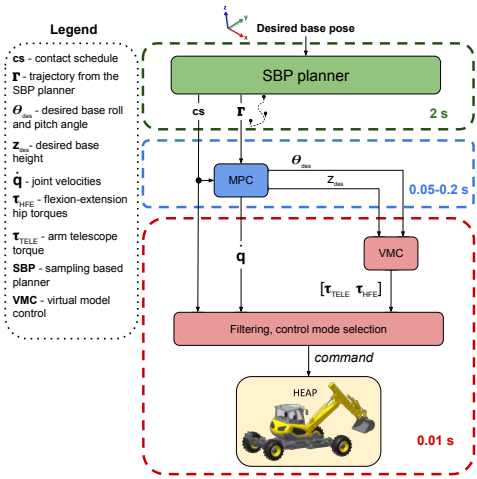


Figure 6.13: HEAP control system with layer execution times on the right. MPC tracks trajectory Γ from SBP using a kinematic model from Eq. 6.17. VMC computes torques for Hip Flexion-Extension and arm Telescopic joint to track roll, pitch and, base height setpoints from the MPC. Joint control mode is selected based on contact schedule.

should be in contact (based on contact schedule) is commanded a constant velocity in HFE joint until contact is established. Late touchdowns require dedicated treatment since they may result in instability. We found that early touchdowns are not problematic; they merely degrade the base tracking performance. With dedicated late touchdown handling, we could perform stepping motion in terrain with a roughness of about ±50 cm.

To execute some of the maneuvers shown in the result section, HEAP needs to control wheel velocities. The lack of wheel speed measurements (no oil flow sensor in the wheel rotary actuators) is mitigated by closing the PI control loop over the base velocity. Given desired wheel velocities ω we can compute linear velocities of the wheel center $v_{center} = \omega R_w$ (R_w is wheel radius). The desired base speed $v_{B,des}$ is approximated by taking the average over grounded wheel linear velocities. Base speed v_B in a local frame can be estimated by differentiating the base position and taking a dot product with the heading of the base. The final estimate is obtained after applying an exponential filter.

Control signal u is computed with a PID control law with feedforward term $k_{ff}\, v_{B,des}$. u is applied to all four wheels since they cannot be controlled individually.

$$u = \mathrm{PID}\big(v_{B,des} - v_B\big) + k_{ff}\, v_{B,des}, \tag{6.23}$$

6.7 Results

The proposed LSTP is implemented entirely on Central Processing Unit (CPU). All components are implemented in C++ programming language with ROS as a communication middleware. SBP uses the OMPL library Sucan, Moll, and Kavraki, 2012 and MPC is implemented using OCS2 library Farbod, n.d., both of which are available as open source. OCS2 relies on Pinocchio library Carpentier et al., 2019 for rigid body dynamics and all the derivatives are computed using CppAd framework Bell, 2012. Terrain processing is common to all robots. This chapter presents results with a different robot in separate sections. For a better impression please refer to the video[3].

6.7.1 Terrain Preprocessing

Normal estimation is a key component in our terrain preprocessing pipeline. It is used by the base pose selector and later by the traversability estimation. Fig. 6.14 left shows the runtime of normal estimation on a fixed size 8m $\times$ 8m map at 3 cm resolution with varying normal estimation radius R. The algorithm complexity is $O(R^2)$ since the number of points used for estimation is proportional to the neighborhood surface (see Appendix 6.11.1). The total runtime of our processing pipeline with varying map sizes (resolution is always 3 cm) is shown in the middle. The pipeline can maintain more than 1 Hz real-time rate updates, even for 20×20 meter maps. The right plot shows the

[3] `https://youtu.be/4eqq4Ucl92o`

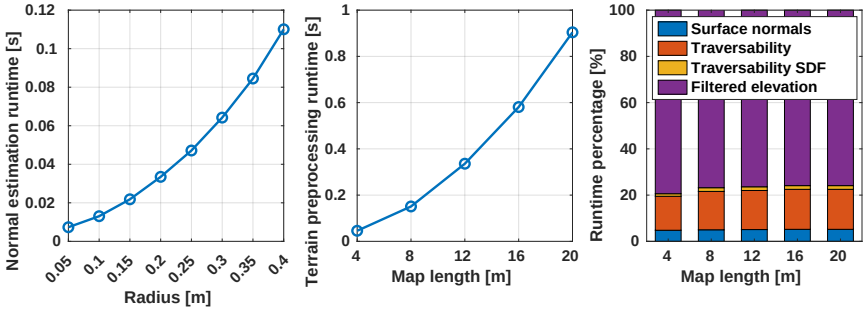
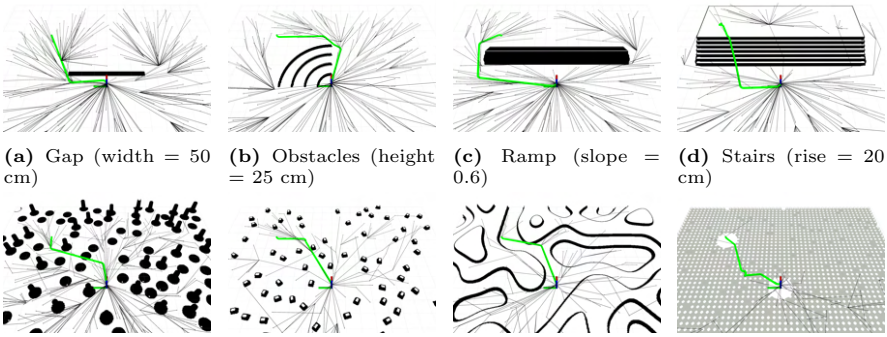


Figure 6.14: Terrain processing runtime analysis with 6 threads. ***Left:*** Normal estimation runtimes for 8×8 meters map (resolution = 3 cm) and different radii. ***Middle:*** Total runtimes for different map sizes at a resolution of 3 cm. ***Right:*** Runtime percentage of each component.

percentage of time spent in each submodule, with the vast majority of time spent estimating wide-neighborhood normals. sdf_3 is excluded from Fig. 6.14 right since it is computed using open-source software; its average runtime is 115 ms for an 8 m $\times$ 8 m map and z coordinate between -0.5 m and 1.9 m.

6.7.2 Legged Robot

We evaluate our planning and control pipeline in simulation first and then validate it on real hardware ANYmal robot Hutter et al., 2016a. For simulation evaluation we create terrains 20m $\times$ 20m in size (3 cm resolution). Start and goal are fixed *SE(2)* states; $(0, 0, 0)$ and $(5, 5, 0)$, respectively. We generate eight different terrain types with three difficulty levels each. Different terrain types are gaps, obstacles, ramps, stairs, mazes, bricks, terrace, and stepping stones; all shown in Fig. 6.15. In each map, the white color denotes the fully traversable area, and the fully untraversable area is shown in a dark color (black or olive). The final path after 2 s planning time is denoted in green. Black lines represent the connections inside the RRT tree. Table 6.2 describes difficulty levels for each terrain. At *Hard* level, some of the features are too difficult for the robot to traverse, and a detour has to be taken (e.g., slope terrain). This highlights the planner feature of negotiating obstacles to shorten the path if possible and take a detour if not.



(a) Gap (width = 50 cm) **(b)** Obstacles (height = 25 cm) **(c)** Ramp (slope = 0.6) **(d)** Stairs (rise = 20 cm)

(e) Maze (# holes = # pillars = 60) **(f)** Bricks (height = 25 cm) **(g)** Terrace (step height = 20 cm) **(h)** Stepping stones (12% removed)

Figure 6.15: ***Top:*** Non-randomized terrains at the highest difficulty level; robot cannot negotiate some terrain features, and a detour must be taken. ***Bottom:*** Four randomized terrain types with the highest difficulty level. We do not know whether a solution exists a priori. If we cannot find a feasible plan within 20 s planning time, the terrain is re-generated until one is found.

6.7.2.1 SBP Success Rate

SBP success rate was tested on a laptop with an Intel i7-10750H@2.60GHz 6-core processor. The success rate is defined as the number of plans that reach the goal over the number of total plans when a feasible path exists. For randomized terrains, we ensure that a feasible path exists by running the planner for 20 seconds. If no path is found, the terrain is re-sampled until a feasible path is found. We consider 6 planning times (0.1, 0.2, 0.5, 1.0, 2.0 and 5.0 seconds), and plan 100 times for every terrain and every planning time. Each randomized terrain is sampled 10 times for a fair evaluation. The result is shown in Tab. 6.3. It reads that the success rate drops with decreasing the planning time; the worst-case scenario has a planning success of 94% for 0.5 seconds of planning time. The planner takes at most 1 second to achieve 100 % success rate on the easy and medium terrains, and 2 seconds for the difficult terrains.

Fig. 6.16 shows two contact schedules discovered by our planner. These two examples highlight that when the terrain is easy, the planner resorts to using well-understood and efficient cyclic gait patterns. On the other hand, in the presence of obstacles, an acyclic contact schedule can emerge and enable the robot to precisely orient its body in anticipation of difficult terrain.

Table 6.2: Terrain features for planner evaluation.

Terrain \ Level	Easy	Medium	Hard	Comment
gap width [m]	0.3	0.4	0.5	*0.5 m is too wide for crossing*
obstacle height [m]	0.15	0.2	0.25	*obstacles 6 cm wide*
ramp slope [deg]	11.3	21.8	31	*max traversable slope:* 25°
stair rise [m]	0.1	0.15	0.2	*stair run 30 cm*
num pillars and holes	20	40	60	*position distributed uniformly*
brick height [m]	0.15	0.2	0.25	*100 bricks that robot can step on*
terrace step height [m]	0	0.1	0.2	*Perlin noise with quantization steps*
% stepping stones removed	4	8	12	*grid with some stones removed at random*

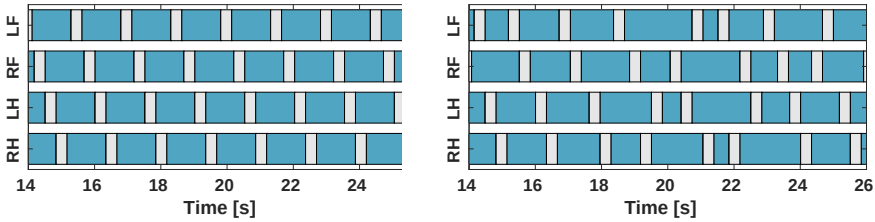


Figure 6.16: Contact schedule visualization (cyan: contact phase, white: swing phase). ***Left:*** planner discovers cyclic contact schedule when walking on the flat ground. ***Right:*** planner uses a highly acyclic gait to walk on the stepping stones.

6.7.2.2 MPC Convergence Rate

The second scenario evaluates the convergence rate of the combined SBP and MPC, defined as the number of times MPC converges for the entire trajectory over the number of plans executed. For evaluation purposes, the MPC has perfect knowledge of robot dynamics, and the controllers do not influence the

Table 6.3: SBPs success rate for ANYmal on different terrain types, difficulty levels, and planning times. The top row shows the minimal planning time where no failures occur (e.g., for the hard difficulty level, with 2 s planning time, the SBP finds a solution in all terrains).

Level / Terrain	Easy → 1.0 s			Medium → 1.0 s			Hard → 2.0 s		
	plan time [s]			plan time [s]			plan time [s]		
	0.1	0.2	0.5	0.1	0.2	0.5	0.1	0.2	0.5
gap	.88	.97	1.0	.81	.97	1.0	.50	.82	1.0
obstacles	.82	.96	1.0	.79	.97	1.0	.69	.96	1.0
ramp	.81	.98	1.0	.81	.96	1.0	.42	.78	.98
stairs	.89	.99	1.0	.83	.97	1.0	.70	.84	.99
maze	.87	.93	1.0	.82	.88	.97	.74	.90	.94
bricks	.83	.92	1.0	.75	.86	1.0	.44	.86	.97
terrace	.81	.95	1.0	.82	.92	1.0	.69	.93	.98
st. stones	.79	.94	.98	.74	.87	.99	.67	.83	.94

success rate. For each non-randomized terrain, ten global plans are executed. For each randomized terrain, we generate two samples for each terrain type and then execute five plans for each sample. The trivial solutions, where the straight-line path on flat ground is enough to connect the start and goal, are removed.

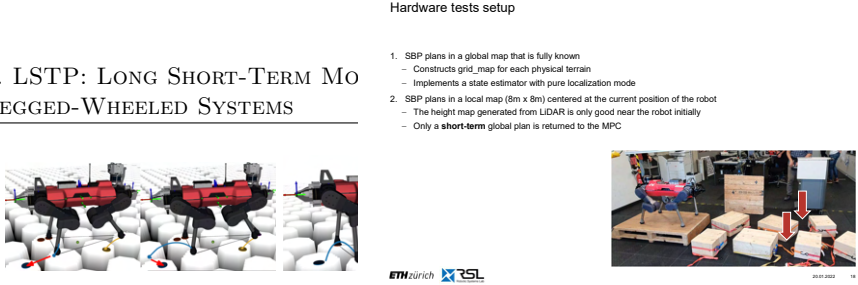
We compare the convergence rate for two different MPC formulations: one with and one without the swing leg joint deviation penalty. We found that the MPC with swing leg penalization always converges, yet the other can diverge on stepping stones with a probability of 30 %. This result demonstrates the benefit of performing whole-body planning in the *initialization step* since the whole-body plans better guide the optimization to the correct local minima. Fig. 6.17a and Fig. 6.17b show how the cost function guides the swing leg to the desired foothold.

6.7.2.3 LSTP Physical Simulation

The proposed planning and control framework was tested in the physical simulation together with the WBC. Since running a physical simulation is computationally expensive, we tested the full simulated stack on each terrain three times without failures.

6.7.2.4 LSTP Hardware Tests

We deploy our planning and control framework on ANYmal. ANYmal has two Robosense RS-Bpearl LiDARs, both in front and back. Bpearls are dome-



(a) No leg deviation penalty **(b)** With leg deviation penalty **(c)** Gap occlusion

Figure 6.17: ***a)*** & ***b)*** Improved optimization convergence through penalizing swing leg deviation. We show an example for the LF leg, whose trajectory planned by the MPC is colored in blue. The reference foothold at the end of the swing phase is shown with the blue disk. The initial solution is shown on the left after the optimization iteration on the right. *a)* Without penalization, the foothold of the optimized trajectory is far away from the reference foothold (distance shown with red arrows). *b)* The optimized swing motion lands smoothly on the reference foothold with penalization. ***c)*** While the LiDAR correctly detects the closer stepping stones, it cannot see holes between the far away stepping stones (marked with red arrows). Hence, the foot placement might be incorrect.

shaped LiDARs with 360°×90° Field of View (FOV). ANYmal has two onboard computers equipped with an Intel i7-8850H@2.60 GHz 6-core processor. The Locomotion PC (LPC) runs MPC at 25 Hz and the WBC at 400 Hz. The controllers are run asynchronously from one another. The Navigation PC (NPC) runs the sampling-based planner and the terrain processing pipeline described in Sec. 6.3.2. Elevation mapping runs at 20 Hz on a Jetson AGX Xavier onboard computer Miki et al., 2022b. As a state estimator we use the fusion of leg kinematics and IMU Bloesch et al., 2013 together with the LiDAR odometry Khattak et al., 2020. All experiments have been performed with a statically stable gait with the maximum recorded velocity of approximately 20 cm/s, which is the same velocity we achieved in the simulation.

Fig. 6.18 depicts the system architecture. The workflow starts from LPC receiving a goal pose, which is then sent to the NPC. Subsequently, the NPC requests a local elevation map from Jetson and performs pre-processing. SBP plans a trajectory using the processed map and sends it to the MPC for tracking.

In our experiments, we let the robot plan in a local map of size 8 m by 8 m centered at the robot's current position. The SBP plans a global path; however, we only send a portion of it to the MPC for tracking. Once the tracking finishes, SBP re-plans with an updated map. This procedure is repeated until the goal is reached. By letting the SBP re-plan, we can avoid issues like the one shown in Fig. 6.17c where the holes are occluded, and as a result, the planner could attempt to place the feet there.

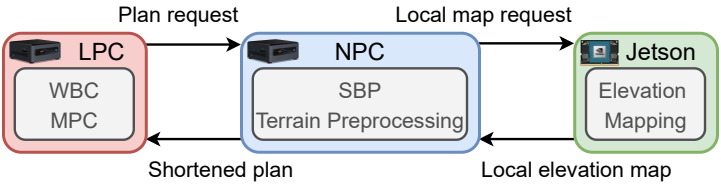


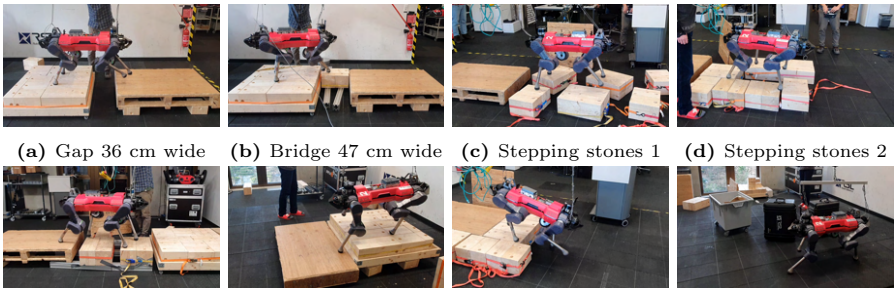
Figure 6.18: ANYmal system architecture. Navigation PC only requests a local height map generated using elevation mapping from Jetson. Once the SBP computes a plan, it is sent to the Locomotion PC, where MPC tracks it.

A range of terrains used for hardware experiments can be found in Fig. 6.19. We could cross 36 cm wide gaps, close to the simulation result of 40 cm and more than the manufacturer's specification of 30 cm. The planner discovers an acyclic contact schedule for the gap-crossing maneuver, similar to the stepping stones scenario. Next, we increase the gap width to 47cm (too wide to cross) and add a narrow bridge. The bridge is too narrow to place all feet on it, so the planner has to coordinate the leg placement precisely. Fig. 6.20 shows the sequence of the robot crossing the bridge. Finally, we also test three different stepping-stone scenarios to showcase the generality of our planner.

Apart from negative obstacles like gaps and stepping stones (see Fig. 6.19), we tested the planner's capability to negotiate steps. The robot was asked to go up and down a step of 20 cm, more than the stair heights recommended in construction, which vary between 14 cm and 19 cm. Experimenting with the maze environment showcases the planner's ability to escape local minima. We constructed a wall around the room corner and left only one narrow passage allowing the robot to get in. Subsequently, the robot was asked to go to the corner from the other side of the wall, where the straight line connection leads to a dead end. With the initialization from the SBP, the robot can walk around and reach the goal. Such a maneuver would be impossible if only MPC planner was used.

6.7.3 Legged-Wheeled Robot

This section shows the planner's capabilities to plan for a legged-wheeled robot with non-steerable wheels. Executing SBPs plans on hardware has been demonstrated in our previous work Bjelonic et al., 2022. For the sake of space, we only show the SBPs evaluation since, in this work, the SBP was generalized to enable base turning and side stepping.



(a) Gap 36 cm wide **(b)** Bridge 47 cm wide **(c)** Stepping stones 1 **(d)** Stepping stones 2

(e) Stepping stones 3 **(f)** Step up, 20 cm **(g)** Step down, 20 cm **(h)** Maze

Figure 6.19: Various terrains tested on the real robot.

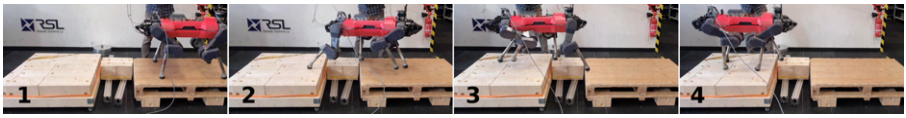


Figure 6.20: Anymal using a bridge to cross a 47 cm wide gap. The bridge is too narrow to place all 4 feet on it. Hence the robot has to figure out a contact schedule and coordinate limb motion to avoid this situation.

The success rate of the SBP is shown in Tab. 6.4. Fig. 6.1 shows an example of a legged wheeled robot navigating stairs using our SBP. The planner is evaluated in the same scenarios as the legged robot. The success rate is comparable to the one for the legged robot shown in Tab. 6.3, but a bit lower. It also takes longer planning times to achieve a 100% success rate for all terrain types. We speculate that this is due to the fact that ANYmal on wheels has a shorter shank compared to ANYmal resulting in shorter reach. The SBP planners use the same tuning for the legged and legged-wheeled robots; we did not need to change any parameters.

6.7.4 HEAP

HEAP is used for experimental verification of the proposed planning and control architecture. We ran the planner on a laptop with AMD Ryzen 9 5900HX CPU 3.3 GHz processor. VMC controller, filtering, wheel velocity control loop, and state estimation run on an onboard computer. Leica iCON iXE3 system provides Global Navigation Satellite System (GNSS) with Real-time Kinematic (RTK) corrections. Two-state information filter Bloesch et al., 2017

Table 6.4: SBPs success rate for ANYmal on wheels across different terrain types, difficulty levels, and planning times. The top row shows the minimal planning time where no failures occur (e.g., for the hard difficulty level, with 5 s planning time, the SBP finds a solution in all terrains).

Level / Terrain	Easy → 2.0 s			Medium → 2.0 s			Hard → 5.0 s		
	plan time [s]			plan time [s]			plan time [s]		
	0.1	0.2	0.5	0.1	0.2	0.5	0.1	0.2	0.5
gap	.68	.91	.99	.73	.96	1.0	.44	.73	.99
obstacles	.63	.91	1.0	.71	.86	1.0	.55	.88	.99
ramp	.76	.91	1.0	.72	.91	1.0	.22	.62	.98
stairs	.82	.96	1.0	.67	.86	1.0	.67	.93	.99
maze	.73	.92	.98	.83	.94	.99	.53	.73	.95
bricks	.76	.90	1.0	.70	.90	1.0	.68	.78	.94
terrace	.78	.97	1.0	.71	.90	.99	.65	.82	.99
st. stones	.64	.83	.95	.52	.75	.92	.50	.74	.91

fuses GNSS measurements with the two Inertial Measurement Unit (IMU) units (from the chassis and the cabin). For more details, please refer to Jud et al., 2021b. In all experiments with HEAP, SBP plans once with 2 s planning time. The plan is then fed to the MPC, which runs between 5 and 20 Hz (depending on the task).

6.7.4.1 LSTP Hardware Tests

We start by testing the combined sampling and optimization planning on hardware. Convergence rate was tested in our previous work Jelavic, Farshidian, and Hutter, 2021, and in this work, we focus on hardware results. Fig. 6.21 shows experiment where HEAP steps over a virtual gap. We did not dig an actual gap in our testing field; however, the height map provided to the planner contains a gap, and hence the stepping-over behavior emerges. Another maneuver where HEAP crosses the virtual bridge is shown in Fig. 6.1. These maneuvers require perceptive planning with both SBP and MPC. Stepping over a gap is challenging for the control system since it has to precisely coordinate the movement of the arm kinematics with the movement of the base such that the contact point does not slip. The control system has to be robust since the wheel speed cannot be precisely controlled, and hence we cannot precisely control the base movement. Without high-frequency local re-planning from the MPC, the gap crossing maneuver would prove challenging due to accumulated drift. The total gap crossing duration is 103 s.

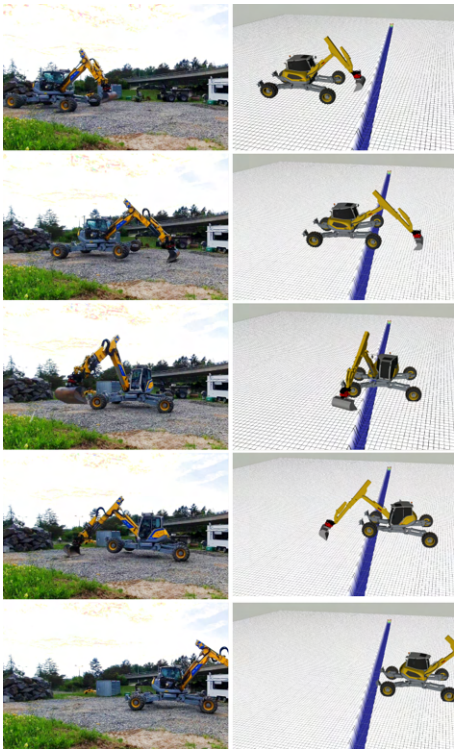
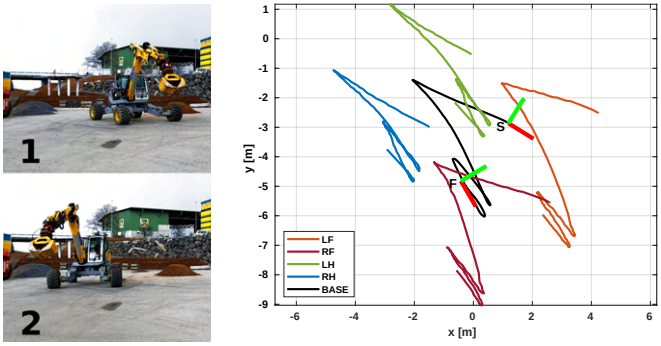


Figure 6.21: ***Top to Bottom:*** HEAP stepping over a virtual gap, the timeline goes from top to bottom. ***Left:*** snapshots of the machine performing the maneuver. ***Right:*** terrain representation as seen by the planner and the controller.

6.7.4.2 Long-Horizon MPC

We also experiment with using solely MPC as a planner and controller to verify MPCs ability to plan over long horizons. In this case, MPC receives a goal pose and a contact schedule without any guiding path. The terrains tested are relatively flat since MPC can fail to converge in the presence of obstacles without good initialization Jelavic, Farshidian, and Hutter, 2021. The prediction horizon is set to 25 s. The start and goal pose, along with the route taken by the base and the wheels, is shown in Fig. 6.22b. The optimization can discover an efficient path that actively utilizes all four steering joints. It involves *crab steering*, a configuration where all four steering joints point in the same direction, resulting in diagonal movement of the base that is not aligned with its heading. The machine is moved 2.53 m sideways with a trajectory that involves four direction changes (discovered by MPC). The motion duration is 46 s, and a contact schedule was externally provided.



(a) Start/Goal **(b)** Crab Steering Paths

Figure 6.22: ***Left:*** Start and end configuration of HEAP performing a driving maneuver. The base is commanded to move sideways, and MPC discovers a motion satisfying the rolling constraint, which involves called *crab steering*. It involves all four wheels pointing in the same direction while the machine moves diagonally. ***Right:*** Base and end-effector positions during the crab steering experiment. The lines denote the path taken by each wheel and the base. The coordinate frames denote the base's starting (S) and finishing (F) position of the base, where red color denotes x axis and the green denotes y axis. For HEAP, x axis is aligned with the direction of travel.

The MPC can also plan stepping motions with a fixed contact schedule. The machine lifts off the legs in the following order: *LH, LF, RH, RF*. Each swing phase lasts 5 s. HEAP is commanded to turn in place multiple times, with approximately 30 degrees yaw increments. Fig. 6.23 shows snapshots of the machine changing its orientation by approximately 108 degrees. A similar scenario is shown in Fig. 6.24 where HEAP steps 6.1 m sideways without driving. Total motion duration is 177 s for the stepping turn and 266 s for the sideways stepping.

It is challenging for the optimization to plan in the presence of nonlinear constraints (e.g., support polygon constraint) over long horizons. Furthermore, the planning times should be short enough to run in real-time. In all cases, the MPC plans with the flat ground assumption. Hence, the control system is also tested since it has to compensate for the unmodelled terrain (the whole field is sloped and bumpy).

6.7.4.3 Whole-Body Controller Terrain Adaptiveness

Lastly, Fig. 6.25 shows an experiment where HEAP is required to adapt to unseen terrain. The planner is given a goal pose ahead of the machine and generates trajectories with flat ground assumption. However, there is a hole

Figure 6.23: HEAP performing a stepping turn in place. The $x-y$ position stays approximately the same, while the azimuth of the chassis changes.

Figure 6.24: HEAP performing a stepping turn and moving laterally. The chassis keeps the azimuth approximately the same as at the beginning of the maneuver.

on one side of the machine (blue arrow) and a hill on the other (red arrow). In the next frame, one can observe that the machine retracted its left leg and extended its right leg to keep the base as leveled as possible. Base and joint

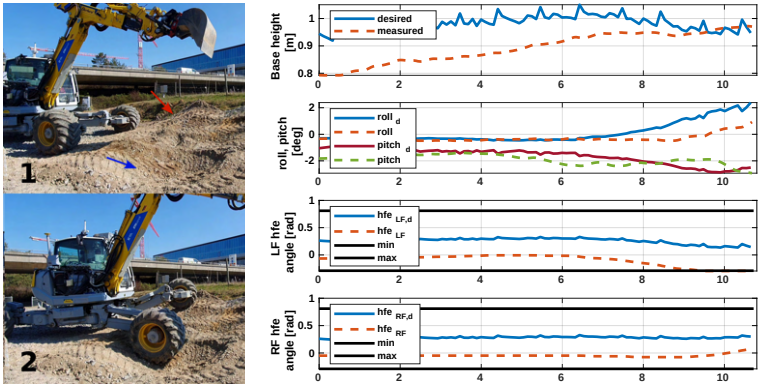


Figure 6.25: ***Top:*** HEAP is commanded forward and there is a hole in front of it (blue arrow) and a hill (red arrow). Since the MPC is planning with a flat terrain assumption, VMC has to coordinate the front legs to keep the chassis leveled. ***Bottom:*** Base and joint tracking. While the base is tracking (with some offset) the MPC commands, HFE joints ignore the MPC references to accommodate for the unmodelled terrain.

tracking during the experiment is also shown in Fig. 6.25. The MPC sends base references to the VMC, which tries to track them. Note that there is some offset since the VMC control has not been tuned with the set point following in mind. The HFE leg joints do not follow the planned values from the MPC

since the terrain map inside the MPC does not match the reality. Note how the left and right HFE joint move in opposite directions since there is a hill on one side and a hole on the other.

6.8 Discussion

6.8.0.1 Initialization step

We chose an optimizing planner like RRT* since the cost allows shaping the robot's behavior (Jelavic, Farshidian, and Hutter, 2021) at the expense of more computation. Since we sample only a three-dimensional subspace, one could use a grid-based method for the base motion planning. We chose RS curves as connections since they elegantly enforce the minimal turning radius for a robot with steerable wheels (HEAP). They are also beneficial for quadrupeds since they naturally limit the robot from walking too much sideways, the least stable way of locomoting. RS curves come with a trade-off: they are far more computationally expensive than straight-line connections (for quadrupeds).

6.8.0.2 Refinement step

While timings are unecessary for quasi-static motions, we compute them because the optimization is formulated as an OCP. Since the MPC only receives and tracks a snippet of the trajectory (e.g., 15 s out of 250 s) in the near future, we trade off some optimality (e.g., cannot speed up the motion) for real-time re-planning; crucial for hardware experiments. We found that using SLQ was possible, however less numerically stable than SQP (especially for HEAP) since SLQ is quite sensitive during the rollout phase.

6.8.0.3 Control

For the torque-controllable legged robot with high bandwidth actuators, we follow the standard approach of using terrain-adaptive whole-body control at the lowest level, which improves robustness. For HEAP, the same method does not work because fine torque control is not possible on all joints. Hence, we run the VMC to be compliant (necessary for hardware experiments) and MPC to track the motion plan. The VMC + kinematic MPC structure avoids a complicated dynamic model inside the MPC, which helps us achieve real-time performance and reduces the amount of tuning for deployment. Optimizing dynamic models for HEAP is numerically challenging because of significant differences in link masses. Moreover, we do not have an accurate dynamic model (e.g., the weight of hoses and oil is not modeled).

6.9 Conclusion

This work presented LSTP, a combination of sampling and optimization-based motion planning for complex legged and legged-wheeled machines. The planner is divided into two stages. The long term *initialization step* is backed by an RRT planner, while the short term *refinement step* backed by the NO. The initialization step computes a contact schedule and a sequence of keyframe configurations used as an initial guess for the optimization. We found that initializing with whole-body configurations can improve optimization convergence in some cases. SBP presented in this work can find initial solutions in less than 1 second, which makes it suitable for real-time re-planning. Fast computation times are achieved by precomputing limb roadmaps offline and then using them to rapidly check collisions and stability at runtime. Lastly, the SBP can handle three robot types: a legged robot, a legged-wheeled robot with non-steerable wheels, and a legged excavator with steerable wheels.

The planner's second stage uses optimization running in an MPC fashion to refine the initial guess from the sampling-based planner. The constant re-planning allows for robustness which is especially important for the HEAP where the hardware does not allow precise control. The proposed two-stage planning approach has been benchmarked on a variety of terrains requiring MPC to stay out of local minima.

We validate the proposed planning and control system on hardware for both quadrupedal robot ANYmal and excavator HEAP. We show that our approach can be executed on hardware in realistic conditions. In particular, for HEAP, we show that the proposed MPC controller is terrain-adaptive and does not require precise terrain models.

6.10 Outlook

The fundamental limitation of our approach is that it only allows for quasi-static motions. The initialization stage of LSTP can only discover static gaits which are not optimized in the second stage. It would be interesting to relax the stability assumption allowing SBP to find more dynamic gaits such as trotting or jumps. Frameworks like Winkler et al., 2018 allow for gait refinement in the second stage; however, one needs to ensure real-time execution and prevent the optimization from getting stuck. Thereby, hierarchical models Li, Frei, and Wensing, 2021 or new efficient solvers for switching time optimization Katayama and Ohtsuka, 2022 could be used.

The other fundamental limitation is the choice of a height map as an environmental representation. Our planner cannot discover contact-rich motions involving overhanging structures (e.g., crawling, ladder climbing), which could be addressed by choosing a different map representation. One big challenge for such confined environments is to devise an efficient strategy for base pose selection.

In addition, the SBPs base pose selection module could be improved. Since a cost function can describe a good base pose, one could use reinforcement learning to train a policy to produce a 3D pose based on the raw terrain observations. This would remove the need for terrain pre-processing, free up computational resources and reduce the amount of tuning. We have already done some preliminary work in this direction.

Lastly, we will implement the SBP running in a separate thread at any time. This way, the MPC could request the new plan before it finishes executing the previous plan, which speeds up maneuver execution.

6.11 Appendix

6.11.1 Terrain Preprocessing

The normal estimation module fits a plane locally for each cell point $\boldsymbol{p}_i = (x_i, y_i)$ inside the grid map. As a result, terrain normal vectors $\boldsymbol{n}_i = [\boldsymbol{n}_i^x \, \boldsymbol{n}_i^y \, 1]^T$ and a height estimate $\tilde{h}_i$ are computed. The tangent plane at point $\boldsymbol{p}_i$ is a function of height h and local coordinates $(dx, dy) = (x - x_i, y - y_i) \in \mathbb{R}^2$,

$$\boldsymbol{n}_i^x dx + \boldsymbol{n}_i^y dy + (h - \tilde{h}_i) = 0. \tag{6.24}$$

Assuming that all the points $\boldsymbol{p}_j$ (height denoted by h_j) within radius R from $\boldsymbol{p}_i$ lie on the plane, we can write Eq. 6.24 in the matrix notation,

$$[\Delta x \; \Delta y \; -1] \cdot \boldsymbol{\theta} = -h_j, \quad \forall j \text{ with } \|\boldsymbol{p}_j - \boldsymbol{p}_i\|_2 < R, \tag{6.25}$$

where $\Delta x = x_j - x_i$, $\Delta y = y_j - y_i$, and $\boldsymbol{\theta} = [\boldsymbol{n}_i^x \; \boldsymbol{n}_i^y \; \tilde{h}_i]^T$ is the vector of parameters. We can then assemble the data matrix $\boldsymbol{D} \in \mathbb{R}^{M \times 3}$ and height vector $\boldsymbol{h} \in \mathbb{R}^M$ and perform the normal estimation by solving the least squares problem:

$$\boldsymbol{\theta}^* = (\boldsymbol{D}^T \boldsymbol{D})^{-1} \boldsymbol{D}^T \boldsymbol{h}. \tag{6.26}$$

Normal estimation is used in computing both surface normals and filtered elevation. The difference is that the former one uses a smaller radius R_s (0.1 m for ANYmal and 0.3 m for HEAP) and does not consider traversability. The latter one performs normal estimation using all traversable points within a larger radius of R_l (0.4 m for ANYmal and 2.5 m for HEAP).

A grid cell (x, y) is categorized as untraversable if on a steep slope or if the surrounding terrain is irregular. Terrain is considered irregular at (x, y) when its height deviates too much from its elevated mean value. The elevated mean h_{em} around point $\boldsymbol{p}_i$ is computed in the local neighbourhood with radius R_l,

$$\mathcal{N}_l(R, \boldsymbol{p}_i) \coloneqq \Big\{ \boldsymbol{p}_j \;\Big|\; \|\boldsymbol{p}_j - \boldsymbol{p}_i\|_2 < R_l \Big\}. \tag{6.27}$$

Define $\mathcal{N}_e(R, \boldsymbol{p}_i)$ as the set of nearby points whose height are above h_{avg}, the average height of $\mathcal{N}_l(R, \boldsymbol{p}_i)$,

$$\mathcal{N}_e(R, \boldsymbol{p}_i) \coloneqq \Big\{ (x, y) \in \mathcal{N}_l(R, \boldsymbol{p}_i) \mid h(x, y) > h_{avg} \Big\}. \tag{6.28}$$

Then we can compute elevated mean h_{em} using equation Eq. 6.29 with the height offset defined in Eq. 6.30.

$$h_{em} = \min\{h_{max},\, h_{avg} + w_o h_o\}, \tag{6.29}$$

$$h_o = \frac{1}{\operatorname{abs} \mathcal{N}_e} \sum_{(x,y) \in \mathcal{N}_e} \Big(h(x, y) - h_{avg} \Big), \tag{6.30}$$

where h_{max} is the maximum height in $\mathcal{N}_l(R, \boldsymbol{p}_i)$ and w_o is a user-defined weight controlling the influence of elevated mean. Elevated mean is motivated by use case on terrains containing negative obstacles such as ditches and stepping stones. We found elevated mean to help correctly assess traversability for terrains like gaps, holes, and stepping stones without a negative effect on others such as stairs, ramps, and pillars. Visualization of the computing elevated mean can be seen in Fig. 6.5b. Upon classifying the traversable regions we compute the sdf_2.

Lastly, the algorithm computes filtered elevation for base pose selection by fitting planes using

$$\mathcal{N}_f(R, \boldsymbol{p}_i) \coloneqq \mathcal{N}_l(R, \boldsymbol{p}_i) \cap \mathcal{T}, \tag{6.31}$$

i.e., all traversable points in $\mathcal{N}_l(R, \boldsymbol{p}_i)$. The rationale behind is that the base of the robot should not adjust itself to the untraversable regions since those are the ones that we want to avoid anyway.

6.11.2 Feasibility Check for Point Foot Robot

6.11.2.1 Full-Support State Connection

The base path connecting the start and the goal of OSM is discretized into N_{seg} segments with $N_{seg}+1$ states. To create a swing phase, we need at least one state to contain a non-grounded leg. Hence we require each swing phase to span across two segments at least, $N_{seg,s} \geq 2$. Larger $N_{seg,s}$ yields finer discretization of swing leg motion in *space* (not in *time*) but decreases the computation speed. The number of segments in OSM cannot be set smaller than $N_{seg,min} \coloneqq K \cdot N_{seg,s}$ with K being the number of the limbs, otherwise we cannot create a feasible contact schedule if all limbs need to have a swing phase. We set N_{seg} slightly larger than $N_{seg,min}$ to allow for longer swing phases if needed.

6.11.2.2 Contact Schedule Computation

We refer to set of discretized states within OSM as $\mathcal{O}$, i.e.

$$\mathcal{O} \coloneqq \Big\{ \boldsymbol{o}_j,\ j = 0, \ldots, K \text{ with } K = N_{seg} \Big\}. \tag{6.32}$$

In Eq. 6.32, $\boldsymbol{o}_0$ denotes the starting state of OSM where $\boldsymbol{o}_K$ denotes the end of it. Their leg configurations are already determined in FSS computation. To find LCB, we search for every state from 1 to $K-1$ for valid roadmap vertices close to footholds at state o_0. If at state j no such vertex is found, we set $j_{lcb} = j-1$. In case the search reaches index $K-1$ it means that the leg can remain in contact during the entire OSM. On the other hand, ECC are computed by searching from index $K-1$ to $N_{seg,s}$ (backwards). If at index j the footholds in $\boldsymbol{o}_K$ cannot be reached, we set $j_{ecc} = j+1$. In case contact can remain for all the searched index, we set $j_{ecc} = N_{seg,s}$ because contact break can at earliest happen at 0.

Given the latest contact break $\boldsymbol{j}_{lcb}[i]$ and earliest contact creation $\boldsymbol{j}_{ecc}[i]$ for the i^{th} swing limb, a feasible contact schedule can be computed by solving the feasibility problem below. The algorithm needs to find indices of states

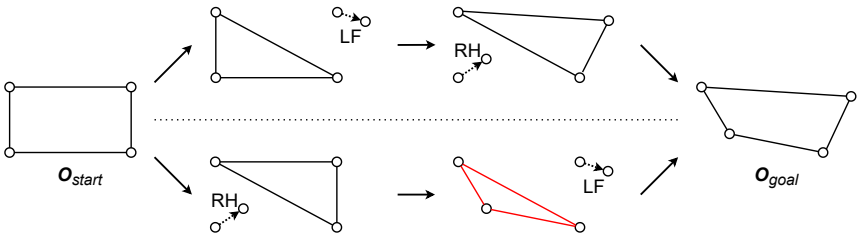


Figure 6.26: Different swing orders generate different support polygons in an OSM. In the upper one, the robot swings first the LF leg and then the RH one, which makes the Support Polygon (SP) large enough. The swing limb order is reversed in the lower option, and this makes the SP very small (highlighted in red) once the LF leg breaks contact.

where contact break $\boldsymbol{j}_{cb}[i]$ and contact creation $\boldsymbol{j}_{cc}[i]$ happen. The feasibility problem below is solved only for the swing legs $\mathcal{SW}$ of OSM.

$$\begin{aligned} \text{find} \quad & \boldsymbol{j}_{cb}[i], \boldsymbol{j}_{cc}[i] \text{ for all } i \in \mathcal{SW} \\ \text{s.t.} \quad & \boldsymbol{j}_{cb}[i] \in \{0, 1, \ldots, \boldsymbol{j}_{lcb}[i]\} \\ & \boldsymbol{j}_{cc}[i] \in \{\boldsymbol{j}_{ecc}[i], \boldsymbol{j}_{ecc}[i]+1, \ldots, N_{seg}\} \\ & \boldsymbol{j}_{cc}[i] - \boldsymbol{j}_{cb}[i] \geq N_{seg,s} \\ & \text{at most one swing leg,} \quad \forall \boldsymbol{o} \in \mathcal{O} \\ & \text{(relaxed) stability constraint,} \quad \forall \boldsymbol{o} \in \mathcal{O}. \end{aligned} \tag{6.33}$$

For a fixed swing limb permutation $\boldsymbol{p}$, it is easy to compute the contact schedule. $\boldsymbol{p}$ tells the swing limb order, e.g. limb at position 0 is the one that swings first. An algorithm for finding a feasible contact schedule is shown in Alg. 5.

As inputs Alg. 5 takes a sequence of OSM states $\mathcal{O}$, ECC and LCB indices $(\boldsymbol{j}_{ecc}, \boldsymbol{j}_{lcb})$ for each leg and a swing limb order $\boldsymbol{p}$. The output of Alg. 5 are the indices of contact breaks $\boldsymbol{j}_{cb}$ and contact creations $\boldsymbol{j}_{cc}$ for all limbs. If a limb does not change its contact state the corresponding ECC and LCB are set to -1 in our implementation. Alg. 5 returns $false$ if no feasible contact schedule can be computed in line 23. The function UNSTABLEINDEX finds the first index t encountered from $\boldsymbol{j}_{cc}[i]$ to $\boldsymbol{j}_{cb}[i]$ for which stability constraints are violated or -1 if no violation. Two examples of the contact plans found with Alg. 5 are shown in Fig. 6.9.

For a quadrupedal robot, the number of different swing orders is at most $4! = 24$ and they result in different support polygons. Fig. 6.26 shows how different swing orders influence on the SPs area. Hence all the combinations

Algorithm 5 Compute contact schedule for a swing order $\boldsymbol{p}$

```
1: function getContactSchedule(p, j_lcb, j_ecc, O)
2:     j_cb ← [−1, −1, −1, −1]^⊤, j_cc ← [−1, −1, −1, −1]^⊤;
3:     for k = p.size() − 1 : 0 do
4:         i ← p[k];
5:         ▷ i is k-th swing limb in p
6:         if k = p.size() − 1 then Commentlast swing limb
7:             j_cc[i] ← N_seg;
8:         else
9:             j_cc[i] ← j_cb[p[k + 1]];
10:            ▷ set cc to the next cb
11:        end if
12:        while true do
13:            j_cb[i] ← min(j_lcb[i], j_cc[i] − N_seg,s);
14:            if j_cc[i] < j_ecc[i] or j_cb[i] < 0 then
15:                return false;
16:            end if
17:            for c = 0 : k − 1 do
18:                ▷ swing limbs prior to i
19:                if j_cb[i] < j_ecc[p[c]] then
20:                    return false;
21:                end if
22:            end for
23:            t ← unstableIndex(p, j_cb[i], j_cc[i], O);
24:            if t = −1 then
25:                break;
26:            else
27:                j_cc[i] ← t − 1;
28:            end if
29:        end while
30:    end for
31:    return {j_cb, j_cc};
32: end function
```

are enumerated and we try to compute a feasible contact schedule using the Alg. 5 (this can be done in parallel). Computation terminates as a feasible contact schedule is found.

6.11.3 Legged-Wheeled Robot Extension

We say that a leg is in the driving mode if it drives forward in longitudinal direction, without any lateral movement. The stepping mode refers to the leg going through a swing phase. The robot enters the stepping mode if its base is turning, side-stepping or when the terrain under the wheel is untraversable. We can identify this situation by looking at the projection of the nominal hip location on the ground. To find LCB for a legged-wheeled robot we first

Algorithm 6 Create one-step motion (OSM)

```
1: function createOSM(o_start, o_goal)
2:     O ← discretize(o_start, o_goal);
3:     ▷ only base poses
4:     if ¬createFullSupportState(o_goal) then
5:         return false;
6:     end if
7:     if ¬isBaseCollisionFree(O) then
8:         return false;
9:     end if
10:    j_lcb ← findLatestContactBreaks(O);
11:    SW ← getSwingLimbs(j_lcb);
12:    j_ecc ← findEarliestContactCreations(O, SW);
13:    for all p ∈ Sym(SW) do
14:        ▷ Sym(SW) are permutations of SW
15:        cs ←getContactSchedule(p, j_lcb, j_ecc, O);
16:        ▷ Alg. 5
17:        if cs ≠ false then
18:            if createSwingMotion(O, cs) then
19:                return O;
20:            end if
21:        end if
22:    end for
23:    return false;
24: end function
```

Algorithm 7 Get feasible OSM lengths

```
1: function feasibleOsmLengths(d, d_total)
2:     ret ← {}, d_to_go ← d_total − d;
3:     if d_to_go < max(L) then
4:         ret.push_back(d_to_go);
5:     end if
6:     for all l ∈ L do
7:         if d_to_go − l > min(L) then
8:             ret.push_back(l);
9:         end if
10:    end for
11:    return ret;
12: end function
```

check what is the leg mode by iterating over the OSM states. Once the leg enters the stepping mode it remains a stepping leg until the end of the OSM. In this case the LCB is searched from the index where the leg switches its state from driving to stepping, same as it was done for the legged robot. If at state j no such vertex is found, we set $j_{lcb} \coloneqq j - 1$. For the driving legs the configuration is found by finding vertices in the roadmap which are close

Algorithm 8 Motion feasibility validation from $\boldsymbol{s}_{start}$ to $\boldsymbol{s}_{end}$

```
1: function isMotionValid(s_start, s_end)
2:     traj ← {};
3:     ▷ state sequence between s_start and s_end
4:     d_total ← distance(s_start, s_end), d ← 0;
5:     while d < d_total do
6:         o_start ← nullptr;
7:         if d = 0 then
8:             o_start ← copyState(s_start);
9:             createFullSupportState(o_start);
10:        else
11:            o_start ← copyState(traj.back());
12:        end if
13:        osm_success ← false;
14:        L ← feasibleOsmLengths(d, d_total);
15:        ▷ Alg. 7
16:        for all l ∈ L do
17:            o_goal ← nullptr;
18:            if d + l = d_total then
19:                o_goal ← copyState(s_end);
20:            else
21:                o_goal ← interp(s_start, s_end, d + l);
22:            end if
23:            O ← createOSM(o_start, o_goal);
24:            ▷ Alg. 6
25:            if O ≠ false then
26:                traj.push_back(O);
27:                d ← d + l;
28:                osm_success ← true;
29:                break;
30:            end if
31:        end for
32:        if ¬osm_success then
33:            return false;
34:        end if
35:    end while
36:    return true;
37: end function
```

to the projection of the nominal hip position on the terrain. For driving legs we can simply establish ground contact same way as during FSS computation (Sec. 6.4.2.2). Similarly, once can find the ECC for the legged-wheeled robot. The computation can be parallelized for different legs.

6.12 Lessons Learned

The last paper introduced LSTP, a planner for legged and legged-wheeled systems based on sampling and optimization. Compared to Paper IV, we extended the planner to handle legged robots with point feet and legged robots with non-steerable wheels. The planner has been benchmarked on various terrains for three different platforms: ANYmal, ANYmal on Wheels, and HEAP. We redesign HEAP's control system and deploy it on hardware. The new control system was able to execute all the maneuvers presented in Paper III with more robustness, and additionally, it can use the arm for locomotion.

The general purpose nonlinear solver IPOPT used in papers II, III, and IV performs the best in robustness and convergence properties; however, it is much slower since it does not exploit the problem structure. We experimented with different strategies inside the MPC and found that DMS is stabler than the SLQ. This is especially true for longer prediction horizons and for HEAP, which has more complicated kinematics than the legged robot. We also found that optimizing offline with TO leaves the optimization more freedom to improve upon the solution from the SBP.

From the deployment point of view, it is clear that today's hydraulic machines are not meant for autonomous operation. While the machine tracks the motion plans, it will only be able to do so somewhat precisely. The actuators need more bandwidth for precise control, and sensing is also limited. For example, the wheel speeds cannot be measured, nor the wheels can be controlled individually. While it was essential to have re-planning capabilities in real-time, the frequency was not as important, and we could execute motions even with low re-planning frequencies (5 Hz-20 Hz).

Parts of the sampling-based planner can profit from data-driven techniques—for example, the base pose selection module. A cost function can describe a good base pose; hence, we could use RL and train a policy outputting a 3D pose based on the raw terrain observations (we did some preliminary work in this direction). Sampling-based planners can profit from learning-based techniques since the algorithm is broken down into sub-modules with clearly defined boundaries. It is less clear how to incorporate a similar strategy into optimization.

7

Conclusion and Outlook

This thesis investigated motion planning and control in the presence of obstacles for legged and legged-wheeled robots. Compared to the related work, we developed a whole-body planning method that can handle systems with many Degrees of Freedoms (DoFs), overcome challenging terrain, compute global plans, and discover a Contact Schedule (CS). The proposed planner removes the need for a multi-level hierarchical planning pipeline, as shown in Fig. 1.1, and it can more effectively utilize the robot's DoFs. The functionality was achieved by combining randomized sampling with optimization. The proposed approaches were demonstrated on a variety of machines of different size and locomotion modality. Besides legged excavators which were the main focus of this thesis, we extended our findings to the other robots, thus demonstrating the generality of the approach. The outcome of this thesis is an increased understanding of how to choose an appropriate technique to solve the Motion Planning Problem (MPP).

7.1 Accomplishments

We showcase a first autonomous deployment of a full-size hydraulic machine for precision harvesting in Chapter 2. Chapter 2 proposes mapping, planning, localization, and control techniques. This is the first time that a complete harvesting system has been investigated and demonstrated on hardware (to the best of our knowledge). During the deployment, Hydraulic Excavator for Autonomous Purpose (HEAP)'s locomotion strategy was to drive and use legs as active suspension. Observing HEAP sometimes getting stuck in wet terrain and comparing it to what human operators can do with legged excavators, we decided to concentrate future efforts on improving the locomotion system.

The autonomous deployment in Chapter 2 resulted in several software contributions and one hardware contribution. To enable mission planning, we built

a prototype of a handheld sensor module for environment scanning. Interestingly, a similar solution emerged in the industry for mobile mapping (Leica, 2022). We also developed a robust algorithm for converting raw point clouds into 2.5D elevation maps deployed in a real forest and several other environments. The implementation is available as open-source[1], and it was used for autonomous embankment excavation (Jud et al., 2021a) and as a ground truth for online elevation mapping (Fankhauser, Bloesch, and Hutter, 2018). The tree detection algorithm used in this thesis is open-source too[2]. The method is purely geometry-based and operates directly on point clouds without extra pre-processing. Planning and path following are also available online[3]. The planning and path following has also been used on the Spacebok robot (Arm et al., 2019), and for other projects on HEAP (Jud et al., 2021a). Lastly, we open-sourced our localization software[4] used for generating ground-truth trajectories. It is used for navigation with ANYmal on Wheels (at the time of writing) and was used in two ETH Robotic Summer Schools (ETH, 2022).

Subsequent papers strived to create a planning and control system with capabilities similar to those of experienced operators. We opted for an optimization-based approach to utilize all DoFs effectively. The proposed optimization planner could perform stepping motions and use the arm similar to how humans use it. Moreover, the planner could discover driving movements too complex for human operators since they require simultaneously coordinating too many DoFs. In the scope of second paper, we derived the robust support polygon constraint, which was indispensable for hardware deployment on HEAP and also used for a legged robot in a student's master thesis.

The second step was to verify the feasibility of the kinematic plans on hardware. We decided to use an optimization-based controller since it allows us to include system constraints. The controller achieves terrain adaptation through Hierarchical Optimizations (HOs) task prioritization. While successfully used for torque-controllable legged robots with electric actuators, HO had not been applied to hydraulic machines. In Chapter 4 we bridge this gap. We could execute whole-body driving and stepping motions on hardware by carefully combining inverse kinematics with full rigid-body inverse dynamics. To our knowledge, this was the first time such maneuvers were shown on a full-sized legged excavator.

[1] `https://github.com/ANYbotics/grid_map/tree/master/grid_map_pcl`
[2] `https://github.com/leggedrobotics/tree_detection`
[3] `https://github.com/leggedrobotics/se2_navigation`
[4] `https://github.com/leggedrobotics/icp_localization`

Chapter 5 builds a sampling-based framework and combines it with an optimization-based planner. We use HEAP as a showcase platform since it has 31 DoF and naive sampling with an Sampling Based Planning (SBP) is too slow. This was the first time we could compute whole-body motions without any initial guess for a system with that many DoFs. Chapter 5 presents an evaluation of what makes the motion planning problem hard for optimization; i.e., it is much harder to deal with the obstacles and complex terrain than a complex robot model. Chapter 5 was the first time a general elevation map was incorporated into the optimization problem for legged-wheeled systems, resulting in a software contribution merged into an open-source package grid_map[5]. We implemented cubic interpolation and cubic convolution, which we found to speed up the optimization convergence for large grid cell sizes.

Finally, in Chapter 6, we extend our findings from previous papers to other robots. The sampling-based planner was extended to handle robots with point feet and legged-wheeled quadrupeds with non-steerable wheels. The proposed SBP was tested on ANYmal on wheels in Bjelonic et al., 2022. The offline optimization from Chapter 3 was replaced with an Model Predictive Control (MPC) based on Direct Multiple Shooting (DMS). This change enabled replanning at frequencies of about 5-50 Hz (depending on the robot and terrain). For HEAP, we revisited the whole-body control system design: we increased robustness, decreased the amount of tuning, and retained terrain adaptation. HEAP could now execute combined stepping and driving motions and use the arm as a supporting limb to step over obstacles. The performance was on par with skilled human operators, thus fulfilling the goal set for ourselves after Chapter 2. At the time of writing, these results have not been achieved by any other excavator/method.

7.2 Outlook

Future research directions are outlined below.

7.2.1 Terrain Processing

One of the bottlenecks in our approach is terrain perception. We rely on traversability for foothold computation, meaning that wrong traversability could result in unstable configurations. For the purpose of this thesis we developed a simple geometric approach in Chapters 5 and 6, however a more

[5] `https://github.com/ANYbotics/grid_map`

principled way of estimating traversability is required (e.g., Frey et al., 2022). A purely geometry-based estimation has limitations; ultimately, we must incorporate visual information. Traversability estimation is an open problem in robotics, and we see a lot of potential for further research on this topic.

7.2.2 Computation

With the advent of cheap Graphics Processing Unit (GPU) computing units like Jetson (NVIDIA, 2022), the planning algorithms should benefit from more computation power. While we parallelized certain portions of the planner, like the terrain processing pipeline, improvements must be made. Randomized sampling has a high potential for parallelization since samples are drawn independently from each other, and in the future parallel sampling planners like Bialkowski, Karaman, and Frazzoli, 2011 will gain importance. On the other hand, the optimization is harder to parallelize since it is iterative, and the descent needs to be done incrementally while re-evaluating gradients. However, once initialized close to optimum, the optimization converges rapidly, and the computation times are low enough for practical applications. With rapid planning times, the whole pipeline (and not just optimization) could be run at high frequency and thus provide more robustness in the system.

7.2.3 Data-Driven Approaches

In the last few years (a Ph.D. time) application of data-driven methods for quadrupedal robots has seen tremendous progress. So far, we have seen an application for control of legged robots (Miki et al., 2022a) showing unprecedented robustness in field deployments. Data-driven approaches can connect well with randomized sampling. We are already witnessing various attempts; Chemin et al., 2021 presents an effort to learn the steer function for connecting new samples to the tree, Yang et al., 2021 shows planning with learned motion costs, and Ichter, Harrison, and Pavone, 2018 tries to learn a good sampling distribution. We experimented with a learned policy for efficient sampling, where we sample in SE(2) space and let the policy determine the full 6D pose. Connecting learning approaches to the SBP planners is an exciting research direction for the future.

Apart from SBP, learning-based approaches can potentially improve the terrain processing pipeline. Works like Chavez-Garcia et al., 2018; Frey et al., 2022 try to learn the traversability function by relying on geometry information. In Chavez-Garcia et al., 2017; Hirose et al., 2018, images are used to

predict terrain properties. Given the success of data-driven methods in other fields like computer vision and their potential to relieve us from handcrafting heuristics, we expect them to become more widespread for traversability estimation.

Lastly, the data-driven control methods seem promising to reduce the amount of tuning when deploying on hardware. This is especially true for hydraulic machines, which exhibit highly non-linear dynamics. The current application of learning-based controllers is limited to excavator arm control (Egli and Hutter, 2022). However, it would be helpful to have a complete whole-body tracking controller tuned entirely from data and without manual labor.

Bibliography

Aceituno-Cabezas, B., Mastalli, C., Dai, H., Focchi, M., Radulescu, A., Caldwell, D. G., Cappelletto, J., Grieco, J. C., Fernández-López, G., and Semini, C. (2017). "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization". *IEEE Robotics and Automation Letters* 3.3, pp. 2531–2538.

API (2021). *API Heavy Machinery Industry, Measurement and Calibration Services*. (Visited on 06/22/2021).

Arm, P., Zenkl, R., Barton, P., Beglinger, L., Dietsche, A., Ferrazzini, L., Hampp, E., Hinder, J., Huber, C., Schaufelberger, D., et al. (2019). "Spacebok: A dynamic legged robot for space exploration". In: *2019 international conference on robotics and automation (ICRA)*. IEEE, pp. 6288–6294.

Arslan, O. and Tsiotras, P. (2013). "Use of relaxation methods in sampling-based algorithms for optimal motion planning". In: *2013 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2421–2428.

Ayrey, E., Fraver, S., Kershaw Jr, J. A., Kenefic, L. S., Hayes, D., Weiskittel, A. R., and Roth, B. E. (2017). "Layer stacking: a novel algorithm for individual forest tree segmentation from LiDAR point clouds". *Canadian Journal of Remote Sensing* 43.1, pp. 16–27.

Babin, P., Dandurand, P., Kubelka, V., Giguère, P., and Pomerleau, F. (2019). "Large-scale 3D Mapping of Sub-arctic Forests". *arXiv preprint arXiv:1904.07814*.

Barraquand, J., Kavraki, L., Latombe, J.-C., Motwani, R., Li, T.-Y., and Raghavan, P. (1997). "A random sampling scheme for path planning". *The International Journal of Robotics Research* 16.6, pp. 759–774.

Bell, B. M. (2012). "CppAD: a package for C++ algorithmic differentiation". *Computational Infrastructure for Operations Research*.

Bellegarda, G. and Byl, K. (2019). "Trajectory optimization for a wheel-legged system for dynamic maneuvers that allow for wheel slip". In: *under review for IEEE Conference on Decision and Control (CDC)*.

Bellicoso, C. D., Bjelonic, M., Wellhausen, L., Holtmann, K., Günther, F., Tranzatto, M., Fankhauser, P., and Hutter, M. (2018a). "Advances in real-world applications for legged robots". *Journal of Field Robotics* 35.8, pp. 1311–1326.

Bellicoso, C. D., Gehring, C., Hwangbo, J., Fankhauser, P., and Hutter, M. (2016). "Perception-less terrain adaptation through whole body control and hierarchical optimization". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 558–564.

Bellicoso, C. D., Jenelten, F., Fankhauser, P., Gehring, C., Hwangbo, J., and Hutter, M. (2017). "Dynamic locomotion and whole-body control for quadrupedal robots". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3359–3365.

Bellicoso, C. D., Jenelten, F., Gehring, C., and Hutter, M. (2018b). "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots". *IEEE Robotics and Automation Letters* 3.3, pp. 2261–2268.

Bellicoso, C. D., Krämer, K., Stäuble, M., Sako, D., Jenelten, F., Bjelonic, M., and Hutter, M. (2019). "Alma-articulated locomotion and manipulation for a torque-controllable robot". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE.

Bellman, R. (1966). "Dynamic programming". *Science* 153.3731, pp. 34–37.

Bialkowski, J., Karaman, S., and Frazzoli, E. (2011). "Massively parallelizing the RRT and the RRT". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 3513–3518.

Bjelonic, M., Bellicoso, C. D., Viragh, Y. de, Sako, D., Tresoldi, F. D., Jenelten, F., and Hutter, M. (2019a). "Keep rollin'—whole-body motion control and planning for wheeled quadrupedal robots". *IEEE Robotics and Automation Letters* 4.2, pp. 2116–2123.

Bjelonic, M., Grandia, R., Geilinger, M., Harley, O., Medeiros, V. S., Pajovic, V., Jelavic, E., Coros, S., and Hutter, M. (2022). "Offline motion libraries and online MPC for advanced mobility skills". *The International Journal of Robotics Research*.

Bjelonic, M., Grandia, R., Harley, O., Galliard, C., Zimmermann, S., and Hutter, M. (2020a). "Whole-body mpc and online gait sequence generation for

wheeled-legged robots". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Bjelonic, M., Sankar, P. K., Bellicoso, C. D., Vallery, H., and Hutter, M. (2019b). "Rolling in the Deep–Hybrid Locomotion for Wheeled-Legged Robots using Online Trajectory Optimization". *arXiv preprint arXiv:1909.07193*.

– (2020b). "Rolling in the Deep–Hybrid Locomotion for Wheeled-Legged Robots using Online Trajectory Optimization". *IEEE Robotics and Automation Letters* 5.2, pp. 3626–3633.

Bloesch, M., Burri, M., Sommer, H., Siegwart, R., and Hutter, M. (2017). "The two-state implicit filter recursive estimation for mobile robots". *IEEE Robotics and Automation Letters* 3.1, pp. 573–580.

Bloesch, M., Hutter, M., Hoepflinger, M. A., Leutenegger, S., Gehring, C., Remy, C. D., and Siegwart, R. (2013). "State estimation for legged robots-consistent fusion of leg kinematics and IMU". *Robotics*.

Bloesch, M., Sommer, H., Laidlow, T., Burri, M., Nuetzi, G., Fankhauser, P., Bellicoso, D., Gehring, C., Leutenegger, S., Hutter, M., et al. (2016). "A primer on the differential calculus of 3d orientations". *arXiv preprint arXiv:1606.05285*.

Bock, H. G. and Plitt, K.-J. (1984). "A multiple shooting algorithm for direct solution of optimal control problems". *IFAC Proceedings Volumes*.

Boston Dynamics (2022a). *Atlas is a research platform designed to push the limits of whole-body mobility.* `https://www.bostondynamics.com/products/spot`. Accessed: 2022-09-29.

– (2022b). *BOSTON DYNAMICS CHOREOGRAPHER DEVELOPER GUIDE.* `https://dev.bostondynamics.com/docs/concepts/choreography/choreographer`. Accessed: 2022-10-09.

– (2022c). *Spot - Automate sensing and inspection, capture limitless data, and explore without boundaries.* `https://www.bostondynamics.com/products/spot`. Accessed: 2022-09-29.

Brakel, P., Bohez, S., Hasenclever, L., Heess, N., and Bousmalis, K. (2021). "Learning coordinated terrain-adaptive locomotion by imitating a centroidal dynamics planner". *arXiv preprint arXiv:2111.00262*.

Brasseur, C., Sherikov, A., Collette, C., Dimitrov, D., and Wieber, P.-B. (2015). "A robust linear MPC approach to online generation of 3D biped walking motion". In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 595–601.

Bretl, T. (2006). "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem". *The International Journal of Robotics Research*.

Bryson, M. (2017). "7.1 PointcloudITD: A software package for individual tree detection and counting". *Deployment and integration of cost-effective high resolution remotely sensed data for the Australian forest industry*, p. 154.

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

Burt, A., Disney, M., and Calders, K. (2019). "Extracting individual trees from lidar point clouds using treeseg". *Methods in Ecology and Evolution* 10.3, pp. 438–445.

Carius, J., Ranftl, R., Farshidian, F., and Hutter, M. (2022). "Constrained stochastic optimal control with learned importance sampling: A path integral approach". *The International Journal of Robotics Research* 41.2, pp. 189–209.

Carius, J., Ranftl, R., Koltun, V., and Hutter, M. (2018a). "Trajectory optimization with implicit hard contacts". *IEEE Robotics and Automation Letters* 3.4, pp. 3316–3323.

– (2019). "Trajectory optimization for legged robots with slipping motions". *IEEE Robotics and Automation Letters* 4.3, pp. 3013–3020.

Carius, J., Wermelinger, M., Rajasekaran, B., Holtmann, K., and Hutter, M. (2018b). "Deployment of an autonomous mobile manipulator at MBZIRC". *Journal of Field Robotics* 35.8, pp. 1342–1357.

Carpentier, J., Saurel, G., Buondonno, G., Mirabel, J., Lamiraux, F., Stasse, O., and Mansard, N. (2019). "The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives". In: *2019 IEEE/SICE International Symposium on System Integration (SII)*. IEEE.

Chavez-Garcia, R. O., Guzzi, J., Gambardella, L. M., and Giusti, A. (2017). "Image classification for ground traversability estimation in robotics". In: *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer, pp. 325–336.

– (2018). "Learning ground traversability from simulations". *IEEE Robotics and Automation letters* 3.3, pp. 1695–1702.

Chemin, J., Fernbach, P., Song, D., Saurel, G., Mansard, N., and Tonneau, S. (2021). "Learning to steer a locomotion contact planner". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4430–4437.

Chen, S. W., Nardari, G. V., Lee, E. S., Qu, C., Liu, X., Romero, R. A. F., and Kumar, V. (2020). "Sloam: Semantic lidar odometry and mapping for forest inventory". *IEEE Robotics and Automation Letters* 5.2, pp. 612–619.

Chen, X., Jiang, K., Zhu, Y., Wang, X., and Yun, T. (2021). "Individual tree crown segmentation directly from uav-borne lidar data using the pointnet of deep learning". *Forests* 12.2, p. 131.

Choudhry, H. and O'Kelly, G. (2018). *Precision forestry: A revolution in the woods*. Retrieved November 27 2019 from https://www.mckinsey.com/industries/paper-forest-products-and-packaging/our-insights/precision-forestry-a-revolution-in-the-woods.

Cordes, F., Babu, A., and Kirchner, F. (2017). "Static force distribution and orientation control for a rover with an actively articulated suspension system". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5219–5224.

Coulter, R. C. (1992). *Implementation of the pure pursuit path tracking algorithm*. Tech. rep. Carnegie-Mellon UNIV Pittsburgh PA Robotics INST.

Cui, J. Q., Lai, S., Dong, X., Liu, P., Chen, B. M., and Lee, T. H. (2014). "Autonomous navigation of UAV in forest". In: *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, pp. 726–733.

Dai, H., Valenzuela, A., and Tedrake, R. (2014). "Whole-body motion planning with centroidal dynamics and full kinematics". In: *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 295–302.

Dai, S., Orton, M., Schaffert, S., Hofmann, A., and Williams, B. (2018). "Improving trajectory optimization using a roadmap framework". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Deits, R. and Tedrake, R. (2014). "Footstep planning on uneven terrain with mixed-integer convex optimization". In: *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 279–286.

Del Prete, A., Tonneau, S., and Mansard, N. (2016). "Fast algorithms to test robust static equilibrium for legged robots". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Di Carlo, J., Wensing, P. M., Katz, B., Bledt, G., and Kim, S. (2018). "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control". In: *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, pp. 1–9.

Diehl, M., Bock, H. G., Diedam, H., and Wieber, P.-B. (2006). "Fast direct multiple shooting algorithms for optimal robot control". In: *Fast motions in biomechanics and robotics*. Springer, pp. 65–93.

Ding, Y., Zhang, M., Li, C., Park, H.-W., and Hauser, K. (2021). "Hybrid sampling/optimization-based planning for agile jumping robots on challenging terrains". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Dolgov, D., Thrun, S., Montemerlo, M., and Diebel, J. (2010). "Path planning for autonomous vehicles in unknown semi-structured environments". *The international journal of robotics research* 29.5, pp. 485–501.

Du, W., Fnadi, M., and Benamar, F. (2020). "Whole-Body Motion Tracking for a Quadruped-on-Wheel Robot via a Compact-Form Controller With Improved Prioritized Optimization". *IEEE Robotics and Automation Letters* 5.2, pp. 516–523.

Egli, P. and Hutter, M. (2022). "A general approach for the automation of hydraulic excavator arms using reinforcement learning". *IEEE Robotics and Automation Letters* 7.2, pp. 5679–5686.

Eidson, J. C. (2006). *Measurement, control, and communication using IEEE 1588*. Springer Science & Business Media.

Englsberger, J., Ott, C., and Albu-Schäffer, A. (2015). "Three-dimensional bipedal walking control based on divergent component of motion". *Ieee transactions on robotics* 31.2, pp. 355–368.

Erez, T., Lowrey, K., Tassa, Y., Kumar, V., Kolev, S., and Todorov, E. (2013). "An integrated system for real-time model predictive control of humanoid robots". In: *2013 13th IEEE-RAS International conference on humanoid robots (Humanoids)*. IEEE, pp. 292–299.

Escande, A., Kheddar, A., and Miossec, S. (2013). "Planning contact points for humanoid robots". *Robotics and Autonomous Systems* 61.5, pp. 428–442.

ETH (2022). *ETH Robotics Summer School*. `https://robotics-summerschool.ethz.ch/`. Accessed: 2022-10-12.

Fahmi, S., Mastalli, C., Focchi, M., and Semini, C. (2019). "Passive Whole-body Control for Quadruped Robots: Experimental Validation over Challenging Terrain". *IEEE Robotics and Automation Letters*.

Fankhauser, P., Bloesch, M., and Hutter, M. (2018). "Probabilistic Terrain Mapping for Mobile Robots with Uncertain Localization". *IEEE Robotics and Automation Letters (RA-L)* 3.4, pp. 3019–3026.

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

Fankhauser, P. and Hutter, M. (2016b). "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation". In: *Robot Operating System (ROS) – The Complete Reference (Volume 1)*. Ed. by A. Koubaa. Springer. Chap. 5.

Farbod, F. (n.d.). *OCS2: Toolbox for Optimal Control of Switched Systems*. URL: `https://github.com/leggedrobotics/ocs2`.

Farshidian, F., Jelavic, E., Satapathy, A., Giftthaler, M., and Buchli, J. (2017a). "Real-time motion planning of legged robots: A model predictive control approach". In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, pp. 577–584.

Farshidian, F., Jelavić, E., Winkler, A. W., and Buchli, J. (2017b). "Robust whole-body motion control of legged robots". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4589–4596.

Farshidian, F., Kamgarpour, M., Pardo, D., and Buchli, J. (2017c). "Sequential linear quadratic optimal control for nonlinear switched systems". *IFAC-PapersOnLine*.

Featherstone, R. (2014). *Rigid body dynamics algorithms*. Springer.

Felis, M. L. (2016). "RBDL: an efficient rigid-body dynamics library using recursive algorithms". *Autonomous Robots*, pp. 1–17.

Felis, M. L. (2017). "RBDL: an efficient rigid-body dynamics library using recursive algorithms". *Autonomous Robots* 41.2, pp. 495–511.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2012). "Distance transforms of sampled functions". *Theory of computing* 8.1, pp. 415–428.

Fernbach, P., Tonneau, S., Stasse, O., Carpentier, J., and Taïx, M. (2020). "C-croc: Continuous and convex resolution of centroidal dynamic trajectories for legged robots in multicontact scenarios". *IEEE Transactions on Robotics* 36.3, pp. 676–691.

Fernbach, P., Tonneau, S., and Taïx, M. (2018). "Croc: Convex resolution of centroidal dynamics trajectories to provide a feasibility criterion for the multi contact planning problem". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Forestry Focus (2018). *Thinning*. Retrieved November 26 2019 from https://www.forestryfocus.ie/growing-forests-3/establishing-forests/thinning/.

Frey, J., Hoeller, D., Khattak, S., and Hutter, M. (2022). "Locomotion Policy Guided Traversability Learning using Volumetric Representations of Complex Environments". *arXiv preprint arXiv:2203.15854*.

Furgale, P. (9, 2014). *Representing Robot Pose The good, the bad, and the ugly*. URL: `http://paulfurgale.info/news/2014/6/9/representing-robot-pose-the-good-the-bad-and-the-ugly` (visited on 02/23/2020).

Furgale, P., Rehder, J., and Siegwart, R. (2013). "Unified temporal and spatial calibration for multi-sensor systems". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1280–1286.

Gaertner, M., Bjelonic, M., Farshidian, F., and Hutter, M. (2021). "Collision-free MPC for legged robots in static and dynamic scenes". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 8266–8272.

Gangapurwala, S., Geisert, M., Orsolino, R., Fallon, M., and Havoutis, I. (2022). "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control". *IEEE Transactions on Robotics*.

Gawel, A., Blum, H., Pankert, J., Krämer, K., Bartolomei, L., Ercan, S., Farshidian, F., Chli, M., Gramazio, F., Siegwart, R., et al. (2019). "A Fully-Integrated Sensing and Control System for High-Accuracy Mobile Robotic Building Construction". *arXiv preprint arXiv:1912.01870*.

Geilinger, M., Poranne, R., Desai, R., Thomaszewski, B., and Coros, S. (2018). "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels". *ACM Transactions on Graphics (TOG)* 37.4, pp. 1–12.

Geisert, M., Yates, T., Orgen, A., Fernbach, P., and Havoutis, I. (2019). "Contact Planning for the ANYmal Quadruped Robot using an Acyclic Reachability-Based Planner". In: *Annual Conference Towards Autonomous Robotic Systems*. Springer, pp. 275–287.

Georgsson, F., Hellström, T., Johansson, T., Prorok, K., Ringdahl, O., and Sandström, U. (2005). *Development of an Autonomous Path Tracking Forest Machine: a status report.*

Giftthaler, M., Farshidian, F., Sandy, T., Stadelmann, L., and Buchli, J. (2017). "Efficient kinematic planning for mobile manipulators with non-holonomic constraints using optimal control". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3411–3417.

Giordano, P. R., Fuchs, M., Albu-Schaffer, A, and Hirzinger, G. (2009). "On the kinematic modeling and control of a mobile platform equipped with steering wheels and movable legs". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, pp. 4080–4087.

Grandia, R., Farshidian, F., Dosovitskiy, A., Ranftl, R., and Hutter, M. (2019a). "Frequency-aware model predictive control". *IEEE Robotics and Automation Letters*.

Grandia, R., Farshidian, F., Ranftl, R., and Hutter, M. (2019b). "Feedback mpc for torque-controlled legged robots". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Grandia, R., Jenelten, F., Yang, S., Farshidian, F., and Hutter, M. (2022). "Perceptive Locomotion through Nonlinear Model Predictive Control". *submitted to IEEE Transactions on Robotics*.

Grandia, R., Taylor, A. J., Ames, A. D., and Hutter, M. (2021). "Multi-layered safety for legged robots via control barrier functions and model predictive control". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Hargraves, C. R. and Paris, S. W. (1987). "Direct trajectory optimization using nonlinear programming and collocation". *Journal of guidance, control, and dynamics* 10.4, pp. 338–342.

Harrison, A. and Newman, P. (2011). "TICSync: Knowing when things happened". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, pp. 356–363.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). "A formal basis for the heuristic determination of minimum cost paths". *IEEE transactions on Systems Science and Cybernetics*.

Hashimoto, K., Hosobata, T., Sugahara, Y., Mikuriya, Y., Sunazuka, H., Kawase, M., Lim, H.-o., and Takanishi, A. (2005). "Realization by biped leg-wheeled robot of biped walking and wheel-driven locomotion". In: *Pro-*

ceedings of the 2005 IEEE International Conference on Robotics and Automation. IEEE, pp. 2970–2975.

Hauser, K., Bretl, T., Latombe, J.-C., Harada, K., and Wilcox, B. (2008). "Motion planning for legged robots on varied terrain". *The International Journal of Robotics Research* 27.11-12, pp. 1325–1349.

Hawkinson, D. (2017). *Forestry Industry*. Retrieved January 22 2021 from https://forestresources.org/resources/woods-to-mill/item/870-addressing-the-labor-shortage-in-the-forestry-industry.

HEAP (Hydraulic Excavator for an Autonomous Purpose) (2018).

Hellström, T., Lärkeryd, P., Nordfjell, T., and Ringdahl, O. (2008). *Autonomous forest machines: Past present and future*.

Hess, W. (2017). *Cartographer Online Documentation*. Retrieved December 06 2019 from https://google-cartographer.readthedocs.io/en/latest/#.

Hess, W., Kohler, D., Rapp, H., and Andor, D. (2016). "Real-time loop closure in 2D LIDAR SLAM". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1271–1278.

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

Holmes, P., Full, R. J., Koditschek, D., and Guckenheimer, J. (2006). "The dynamics of legged locomotion: Models, analyses, and challenges". *SIAM review* 48.2, pp. 207–304.

Holopainen, M., Vastaranta, M., and Hyyppä, J. (2014). "Outlook for the next generation's precision forestry in Finland". *Forests* 5.7, pp. 1682–1694.

Hornung, A. et al. (2014). "Humanoid Robot Navigation in Complex Indoor Environments". PhD thesis. University of Freiburg.

Hutter, M., Gehring, C., Jud, D., Lauber, A., Bellicoso, C. D., Tsounis, V., Hwangbo, J., Bodie, K., Fankhauser, P., Bloesch, M., et al. (2016a). "Anymal-a highly mobile and dynamic quadrupedal robot". In: *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, pp. 38–44.

Hutter, M., Leemann, P., Hottiger, G., Figi, R., Tagmann, S., Rey, G., and Small, G. (2016b). "Force control for active chassis balancing". *IEEE/ASME Transactions on Mechatronics* 22.2, pp. 613–622.

Hutter, M., Leemann, P., Stevsic, S., Michel, A., Jud, D., Hoepflinger, M., Siegwart, R., Figi, R., Caduff, C., Loher, M., et al. (2015). "Towards optimal force distribution for walking excavators". In: *2015 international conference on advanced robotics (ICAR)*. IEEE, pp. 295–301.

Hwan Jeon, J., Karaman, S., and Frazzoli, E. (2011). "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT". In: *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE.

Hyyti, H., Lehtola, V. V., and Visala, A. (2018). "Forestry crane posture estimation with a two-dimensional laser scanner". *Journal of Field Robotics* 35.7, pp. 1025–1049.

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

Ioan, D., Prodan, I., Olaru, S., Stoican, F., and Niculescu, S.-I. (2021). "Mixed-integer programming in motion planning". *Annual Reviews in Control*.

Ishihara, K., Itoh, T. D., and Morimoto, J. (2019). "Full-body optimal control toward versatile and agile behaviors in a humanoid robot". *IEEE Robotics and Automation Letters*.

item (2021). *item Industrietechnik GmbH*. (Visited on 06/24/2021).

Jarrault, P., Grand, C., and Bidaud, P. (2011). "Robust obstacle crossing of a wheel-legged mobile robot using minimax force distribution and self-reconfiguration". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2753–2758.

Jelavic, E., Berdou, Y., Jud, D., Kerscher, S., and Hutter, M. (2020). "Terrain-adaptive planning and control of complex motions for walking excavators". In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, pp. 2684–2691.

Jelavic, E., Farshidian, F., and Hutter, M. (2021). "Combined sampling and optimization based planning for legged-wheeled robots". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 8366–8372.

Jelavic, E., Gonzales, J, and Borrelli, F. (2017). "Autonomous drift parking using a switched control strategy with onboard sensors". *IFAC-PapersOnLine* 50.1, pp. 3714–3719.

Jelavic, E. and Hutter, M. (2019). "Whole-body motion planning for walking excavators". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2292–2299.

Jelavic, E., Jud, D., Egli, P., and Hutter, M. (2022a). "Robotic Precision Harvesting: Mapping, Localization, Planning and Control for a Legged Tree Harvester". *Field Robotics* 2, pp. 1386–1431.

Jelavic, E., Kapgen, T., Kerscher, S., Jud, D., and Hutter, M. (2022b). "Harveri: A Small (Semi-) Autonomous Precision Tree Harvester". In: *Innovation in Forestry Robotics: Research and Industry Adoption, ICRA 2022 IFRRIA Workshop*. ETH Zurich, Institute of Robotics and Intelligent Systems.

Jelavic, E., Nubert, J., and Hutter, M. (2022). "Open3D SLAM: Point Cloud Based Mapping and Localization for Education". In: *Robotic Perception and Mapping: Emerging Techniques, ICRA 2022 Workshop*. ETH Zurich, Robotic Systems Lab, p. 24.

Jelavic, E., Qu, K., Farshidian, F., and Hutter, M. (2022c). "LSTP: Long Short-Term Motion Planning for Legged and Legged-Wheeled Systems". *submitted to IEEE Transactions on Robotics*.

Jenelten, F., Grandia, R., Farshidian, F., and Hutter, M. (2022). "TAMOLS: Terrain-Aware Motion Optimization for Legged Systems". *IEEE Transactions on Robotics*.

Jenelten, F., Miki, T., Vijayan, A. E., Bjelonic, M., and Hutter, M. (2020). "Perceptive locomotion in rough terrain–online foothold optimization". *IEEE Robotics and Automation Letters*.

Johns, R. L., Wermelinger, M., Mascaro, R., Jud, D., Gramazio, F., Kohler, M., Chli, M., and Hutter, M. (2020). "Autonomous dry stone". *Construction Robotics* 4.3, pp. 127–140.

Jud, D., Hurkxkens, I., Girot, C., and Hutter, M. (2021a). "Robotic embankment". *Construction Robotics* 5.2, pp. 101–113.

Jud, D., Kerscher, S., Wermelinger, M., Jelavic, E., Egli, P., Leemann, P., Hottiger, G., and Hutter, M. (2021b). "HEAP-The autonomous walking excavator". *Automation in Construction* 129, p. 103783.

Jud, D., Leemann, P., Kerscher, S., and Hutter, M. (2019). "Autonomous Free-Form Trenching Using a Walking Excavator". *IEEE Robotics and Automation Letters* 4.4, pp. 3208–3215.

Jukka Hämekoski (2016). *Practical and universal tree harvester to increase the possibilities of logging machines*. Retrieved July 28 2021 from https://www.ins-news.com/fi/100/779/1403/Practical-and-universal-tree-harvester-to-increase-the-possibilities-of-logging-machines-.htm.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. (2003). "Biped walking pattern generation by using preview control of zero-moment point". In: *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*. Vol. 2. IEEE, pp. 1620–1626.

Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., and Schaal, S. (2010). "Fast, robust quadruped locomotion over challenging terrain". In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2665–2670.

Kallman, M. and Mataric, M. (2004). "Motion planning using dynamic roadmaps". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. IEEE.

Kalmari, J., Backman, J., and Visala, A. (2014). "Nonlinear model predictive control of hydraulic forestry crane with automatic sway damping". *Computers and Electronics in Agriculture* 109, pp. 36–45.

Karaman, S. and Frazzoli, E. (2011). "Sampling-based algorithms for optimal motion planning". *The international journal of robotics research* 30.7, pp. 846–894.

Kashiri, N., Baccelliere, L., Muratore, L., Laurenzi, A., Ren, Z., Hoffman, E. M., Kamedula, M., Rigano, G. F., Malzahn, J., Cordasco, S., et al. (2019). "CENTAURO: A hybrid locomotion and high power resilient manipulation platform". *IEEE Robotics and Automation Letters*.

Katayama, S. and Ohtsuka, T. (2022). "Whole-body model predictive control with rigid contacts via online switching time optimization". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". *IEEE transactions on Robotics and Automation* 12.4, pp. 566–580.

Keys, R. (1981). "Cubic convolution interpolation for digital image processing". *IEEE transactions on acoustics, speech, and signal processing* 29.6, pp. 1153–1160.

Khattak, S., Nguyen, H., Mascarich, F., Dang, T., and Alexis, K. (2020). "Complementary Multi–Modal Sensor Fusion for Resilient Robot Pose Estimation in Subterranean Environments". In: *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, pp. 1024–1029.

Kim, D., Kwon, Y., and Yoon, S.-E. (2018). "Dancing PRM*: Simultaneous planning of sampling and optimization with configuration free space approximation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Kim, S., Jang, K., Park, S., Lee, Y., Lee, S. Y., and Park, J. (2019). "Whole-body control of non-holonomic mobile manipulator based on hierarchical quadratic programming and continuous task transition". In: *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE, pp. 414–419.

Klamt, T. and Behnke, S. (2017). "Anytime hybrid driving-stepping locomotion planning". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4444–4451.

– (2018). "Planning hybrid driving-stepping locomotion on multiple levels of abstraction". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1695–1702.

Klamt, T., Rodriguez, D., Schwarz, M., Lenz, C., Pavlichenko, D., Droeschel, D., and Behnke, S. (2018). "Supervised autonomous locomotion and manipulation for disaster response with a centaur-like robot". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Koenemann, J., Del Prete, A., Tassa, Y., Todorov, E., Stasse, O., Bennewitz, M., and Mansard, N. (2015). "Whole-body model-predictive control applied to the HRP-2 humanoid". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3346–3351.

Komatsu Corporation (2019). *Harvesters - Product Catalogue*. Retrieved November 26 2019 from https://www.komatsuforest.com/forest-machines/our-wheeled-harvesters.

Konolige, K., Grisetti, G., Kümmerle, R., Burgard, W., Limketkai, B., and Vincent, R. (2010). "Efficient sparse pose adjustment for 2D mapping". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 22–29.

Kuffner, J. J., Kagami, S., Nishiwaki, K., Inaba, M., and Inoue, H. (2002). "Dynamically-stable motion planning for humanoid robots". *Autonomous robots*.

Kuwata, Y., Teo, J., Karaman, S., Fiore, G., Frazzoli, E., and How, J. (2008). "Motion planning in complex environments using closed-loop prediction". In: *AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 7166.

Kweon, I.-S., Hebert, M, Krotkov, E., and Kanade, T (1989). "Terrain mapping for a roving planetary explorer". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 997–1002.

La Hera, P. M., Mettin, U., Westerberg, S., and Shiriaev, A. S. (2009). "Modeling and control of hydraulic rotary actuators used in forestry cranes". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1315–1320.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

LaValle, S. M. and Kuffner Jr, J. J. (2001). "Randomized kinodynamic planning". *The international journal of robotics research*.

LaValle, S. M. et al. (1998). "Rapidly-exploring random trees: A new tool for path planning".

Layeghi, D., Tonneau, S., and Mistry, M. (2022). "Optimal Control via Combined Inference and Numerical Optimization". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3429–3435.

Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. (2020). "Learning quadrupedal locomotion over challenging terrain". *Science robotics*.

Leica (2022). *Leica BLKGO - Mobile Mapping. Simplified.* `https://shop.leica-geosystems.com/leica-blk/blk2go`. Accessed: 2022-10-12.

Leu, J., Wang, M., and Tomizuka, M. (2022). "Long-Horizon Motion Planning via Sampling and Segmented Trajectory Optimization". In: *2022 European Control Conference (ECC)*.

Leu, J., Zhang, G., Sun, L., and Tomizuka, M. (2021). "Efficient robot motion planning via sampling and optimization". In: *2021 American Control Conference (ACC)*. IEEE.

Leven, P. and Hutchinson, S. (2002). "A framework for real-time path planning in changing environments". *The International Journal of Robotics Research*.

Li, B., Li, L., Acarman, T., Shao, Z., and Yue, M. (2021a). "Optimization-based maneuver planning for a tractor-trailer vehicle in a curvy tunnel: A weak reliance on sampling and search". *IEEE Robotics and Automation Letters*.

Li, H., Frei, R. J., and Wensing, P. M. (2021). "Model hierarchy predictive control of robotic systems". *IEEE Robotics and Automation Letters*.

Li, H. and Wensing, P. M. (2020). "Hybrid systems differential dynamic programming for whole-body motion planning of legged robots". *IEEE Robotics and Automation Letters* 5.4, pp. 5448–5455.

Li, L., Long, X., and Gennert, M. A. (2016). "Birrtopt: A combined sampling and optimizing motion planner for humanoid robots". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*.

Li, Q., Nevalainen, P., Peña Queralta, J., Heikkonen, J., and Westerlund, T. (2020). "Localization in unstructured environments: Towards autonomous robots in forests with delaunay triangulation". *Remote Sensing* 12.11, p. 1870.

Li, T., Calandra, R., Pathak, D., Tian, Y., Meier, F., and Rai, A. (2021b). "Planning in learned latent action spaces for generalizable legged locomotion". *IEEE Robotics and Automation Letters*.

Liao, F., Lai, S., Hu, Y., Cui, J., Wang, J. L., Teo, R., and Lin, F. (2016). "3D motion planning for UAVs in GPS-denied unknown forest environment". In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, pp. 246–251.

Lim, J., Lee, I., Shim, I., Jung, H., Joe, H. M., Bae, H., Sim, O., Oh, J., Jung, T., Shin, S., et al. (2017). "Robot system of DRC-HUBO+ and control strategy of team KAIST in DARPA robotics challenge finals". *Journal of Field Robotics* 34.4, pp. 802–829.

Lindroos, O., Ringdahl, O., La Hera, P., Hohnloser, P., and Hellström, T. H. (2015). "Estimating the position of the harvester head–a key step towards the precision forestry of the future?" *Croatian Journal of Forest Engineering: Journal for Theory and Application of Forestry Engineering* 36.2, pp. 147–164.

Magnus Gustavsson (2016). *The radio-controlled bio-energy harvester forest ebeaver*. Retrieved July 28 2021 from https://www.ebeaver.se/En/Default.aspx.

Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G., and Semini, C. (2020). "Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control". *IEEE Transactions on Robotics*.

Medeiros, S, Bjelonic, M., Jelavic, E., Siegwart, R., Meggiolaro, A, Hutter, M., et al. (2019). "Trajectory Optimization for Wheeled Quadrupedal Robots Driving in Challenging Terrain". In: *9th International Symposium on Adaptive Motion of Animals and Machines (AMAM 2019)*.

Medeiros, V. S., Jelavic, E., Bjelonic, M., Siegwart, R., Meggiolaro, M. A., and Hutter, M. (2020). "Trajectory Optimization for Wheeled-Legged Quadrupedal Robots Driving in Challenging Terrain". *IEEE Robotics and Automation Letters* 5.3, pp. 4172–4179.

Melon, O., Geisert, M., Surovik, D., Havoutis, I., and Fallon, M. (2020). "Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Melon, O., Orsolino, R., Surovik, D., Geisert, M., Havoutis, I., and Fallon, M. (2021). "Receding-horizon perceptive trajectory optimization for dynamic legged locomotion with learned initialization". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Menzi Muck (2020). *Menzi Muck Construction Equipment Accessories*. Retrieved March 30 2021 from https://www.menzimuck.com/en/product-groups/menzi-construction-equipment-accessories/.

MenziMuckCom (2011). *Menzi Muck 2006 im Wetten, dass..?* `https://www.youtube.com/watch?v=8Y85nfQuU0M`. Accessed: 2022-10-10.

Miettinen, M., Ohman, M., Visala, A., and Forsman, P. (2007). "Simultaneous localization and mapping for forest harvesters". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, pp. 517–522.

Mikhaylov, M. and Lositskii, I. (2018). "Control and navigation of forest robot". In: *2018 25th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*. IEEE, pp. 1–2.

Miki, T., Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. (2022a). "Learning robust perceptive locomotion for quadrupedal robots in the wild". *Science Robotics*.

Miki, T., Wellhausen, L., Grandia, R., Jenelten, F., Homberger, T., and Hutter, M. (2022b). "Elevation Mapping for Locomotion and Navigation using GPU". *arXiv preprint arXiv:2204.12876*.

Morales, D. O., Westerberg, S., La Hera, P., Mettin, U., Freidovich, L. B., and Shiriaev, A. S. (2011). "Open-loop control experiments on driver assistance for crane forestry machines". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1797–1802.

Mordatch, I., Todorov, E., and Popović, Z. (2012). "Discovery of complex behaviors through contact-invariant optimization". *ACM Transactions on Graphics (TOG)* 31.4, pp. 1–8.

Morita, M., Nishida, T., Arita, Y., Shige-eda, M., Maria, E. di, Gallone, R., and Giannoccaro, N. I. (2018). "Development of robot for 3D measurement of forest environment". *Journal of Robotics and Mechatronics* 30.1, pp. 145–154.

Mowshowitz, A., Tominaga, A., and Hayashi, E. (2018). "Robot navigation in forest management". *Journal of Robotics and Mechatronics* 30.2, pp. 223–230.

Naesset, E. (1997). "Determination of mean tree height of forest stands using airborne laser scanner data". *ISPRS Journal of Photogrammetry and Remote Sensing* 52.2, pp. 49–56.

Neunert, M., Farshidian, F., and Buchli, J. (2016). "Efficient whole-body trajectory optimization using contact constraint relaxation". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 43–48.

Neunert, M., Stäuble, M., Giftthaler, M., Bellicoso, C. D., Carius, J., Gehring, C., Hutter, M., and Buchli, J. (2018). "Whole-body nonlinear model predictive control through contacts for quadrupeds". *IEEE Robotics and Automation Letters* 3.3, pp. 1458–1465.

Nevalainen, P., Li, Q., Melkas, T., Riekki, K., Westerlund, T., and Heikkonen, J. (2020). "Navigation and mapping in forest environment using sparse point clouds". *Remote Sensing* 12.24, p. 4088.

Nocedal, J. and Wright, S. J. (1999). *Numerical optimization*. Springer.

Norby, J. and Johnson, A. M. (2020). "Fast global motion planning for dynamic legged robots". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3829–3836.

NVIDIA (2022). *Advanced AI Embedded Systems*. `https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/`. Accessed: 2022-10-12.

Oliveira, L. F., Moreira, A. P., and Silva, M. F. (2021). "Advances in Forest Robotics: A State-of-the-Art Survey". *Robotics* 10.2, p. 53.

Orin, D. E., Goswami, A., and Lee, S.-H. (2013). "Centroidal dynamics of a humanoid robot". *Autonomous robots* 35.2, pp. 161–176.

Ortiz Morales, D., Westerberg, S., La Hera, P. X., Mettin, U., Freidovich, L., and Shiriaev, A. S. (2014). "Increasing the level of automation in the forestry logging process with crane trajectory planning and control". *Journal of Field Robotics* 31.3, pp. 343–363.

Parker, R., Bayne, K., and Clinton, P. W. (2016). "Robotics in forestry". *NZ Journal of Forestry* 60.4, p. 9.

Pizetta, I. H. B., Brandao, A. S., and Sarcinelli-Filho, M. (2018). "Control and obstacle avoidance for an uav carrying a load in forestal environments". In: *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, pp. 62–67.

Pomerleau, F., Colas, F., Siegwart, R., and Magnenat, S. (2013). "Comparing ICP Variants on Real-World Data Sets". *Autonomous Robots* 34.3, pp. 133–148.

Posa, M., Cantu, C., and Tedrake, R. (2014). "A direct method for trajectory optimization of rigid bodies through contact". *The International Journal of Robotics Research* 33.1, pp. 69–81.

Poulakakis, I. and Grizzle, J. W. (2009). "The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper". *IEEE Transactions on Automatic Control* 54.8, pp. 1779–1793.

Powell, M. J., Cousineau, E. A., and Ames, A. D. (2015). "Model predictive control of underactuated bipedal robotic walking". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 5121–5126.

Rebula, J. R., Neuhaus, P. D., Bonnlander, B. V., Johnson, M. J., and Pratt, J. E. (2007). "A controller for the littledog quadruped walking on rough terrain". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1467–1473.

Reeds, J. and Shepp, L. (1990). "Optimal paths for a car that goes both forwards and backwards". *Pacific journal of mathematics* 145.2, pp. 367–393.

Reid, W., Pérez-Grau, F. J., Göktoğan, A. H., and Sukkarieh, S. (2016). "Actively articulated suspension for a wheel-on-leg rover operating on a martian analog surface". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 5596–5602.

Renner, M., Sweeney, S., and Kubit, J. (2008). *Green Jobs: Towards decent work in a sustainable, low-carbon world*. UNEP.

Rossmann, J., Krahwinkler, P., and Schlette, C. (2010). "Navigation of mobile robots in natural environments: Using sensor fusion in forestry". *J. Syst. Cybern. Inform* 8.3, pp. 67–71.

Rossmann, J., Schluse, M., Schlette, C., Buecken, A., Krahwinkler, P., and Emde, M. (2009). "Realization of a highly accurate mobile robot system for multi purpose precision forestry applications". In: *2009 International Conference on Advanced Robotics*. IEEE, pp. 1–6.

Roßmann, J., Krahwinkler, P., and Bücken, A. (2009). "Mapping and navigation of mobile robots in natural environments". In: *Advances in Robotics Research*. Springer, pp. 43–52.

Rusu, R. B. (2010). "Semantic 3d object maps for everyday manipulation in human living environments". *KI-Künstliche Intelligenz* 24.4, pp. 345–348.

Rusu, R. B. and Cousins, S. (2011). "3d is here: Point cloud library (pcl)". In: *2011 IEEE international conference on robotics and automation*. IEEE, pp. 1–4.

Sardain, P. and Bessonnet, G. (2004). "Forces acting on a biped robot. Center of pressure-zero moment point". *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 34.5, pp. 630–637.

Schwarz, M., Rodehutskors, T., Droeschel, D., Beul, M., Schreiber, M., Araslanov, N., Ivanov, I., Lenz, C., Razlaw, J., Schüller, S., et al. (2017). "NimbRo Rescue: Solving disaster-response tasks with the mobile manipulation robot Momaro". *Journal of Field Robotics* 34.2, pp. 400–425.

Science and Council, T. F. (2007). *HSL. A collection of Fortran codes for large scale scientific computation.*

Sentis, L. and Khatib, O. (2006). "A whole-body control framework for humanoids operating in human environments". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, pp. 2641–2648.

Shan, T. and Englot, B. (2018). "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4758–4765.

Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., and Rus, D. (2020). "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5135–5142.

Sherikov, A., Dimitrov, D., and Wieber, P.-B. (2015). "Balancing a humanoid robot with a prioritized contact force distribution". In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 223–228.

Short, A. and Bandyopadhyay, T. (2017). "Legged motion planning in complex three-dimensional environments". *IEEE Robotics and Automation Letters* 3.1, pp. 29–36.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2010). *Robotics: modelling, planning and control*. Springer Science & Business Media.

Silvere (2017). *We know your trees*. Retrieved November 27 2019 from https://silvere.com/english.

Sleiman, J.-P., Farshidian, F., and Hutter, M. (2021). "Constraint handling in continuous-time ddp-based model predictive control". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 8209–8215.

Sleiman, J.-P., Farshidian, F., Minniti, M. V., and Hutter, M. (2021). "A unified mpc framework for whole-body dynamic locomotion and manipulation". *IEEE Robotics and Automation Letters*.

Smith, J. A., Sharf, I., and Trentini, M. (2006). "PAW: a hybrid wheeled-leg robot". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE.

Song, J. and Sharf, I. (2020). "Time Optimal Motion Planning with ZMP Stability Constraint for Timber Manipulation". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4934–4940.

– (2021). "Stability Constrained Mobile Manipulation Planning on Rough Terrain". *arXiv preprint arXiv:2105.04396*.

Şucan, I. A. and Chitta, S. (2012). "Motion planning with constraints using configuration space approximations". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE.

Sucan, I. A., Moll, M., and Kavraki, L. E. (2012). "The open motion planning library". *IEEE Robotics & Automation Magazine* 19.4, pp. 72–82.

Sun, J., You, Y., Zhao, X., Adiwahono, A. H., and Chew, C. M. (2020). "Towards More Possibilities: Motion Planning and Control for Hybrid Locomotion of Wheeled-Legged Robots". *IEEE Robotics and Automation Letters* 5.2, pp. 3723–3730.

Swedish Forest Agency (2014). *Swedish Statistical Yearbooks of Forestry*. Retrieved November 26 2019 from http://klimatetochskogen.nu/en/information-sources/reports/161-sks2014-forestry-yearbook-2014/.

Tanaka, K., Okamoto, Y., Ishii, H., Kuroiwa, D., Yokoyama, H., Inoue, S., Shi, Q., Okabayashi, S., Sugahara, Y., and Takanishi, A. (2017). "A study on path planning for small mobile robot to move in forest area". In: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, pp. 2167–2172.

Tanaka, T. and Hirose, S. (2008). "Development of leg-wheel hybrid quadruped "AirHopper" design of powerful light-weight leg with wheel". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 3890–3895.

Tominaga, A., Eiji, H., and Mowshowitz, A. (2018). "Development of Navigation System in Field Robot for Forest Management". In: *2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*. IEEE, pp. 1142–1147.

Tonneau, S., Del Prete, A., Pettré, J., Park, C., Manocha, D., and Mansard, N. (2018a). "An efficient acyclic contact planner for multiped robots". *IEEE Transactions on Robotics* 34.3, pp. 586–601.

Tonneau, S., Mansard, N., Park, C., Manocha, D., Multon, F., and Pettré, J. (2018b). "A reachability-based planner for sequences of acyclic contacts in cluttered environments". In: *Robotics Research*. Springer.

Tonneau, S., Song, D., Fernbach, P., Mansard, N., Taïx, M., and Del Prete, A. (2020). "SL1M: Sparse L1-norm Minimization for contact planning on uneven terrain". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Tranzatto, M., Dharmadhikari, M., Bernreiter, L., Camurri, M., Khattak, S., Mascarich, F., Pfreundschuh, P., Wisth, D., Zimmermann, S., Kulkarni, M., et al. (2022a). "Team CERBERUS Wins the DARPA Subterranean Challenge: Technical Overview and Lessons Learned". *arXiv preprint arXiv:2207.04914*.

Tranzatto, M., Mascarich, F., Bernreiter, L., Godinho, C., Camurri, M., Khattak, S., Dang, T., Reijgwart, V., Loeje, J., Wisth, D., et al. (2022b). "Cerberus: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the darpa subterranean challenge". *arXiv preprint arXiv:2201.07067*.

Tremblay, J.-F., Béland, M., Gagnon, R., Pomerleau, F., and Giguère, P. (2020). "Automatic three-dimensional mapping for tree diameter measure-

ments in inventory operations". *Journal of Field Robotics* 37.8, pp. 1328–1346.

Tsounis, V., Alge, M., Lee, J., Farshidian, F., and Hutter, M. (2020). "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning". *IEEE Robotics and Automation Letters*.

Tsubouchi, T., Asano, A., Mochizuki, T., Kondou, S., Shiozawa, K., Matsumoto, M., Tomimura, S., Nakanishi, S., Mochizuki, A., Chiba, Y., et al. (2014). "Forest 3D mapping and tree sizes measurement for forest management based on sensing technology for mobile robots". In: *Field and Service Robotics*. Springer, pp. 357–368.

United Nations Food and Agricultural Organization (2018). *2018 The State of the World's Forests*. Retrieved November 26 2019 from http://www.fao.org/state-of-forests/en/.

Viragh, Y. de, Bjelonic, M., Bellicoso, C. D., Jenelten, F., and Hutter, M. (2019). "Trajectory optimization for wheeled-legged quadrupedal robots using linearized zmp constraints". *IEEE Robotics and Automation Letters* 4.2, pp. 1633–1640.

Von Carlowitz, H. C. and Rohr, J. B. von (1732). *Sylvicultura oeconomica*.

Vukobratović, M. and Borovac, B. (2004). "Zero-moment point—thirty five years of its life". *International journal of humanoid robotics* 1.01, pp. 157–173.

Wächter, A. and Biegler, L. T. (2006). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". *Mathematical programming* 106.1, pp. 25–57.

Wahrburg, A. and Jelavic, E. (2017). *A semi-automatic, interactive tool to identify physical parameters of a mechanical load*.

Wahrburg, A., Jelavic, E., Klose, S., and Listmann, K. D. (2017). "Robust Semi-Automatic Identification of Compliantly Coupled Two-Mass Systems". *IFAC-PapersOnLine* 50.1, pp. 14569–14574.

Wahrburg, A., Klose, S., and Jelavic, E. (2021). *Robust automatic method to identify physical parameters of a mechanical load with integrated reliability indication*. US Patent 10,969,756.

Wellhausen, L. and Hutter, M. (2021). "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.

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

Westerberg, S. (2014). "Semi-automating forestry machines: Motion planning, system integration, and human-machine interaction". PhD thesis. Umeå Universitet.

Wieber, P.-B. (2002). "On the stability of walking systems". In: *Proceedings of the international workshop on humanoid and human friendly robotics*.

Wilcox, B. H. (2012). "ATHLETE: A limbed vehicle for solar system exploration". In: *2012 IEEE Aerospace Conference*. IEEE, pp. 1–9.

Winkler, A. W. (2018a). *Ifopt - A modern, light-weight, Eigen-based C++ interface to Nonlinear Programming solvers Ipopt and Snopt.*

Winkler, A. W., Bellicoso, C. D., Hutter, M., and Buchli, J. (2018). "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization". *IEEE Robotics and Automation Letters* 3.3, pp. 1560–1567.

Winkler, A. W., Farshidian, F., Pardo, D., Neunert, M., and Buchli, J. (2017). "Fast trajectory optimization for legged robots using vertex-based zmp constraints". *IEEE Robotics and Automation Letters* 2.4, pp. 2201–2208.

Winkler, A. W., Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G., and Semini, C. (2015). "Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Winkler, A. (2018b). "Ifopt-A modern, light-weight, Eigen-based C++ interface to Nonlinear Programming solvers Ipopt and Snopt".

Winter, K., Hayes, I. J., and Colvin, R. (2010). "Integrating requirements: the Behavior Tree philosophy". In: *2010 8th IEEE International Conference on Software Engineering and Formal Methods*. IEEE, pp. 41–50.

Wooden, D., Malchano, M., Blankespoor, K., Howardy, A., Rizzi, A. A., and Raibert, M. (2010). "Autonomous navigation for BigDog". In: *2010 IEEE international conference on robotics and automation*. Ieee, pp. 4736–4741.

Xie, C., Berg, J. van den, Patil, S., and Abbeel, P. (2015). "Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4187–4194.

Yang, B., Wellhausen, L., Miki, T., Liu, M., and Hutter, M. (2021). "Real-time optimal navigation planning using learned motion costs". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 9283–9289.

Yue, Y., Yang, C., Senarathne, P., Zhang, J., Wen, M., and Wang, D. (2018). "Online Collaborative 3D Mapping in Forest Environment". *Forest* 2.1.2710, pp. 4–0915.

Zhang, C., Yong, L., Chen, Y., Zhang, S., Ge, L., Wang, S., and Li, W. (2019a). "A Rubber-Tapping Robot Forest Navigation and Information Collection System Based on 2D LiDAR and a Gyroscope". *Sensors* 19.9, p. 2136.

Zhang, J. and Singh, S. (2014). "LOAM: Lidar Odometry and Mapping in Real-time." In: *Robotics: Science and Systems*. Vol. 2, p. 9.

– (2015). "Visual-lidar odometry and mapping: Low-drift, robust, and fast". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2174–2181.

Zhang, W., Wan, P., Wang, T., Cai, S., Chen, Y., Jin, X., and Yan, G. (2019b). "A novel approach for the detection of standing tree stems from plot-level terrestrial laser scanning data". *Remote sensing* 11.2, p. 211.

Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., and Srinivasa, S. S. (2013). "Chomp: Covariant hamiltonian optimization for motion planning". *The International Journal of Robotics Research* 32.9-10, pp. 1164–1193.

Curriculum Vitae

Name	Edo Jelavić
Born	February 27, 1991 in Zagreb, Croatia
Email	edo.jelavic@gmail.com

Education

2009–2013	**MBSc in Electricacl Engineering and Information Technology, University of Zagreb.** Graduated with distinction
2013–2016	**MSc in Electrical Engineering and Information Technology, ETH Zurich.** Tutor: Prof. M. Morari
2015–2016	**Visiting Researcher at the Model Predictive Control Lab, University of California, Berkeley.** Supervisor: Prof. F. Borrelli
2017–2022	**PhD student at the RSL, ETH Zurich, Switzerland.** Supervisors: Prof. M. Hutter, Prof. E. Frazzoli

Work Experience

2012	***Ericsson Nikola Tesla in Zagreb*, summer internship in Software Engineering** • Design and implementation of load balancing Functionalities for EVO controller
2015	**ABB Corporate Reserach Germany, an internship in Robotics & Manufacturing** • Development of reliable parameter identification for compliant multi-mass systems • Developed automatic reliability judgment for identified parameters

2016–2017	**ETH Zurich, research assistant in Agile and Dexterous Robotics Lab** • Co-developed a robust contact force controller for the HyQ robot, based on internal model principle • Implemented a Sequential Linear Quadratic solver for Model Predictive Control • Kistler force sensor integration with an intern under my supervision

Teaching Experience

2009–2013	**University of Zagreb, teaching assistant** • Digital Logics, Electrical Circuits, Electronics 1, Probability Theory and Statistics
2015–2018	**ETH Zurich, teaching assistant** • Linear Systems Theory, Control Systems 1
2020–2022	**ETH Zurich, Lecturer for *Programming for Robotics Introduction to ROS*** • Managed a team of 7 teaching assistants, organized course logistics, and prepared the software exercises for the students
2021–2022	**ETH Zurich, Lecturer in Robotic Summer School SLAM tutorial** • Developed and integrated a new SLAM software, helped with the control stack development, supervised student helpers

Scholarships and Awards

2011,2012	*Josip Lončar*- award for outstanding academic performance (presented to top 1% students)
2013	*Naumann-Etienne Foundation Scholarship*
2013–2015	*ETH Excellence Scholarship and Opportunity Programme (ESOP)*
2016	*Zeno-Karl Schindler Foundation Master Thesis Grant*

List of Publications

For a more detailed and up-to-date list, please visit *Google Scholar*.

First Author Publications

- Jelavic, E., Qu, K., Farshidian, F., and Hutter, M. (2022c). "LSTP: Long Short-Term Motion Planning for Legged and Legged-Wheeled Systems". *submitted to IEEE Transactions on Robotics*

- Jelavic, E., Jud, D., Egli, P., and Hutter, M. (2022a). "Robotic Precision Harvesting: Mapping, Localization, Planning and Control for a Legged Tree Harvester". *Field Robotics* 2, pp. 1386–1431

- Jelavic, E., Farshidian, F., and Hutter, M. (2021). "Combined sampling and optimization based planning for legged-wheeled robots". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 8366–8372

- Jelavic, E., Berdou, Y., Jud, D., Kerscher, S., and Hutter, M. (2020). "Terrain-adaptive planning and control of complex motions for walking excavators". In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, pp. 2684–2691

- Jelavic, E. and Hutter, M. (2019). "Whole-body motion planning for walking excavators". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2292–2299

- Jelavic, E., Gonzales, J, and Borrelli, F. (2017). "Autonomous drift parking using a switched control strategy with onboard sensors". *IFAC-PapersOnLine* 50.1, pp. 3714–3719

- Jelavic, E., Nubert, J., and Hutter, M. (2022). "Open3D SLAM: Point Cloud Based Mapping and Localization for Education". In: *Robotic Perception and Mapping: Emerging Techniques, ICRA 2022 Workshop*. ETH Zurich, Robotic Systems Lab, p. 24

- Jelavic, E., Kapgen, T., Kerscher, S., Jud, D., and Hutter, M. (2022b). "Harveri: A Small (Semi-) Autonomous Precision Tree Harvester". In: *Innovation in Forestry Robotics: Research and Industry Adoption, ICRA 2022 IFRRIA Workshop*. ETH Zurich, Institute of Robotics and Intelligent Systems

Co-Authored Publications

- Farshidian, F., Jelavic, E., Satapathy, A., Giftthaler, M., and Buchli, J. (2017a). "Real-time motion planning of legged robots: A model predictive control approach". In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, pp. 577–584
- Medeiros, V. S., Jelavic, E., Bjelonic, M., Siegwart, R., Meggiolaro, M. A., and Hutter, M. (2020). "Trajectory Optimization for Wheeled-Legged Quadrupedal Robots Driving in Challenging Terrain". *IEEE Robotics and Automation Letters* 5.3, pp. 4172–4179
- Medeiros, S, Bjelonic, M., Jelavic, E., Siegwart, R., Meggiolaro, A, Hutter, M., et al. (2019). "Trajectory Optimization for Wheeled Quadrupedal Robots Driving in Challenging Terrain". In: *9th International Symposium on Adaptive Motion of Animals and Machines (AMAM 2019)*
- Bjelonic, M., Grandia, R., Geilinger, M., Harley, O., Medeiros, V. S., Pajovic, V., Jelavic, E., Coros, S., and Hutter, M. (2022). "Offline motion libraries and online MPC for advanced mobility skills". *The International Journal of Robotics Research*
- Farshidian, F., Jelavić, E., Winkler, A. W., and Buchli, J. (2017b). "Robust whole-body motion control of legged robots". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4589–4596
- Jud, D., Kerscher, S., Wermelinger, M., Jelavic, E., Egli, P., Leemann, P., Hottiger, G., and Hutter, M. (2021b). "HEAP-The autonomous walking excavator". *Automation in Construction* 129, p. 103783
- Wahrburg, A., Jelavic, E., Klose, S., and Listmann, K. D. (2017). "Robust Semi-Automatic Identification of Compliantly Coupled Two-Mass Systems". *IFAC-PapersOnLine* 50.1, pp. 14569–14574

Patents

- Wahrburg, A., Klose, S., and Jelavic, E. (2021). *Robust automatic method to identify physical parameters of a mechanical load with integrated reliability indication.* US Patent 10,969,756
- Wahrburg, A. and Jelavic, E. (2017). *A semi-automatic, interactive tool to identify physical parameters of a mechanical load*

List of Supervised Projects

This section provides a list of student projects that were supervised by the author as part of the doctoral studies.

Master's Theses

Six months full-time research project

- Riessbacher, Jeroen (2022): "Tree Manipulation with an Autonomous Harvester"
- Eyschen, Pol (2022): "Mapping and Augmented Reality in Dynamic Environments"
- Kapgen, Tun (2022) "Teleoperated and Autonomous Tree Harvesting"
- Qu, Kaixian (2021): "Combined sampling and optimization based planning for agile quadrupeds"
- Pajović, Vuk (2021): "Dynamic Motions for Wheeled-legged Robots using Sampling-Based Initializations of Trajectory Optimizations"
- Lorentz, Pierre-Jean (2021): "Using Reinforcement Learning to Perform Acrobatic Maneuvers with a Full Size Skid Steer Loader"
- Cliff, Li Yuan (2020): "Learning Acrobatic Maneuvers for Agile Skid-Steer Vehicles"
- Scott, Timothy (2020): "Visual-Inertial LiDAR Odometry Fusion"
- Berdou, Yannick (2019): "Combined Inverse Kinematics and Whole-Body Inverse Dynamics Control for HEAP"
- Staub, Romeo (2019): "Visual-Inertial Odometry for an Autonomous Walking Excavator"
- Dubath, Benoit (2018): "Design of a learning DC motor controller"
- Pietrasik, Lukasz (2018): "Hierarchical motion planning through sampling and optimization"
- Mayrhofer, Dominic (2018): "Adaptive Control of Industrial Valves"
- Stephens, Brett (2017): "Survey of Control Techniques for Trajectory Following in Autonomous Vehicles"

- Iglesia Valls, Miguel de la (2017): "Learning Model Predictive Control for Autonomous Racing"

Semester Projects

Four months part-time research project

- Fan, Haoying (2021): "Motion-based Actuator Model Identification for RL on ANYexo"
- Doshi, Kiran (2021): "An Acrobatic Manoeuvre for a Skid-Steer Vehicle Using Reinforcement Learning"
- Xing, Chunwei (2021): "Learning Driven Pose Selection for Sampling-Based Planning"
- Jermann, Tizian (2021): "Optimal Mission Planning of a Mobile Manipulator for Construction"
- Abecassis, Lison (2020): "Collision-Free Initializations for Trajectory Optimization using Deep RL"
- Bänninger, Philipp (2020): "Mode Sampling for Path Integral Control"
- Blass, Lukas (2020): "Dynamic Roadmap Planning for Nonholonomic Systems"
- Cathomas, Pascal (2019): "Joint State Estimation Through Inertial Sensing"
- Streiff, Dominic (2019): "Mission Planning for Autonomous Tree Harvester"
- Häberling, Claudia (2019): "Autonomous Exploration with Quadrupedal Robot ANYmal"
- Du, Yipai (2018): "Mapping and Localization for Autonomous Forestry"
- Fiducioso, Marcello (2017): "Modelling and Control of a Self-driving Go-kart"
- Bountouris, Panagiotis(2017): "Modelling and Control of a Self-driving Go-kart"

Studies on Mechatronics

Four months part-time literature research project

- Gnägi, Tobias (2018): "Coverage Path Path Planning for Legged Robots"
- Wilder, Oliver (2017): "Survey of Control Techniques for Trajectory Following in Autonomous Vehicles"
- Bernasconi, Gianmarco (2017): "A Review of the SLAM Problem"

For robotic autonomy, motion planning plays an important role, especially in the presence of obstacles. Overcoming obstacles requires adapting locomotion strategy to the surrounding terrain, a pattern that can be observed in humans and animals. Traditionally, the problem is decomposed into smaller subproblems using simplified models and heuristics, which often cannot capture the coupled dynamics in the system. Hence they often plan motions not fully utilizing the robot's capabilities.

This dissertation extends the locomotion capabilities of legged robots, emphasizing legged excavators. It develops motion planning algorithms utilizing all degrees of freedom for overcoming challenging terrain. We formulate the motion planning problem in a general way for multiple robot types and explore concepts for solving it. To this end, optimization and randomized sampling play a central role in computing global, whole-body motions presented in this thesis.