CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Department of Computer Science

Master's thesis

Free-gait motion planning for hexapod walking robot

David Valouch

August 2021

Supervisor: prof. Ing. Jan Faigl, Ph.D.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Valouch** Jméno: **David** Osobní číslo: **460514**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Plánování pohybu šestinohého kráčejícího robotu s volným vzorem chůze

Název diplomové práce anglicky:

Free-gait motion planning for hexapod walking robot

Pokyny pro vypracování:

- Seznamte se s přístupy plánování pohybu pro kráčející roboty [1,2,3,4] se zaměřením na šestinohé kráčející roboty [5,6] a související omezení a jejich hodnocení [7].
- Navrhněte metodu plánování založenou na metodách prohledávání [8], která bude řešit úlohu přesného plánování pohybu motivovanou
nasazením robotu v jeskynních systémech. Zaměřte navrhované řešení na přístupy, kde není předepsána sekvence výkyvů nohou, takzvané metody s volným vzorem chůze.
- Implementujte a vyhodnoťte navrhované řešení v reprezentativních experimentálních scénářích.

Seznam doporučené literatury:

[1] Hauser, L: Motion Planning for Legged and Humanoid Robots, Ph.D. thesis, Stanford University, 2008.
[2] Perrin, N., Ott, C., Englsberger, J., Stasse, O., Lamiraux, F., & Caldwell, D.G: Continuous Legged Locomotion Planning. IEEE Transactions on Robotics, 33:234-239 (2017).
[3] Belter, D., Labecki, P., Skrzypczynski, P.: Adaptive Motion Planning for Autonomous Rough Terrain Traversal with a Walking Robot. Journal of Field Robotics 33(3):337-370(2016).
[4] Vernaza, P., Likhachev, M., Bhattacharya, S., Chitta, S., A. Kushleyev, A. and Lee, D.D.: Search-based planning for a legged robot over rough terrain, IEEE International Conference on Robotics and Automation (ICRA), 2009, pp. 2380-2387.
[5] Winkler, A. W., Bellicoso, C. D., Hutter, M. and Buchli, J., Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization, IEEE Robotics and Automation Letters, 3(3):1560-1567 (2018).
[6] Čížek, P., Faigl, J., and Masri, D.: Foothold placement planning with a hexapod crawling robot, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 4096-4101.
[7] Belter, D.: Efficient Modeling and Evaluation of Constraints in Path Planning for Multi-Legged Walking Robots. IEEE Access, 7:107845-107862 (2019).
[8] Arain, M.A., Havoutis, I., Semini, C., Buchli, J., Caldwell, D.G.: A comparison of Search-based Planners for a Legged Robot, International Workshop on RobotMotion and Control (RoMoCo), 2013, pp. 104-109.

Jméno a pracoviště vedoucí(ho) diplomové práce:

prof. Ing. Jan Faigl, Ph.D., centrum umělé inteligence FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **18.09.2020** Termín odevzdání diplomové práce: **13.08.2021**

Platnost zadání diplomové práce: **19.02.2022**

prof. Ing. Jan Faigl, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Declaration

I declare that the presented work was developed independently ant that I have listed all sources of the information used within it in accordance with the methodical instructions for observing the ethiclal principles in the preparation of university theses.

Prague, August 13, 2021

. . . . . .
David Valouch

Acknowledgement

I am grateful to prof. Jan Faigl for guidance, motivation, and the time he devoted to reviewing my thesis. I would like to thank all my friends and colleagues for encouragement; notably, Jiří K. for his sense of humor, and Jindřiška D. for her hindsight. And most of all, I thank my family and my girlfriend, I could not finish this work without their support.

Abstract

A free-gait motion planning approach for quasi-static locomotion of a hexapod walking robot on terrains with limited available footholds is proposed. The approach avoids using a prescribed gait pattern allowing an arbitrary sequence of leg swings. Furthermore, it is allowed that some legs do not need to be placed on the terrain for an extended period of time. A finite set of considered footholds is used to define a graph-search domain where pairings of the robot legs and footholds define possible states. However, the connectivity of the search domain is limited by kinematics, stability, and obstacles, which makes the problem challenging. The graph search provides a candidate sequence of stances and intermediate configurations representing plausible steps. The final robot path is found by solving constrained motion planning sub-problems to connect the intermediate configurations into a valid path. The proposed parametrization based on polynomial curves showed to be a suitable choice that provides not only fast convergence of the path optimization but also natural-looking motion. The feasibility of the proposed solver has been empirically validated in challenging scenarios with only a few footholds for a small hexapod walking robot. Moreover, based on the initial implementation of the complex problem solver, improvements have been proposed that significantly reduce the computational requirements making the developed HexaPlanner framework suitable for deployments on multi-legged walking robots.

Keywords: motion planning; legged robot; free-gait

Abstrakt

Diplomová práce představuje novou metodu plánování kvazi-statického pohybu kráčejících robotů s volným vzorem chůze v terénech s omezenou oporou nohou. Navrhovaný přístup nevyužívá předepsaného vzoru chůze, což umožňuje využít libovolnou posloupnost kroků. Vyvinutý plánovač pohybu dále umožňuje, aby některé nohy mohly být delší dobu mimo opěry, čímž je snížena závislost na umístění nohou v terénu. K definování domény pro hledání požadované sekvence je vytvořena grafová reprezentace, ve které dvojice nohy-robotu a opěry-nohou definují možné stavy. Konstrukce grafu reprezentující prohledávanou doménu uvažuje konečnou množinu možných opěrných bodů. Konektivita prohledávané domény je však limitována omezujícími podmínkami kinematiky, stability a překážkami, což činí problém velmi náročným. Navržené řešení prohledávání grafu tvoří cestu, která odpovídá kandidátní sekvenci postojů a mezilehlých konfigurací robotu představujících pravděpodobně uskutečnitelné kroky. Plán pohybu robotu je z nalezené sekvence získán řešením dílčích plánovacích podproblémů propojením mezilehlých konfigurací s uvážením omezení pohybu. Navrhovaná parametrizace výsledné cesty robotu založená na polynomiálních křivkách se ukázala jako vhodná volba, která poskytuje nejen rychlou konvergenci optimalizace cesty, ale také přirozeně vypadající pohyb. Schůdnost navrhovaného řešiče byla empiricky ověřena v náročných plánovacích scénářích s malým počtem opěrných bodů pro malý šestinohý kráčející robot. Kromě toho byla na základě počáteční implementace řešení komplexního problému navržena vylepšení, která významně snižují výpočetní nároky. Vyvinutý softwarový rámec HexaPlanner tak představuje vhodné řešení pro nasazení na vícenohých kráčejících robotech.

Klíčová slova: plánování pohybu; kráčející robot; volný vzor chůze

Used Abbreviations

COG	Center of Gravity
COM	Center of Mass
LSP	Least Squares Pose
PSO	Particle Swarm Optimization
SDF	Signed Distance Field
SOCP	Second Order Cone Program
SP	Support Polygon
w.r.t	with respect to

List of Used Symbols

$\mathcal{C}$ ..... Configuration space
$\mathcal{C}_{\text{feasible}}$ ..... Feasible subset of a configuration space
$\mathcal{P}$ ..... Planning task
σ ..... Stance
Σ ..... Set of all stances
$\mathcal{C}_\sigma$ ..... Contact manifold
$\mathcal{F}_\sigma$ ..... Stance region
$\mathcal{C}_\Sigma$ ..... stance-configuration space
q_σ ..... stance-configuration
$\mathcal{C}_{\Sigma,\text{feasible}}$ ..... Feasible joint configuration space
$f_{\mathcal{C}_\sigma}$ ..... Contact constraint function
$g_{\mathcal{S}_\sigma}$ ..... Support polygon constraint function
$g_{\boldsymbol{\theta}}$ ..... Joint constraint function
$g_{\text{SDF}\,\sigma}$ ..... Terrain collision constraint function
G_Σ ..... Stance graph

Contents

List of Figures

List of Tables

Chapter 1
Introduction

As early as in the 1980s, scientists have begun suggesting the use of robots for search-and-rescue missions [1], as mobile robots promise to navigate environments too dangerous for humans. At that time, robotics has not been advanced enough for such missions. In 1995, two major disasters – the Oklahoma City bombing and the Hanshin-Awaji earthquake in Japan – motivated advancement and further research effort towards rescue robots. In both disasters, the rescuers struggled to find survivors trapped under collapsed buildings.

The first recorded use of robots in a disaster relieve is dated after the 2001 terrorist attacks on the World Trade Center [1], where some areas were unbearably hot for human rescuers due to fire. A robotic rescue team from Washington, D.C. was led by Lt. Col. John Blitch of the U.S. Army, who founded the Center for Robot-Assisted Search and Rescue (CRASAR) [2]. The additional robotic team at the scene was lead by Dr. Robin Murphy from the University of South Florida, now the director of Humanitarian Robotics and AI Laboratory [3], formerly CRASAR. The robots deployed were tracked robots such as shown in Fig. 1a. Although the robots did not find any survivors, the real-world experience provided invaluable feedback for the researchers. In later years, Dr. Murphy helped to deploy robots in response to nearly 30 disasters [1].

(a) New York 2001[1] (b) Mexico 2017[2] (c) DARPA SubT Challenge 2020 [4]

Figure 1: Robots deployed in real-world or simulated rescue scenario: (left) A tracked robot used in the aftermath of terrorist attacks in New York; (middle) Snake-like robot used to search rubble after earthquakes in Mexico; (right) Tracked and hexapod robot deployed in the Tunnel circuit of the DARPA Subterranean Challenge.

Another example comes from the 2017 earthquakes in Mexico, where roboticists from the Biorobotics Laboratory at the Carnegie Mellon University (CMU) brought their biologically inspired snake-like robots to the scene Fig. 1b. Deployed robots confirmed that nobody was trapped under the debris without the need to move large parts of the rubble or endangering human rescuers.

In 2018–2021, the Defense Advanced Research Projects Agency (DARPA) of the United States Department of Defense organized the DARPA Subterranean Challenge (SubT), a competition of robotics teams worldwide. In the competition, teams deploy their robots in simulated underground search-and-rescue scenarios. The goal is to find *artifacts* such as survivors, gas-leaks, tools, backpacks, gas leaks scattered in an underground space, classify them, and report their location to score. Points are awarded

[1]Photo adopted form https://discovermagazine.com/technology/after-disaster-strikes-a-robot-might-save-your-life
[2]Photo adopted from https://cmu.edu/news/stories/archives/2017/september/snakebot-mexico.html

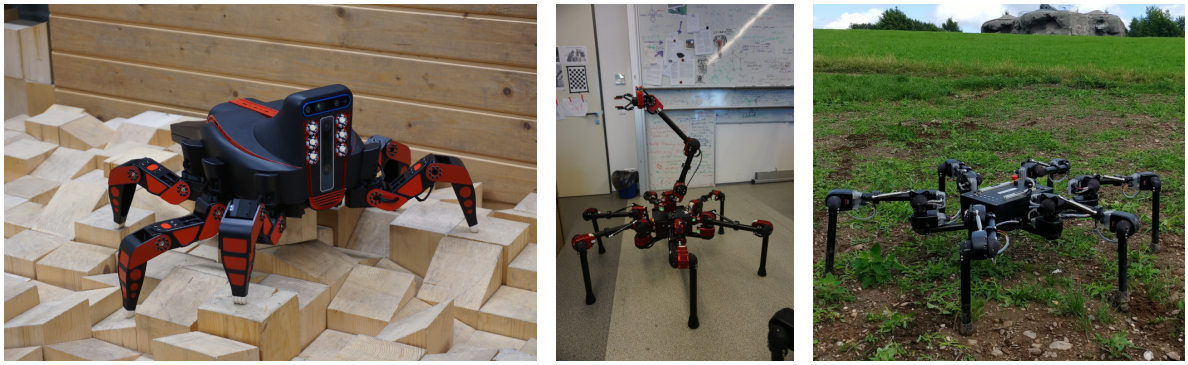
Figure 2: Example of the terrain traversed by hexapod walking robot and tracked robot. There are visible footprints of the tracked robot, while the legged robot does not make any significant marks when crawling over the sloped terrain.

for all correctly identified artifacts whose position is within five-meter precision from the ground truth location. One of the main difficulties of navigating an underground environment is the lack of wireless communication. Since underground environments are communication restricted, the robots need to be autonomous to a large extent. Autonomy improves the speed at which a robot can progress, and it also reduces requirements on the operator in potentially high-stress conditions. Indeed, the robotic team responding to the 9/11 attacks reported difficulties operating the robots in the harsh environment [1].

Regarding types of ground mobile robots, we can distinguish wheeled, tracked, and legged robots. Wheeled and tracked robots can be considered simpler to design, and control than legged robots because wheeled and tracked robots are more commonly used in practical applications nowadays. On the other hand, legged robots tend to have more complex construction and more degrees of freedom; thus, they are challenging to control. However, a legged robot can potentially traverse rougher terrain than a comparably sized wheeled or tracked robot. Also, legged robots are potentially less likely to disturb the environment, e.g., see Fig. 2, which may be very desirable in search-and-rescue scenarios, where the underground cave system or rubble of a collapsed building may be unstable. Notably, the hexapod walking robots provide increased stability over their competitors such as quadruped platforms, e.g., Spot [5] and ANYmal [6], or biped/humanoid robots such as Atlas [7] or HRP2 [8].

This thesis focuses on improving the autonomous traversability capabilities of legged robots motivated by hexapod walking platforms used in the Computational Robotics Laboratory of the Artificial Intelligence Center at the FEE, CTU in Prague. More specifically, we target to develop autonomous planning methods for our experimental six-legged (*hexapod*) walking robot SCARAB II and hexapod walking platforms Daisy and Lily from the HEBI Robotics, see the robots in Fig. 3. We aim to create a motion planner that balances simplifications of the original problem and computational complexity to perform robust online planning using the robot's onboard hardware. In general, such a challenging goal can be recognized to be too ambitious. Therefore, the thesis can be considered an attempt to implement any tractable solution to the planning task and develop a general code base that can be further improved and expanded upon.

The proposed idea builds on the previous work that is overviewed in the following section. Then, the thesis is structured as follows. The theoretical background is introduced in Chapter 2. An overview of the related work is presented in Chapter 3. The addressed problem is formally defined in Chapter 4



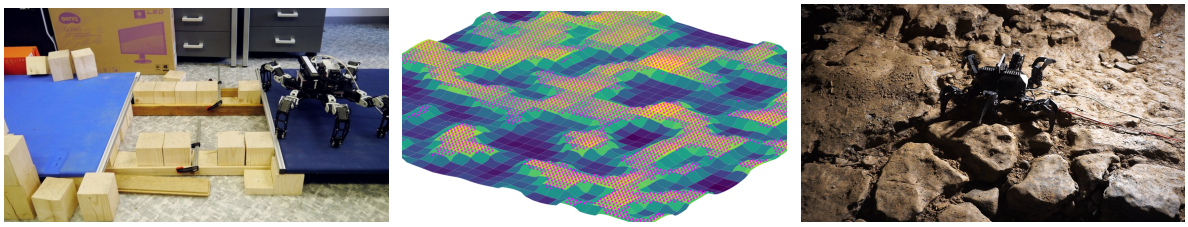
(a) SCARAB II (b) Daisy (c) Lily

Figure 3: Legged robotic platforms: (left) SCARAB II is the second generation of small hexapod robot developed at the Computational Robotics Laboratory, the Czech Technical University; (middle) Daisy a robotic kit by HEBI the Robotics. It is a large hexapod robot primarily intended for indoor research; (right) Lily is a waterproof version of Daisy.

together with the used notation. The proposed solution is described in Chapter 5 followed by an overview of the realized implementation in Chapter 6. The evaluation results of the developed solution are presented in Chapter 7. Possible future research directions and improvements of the proposed solution are discussed in Chapter 8. Finally, concluding remarks are dedicated in Chapter 9 with the remarks on the possible future work.

1.1 On Overview of the Previous Work

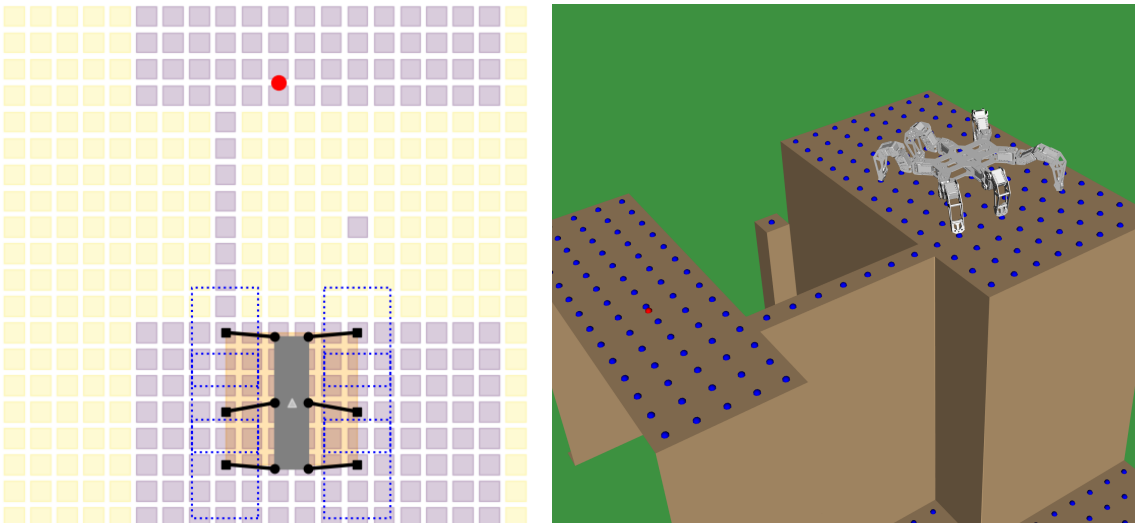
The problem of legged locomotion planning can be represented by the *Obstacle Race* scenario inspired by crossing a river stream using stones sticking from the water. A laboratory setup of such a scenario used in [9] is visualized in Fig. 4a. Although it might seem like a very specific scenario, in highly



(a) Obstacle Race (b) Generated Cave Floor (c) Real cave floor

Figure 4: (left) Photo of the original 'obstacle race' scenarios; (middle) Virtual cave floor scenario with filtered footholds; (right) Hexapod robot crawling on a rocky terrain in the Bull Rock Cave in Moravia, Czech Republic.

structured terrain, there can only be relatively few and sparse safe footholds for the robot to use, e.g., an uneven cave floor where we need to avoid inclined or sunken footholds, see Figs. 4b and 4c. In [9], the problem is addressed by Diar Masri, Petr Čížek, and Jan Faigl, further described in Section 3.4.2. However, the drawback of their approach is using a regular gait, for which the robot moves from one configuration with all legs on the terrain to the next configuration also with all legs on the terrain. The employed simplification needs to use a two-dimensional terrain abstraction that limits the method to relatively simple scenarios such as shown in Fig. 4a.



(a) 2D environment (b) 3D environment

Figure 5: Proposed more challenging *obstacle race* scenario than used in [9]; (left) simplified 2D environment; (right) 3D environment.

We are motivated to address the drawbacks of [9] and the very first idea for a revised solution of the obstacle race is to employ a graph-search algorithm, where the graph vertices would represent particular legs placement on the terrain footholds. Besides, it is allowed that a leg is not placed, and the leg foot tip can be in the air. The initial concept has been inspired by the planning approach for humanoid robots [10]. However, the goal is not to force the robot to move between configurations with all legs on the ground as in [9], but to allow a leg not to be placed at some foothold for an extended period if there is no viable foothold for it. A *tougher obstacle race* scenario such as shown in Fig. 5b has been devised to evaluate the desired behavior. Instead of two beams with holes as in the original *obstacle race* as of Fig. 4a, the new scenario features one full beam on one side and a single stepping pad in place of the second beam. A proof-of-concept implementation in the Julia programming language has been quickly produced for a planning task in a simplified 2D environment with the assumed finite set of relatively sparse footholds, see in Fig. 5a. Even though the first developed solution has been promising, it becomes apparent that the combinatorial complexity of the graph representation would be computationally challenging. Nevertheless, a notable finding has been observed that such a method runs relatively fast in scenarios with a limited number of footholds. Hence, the goal of this thesis become to develop gait-free locomotion planning that will form a code-based allowing to solving scenarios similar to *obstacle race*. We follow the work of K. Hauser, who elaborates in-depth the legged locomotion planning in [11], here overviewed in Section 3.4.3. Before describing the related work and proposed method, the following chapter introduced the basic theoretical background of the employed techniques and representations of the search space and trajectories.

Chapter 2
Theoretic Background

In this chapter, a brief description of the basic theory used in the thesis is provided. First, the concept of the configuration space and degrees of freedom is introduced. Further, the kinematics of the robot composed of serial kinematic chains is formalized. Finally, a brief description of the Bézier curve parametrization is presented because it is used in the trajectory parametrization in the proposed approach presented in Section 5.4.

2.1 Mechanical System Specification

- **Degrees of Freedom (DOF)** – is the minimal number of independent parameters needed to fully describe the state of a mechanical system.
- **Controllable Degree of Freedom (CDOF)** – is the number of DOF that can be controlled, i.e., the robot actuators.

According to the ratio of the CDOF and the total DOF (TDOF) on the robot, we can distinguish robots that are

- **Holonomic** – CDOF = TDOF;
- **Nonholonomic** – CDOF $<$ TDOF;
- **Redundant** – CDOF $>$ TDOF.

A legged robot is fully specified by the position of its body (6 DOF) that is composed of

- **Translation** – *x, y, z* coordinates,
- **Rotation** – *yaw, pitch, roll* angles,

and by the set position of all its actuated joints. For rotational joints, the position of the actuator can be described by a vector of the angle values $\boldsymbol{\theta}$, by which individual joints are rotated. The actuators represent the Controllable DOF, and thus $\dim(\boldsymbol{\theta})$ = CDOF; thus, legged robots are nonholonomic.

2.2 Configuration Space and Planning

Configuration space $\mathcal{C}$ of a robot is a set of configurations, where a configuration is a complete description of the robot's state, including its position in space and all set position of all controllable degrees of freedom. *Feasible subset* of the configuration space $\mathcal{C}_{\text{feasible}} \subseteq \mathcal{C}$ is the set of all configurations that are feasible under the physical constraints. In the literature, it can be found that the set of collision-free configurations $\mathcal{C}_{\text{free}}$ is considered as the feasible subset. However, for this thesis, it is useful to explicitly enforce more constraints than just collisions. It holds that $\mathcal{C}_{\text{feasible}} \subseteq \mathcal{C}_{\text{free}}$, therefore we distinguish these two sets.

The constraints can be expressed by an *equality constraint function* $f : \mathcal{C} \to \mathbb{R}^k$ and an *inequality constraint function* $g : \mathcal{C} \to \mathbb{R}^m$, such that $f(q) = \mathbf{0}$ and $g(q) \leq \mathbf{0}$ *iff* $q \in \mathcal{C}_{\text{feasible}}$. The equality constraint decreases the topological dimensions of $\mathcal{C}_{\text{feasible}}$. The set $\mathcal{C}_{\text{feasible}}$ forms a submanifold of $\mathcal{C}$. More specifically, the dimension is reduced by the number of equalities [12] as

$$\dim(\mathcal{C}_{\text{feasible}}) \;=\; \dim(\mathcal{C}) - k. \tag{1}$$

In planning tasks, there might be multiple constraint functions. In general, the constraint functions depend on some discrete part of the configuration space representation. The submanifolds of the configuration space defined by the individual constraint functions are called *modes* [12]. Hence, a planning problem with multiple modes is called *multi-modal* planning problem.

Legged locomotion planning is a multi-modal planning problem, and the legged robots interact with the terrain using their foot tips. More generally, we refer to the parts of the robot's body meant for interaction with the terrain as *effectors*. The modes correspond to the placement of the robot effectors on the terrain, and we say these effectors are *fixed* by the mode. The modes in the legged locomotion are called *stances* [13]. The equality constraint function describes the kinematic constraints for the effectors fixed by the stance.

In addition to discrete parameters, the constraint function can also be parametrized by continuous parameters. In such a case, we can talk about continuous *families* of modes [14]. The family contains all modes defined by the same combination of discrete parameters. An example for legged locomotion can be climbing a ladder, where the discrete component defining the mode family is a mapping of a specific leg onto a specific rung of the ladder, and the continuous parameter is the position on the rung. The continuous parameters are fixed within the mode family. In order to change the continuous parameters within a mode family, it is needed to leave the family and enter in with a new parameter value. Continuing with the ladder analogy, the problem is to change the leg position on the ladder rung; it is needed to go off the rung and grab it at a different position.

In path planning, we can define a *path* as a continuous mapping from the closed interval $[0, 1]$ to $\mathcal{C}_{\text{feasible}}$ as

$$\pi : [0, 1] \to \mathcal{C}_{\text{feasible}}. \tag{2}$$

We can further distinguish a *trajectory* as time-parametrized path

$$q : [t_0, t_f] \to \mathcal{C}_{\text{feasible}}. \tag{3}$$

Path planning is a problem to find a path or trajectory satisfying the respective constraints. On the other hand, motion planning is to find a sequence of actions to execute the path. Since in the presented approach, we need to determine individual position values of the robot actuators that represent configurations in $\mathcal{C}_{\text{feasible}}$, the path in configuration space provides a solution of the motion planning.

The herein in addressed motion planning task $\mathcal{P}$ is given by an initial configuration $q_0 \in \mathcal{C}_{\text{feasible}}$ and a goal region $\mathcal{G} \subseteq \mathcal{C}_{\text{feasible}}$. Hence, a solution to the planning task is a path π such that $\pi(0) = q_0$ and $\pi(1) \in \mathcal{G}$. In multi-modal planning, finding a valid path also means finding a sequence of modes that are described in Section 4.3.

2.3 Kinematics of Robots Composed of Serial Chains

We are considering walking robots with a structure that can be represented by a directed rooted tree with branches representing the legs that form a serial chain of actuators. A schema of a branch representing a leg of a hexapod robot is depicted in Fig. 6. The edges of the graph correspond to the individual robot links $l \in L$. The vertices correspond to joints/actuators connecting the links.

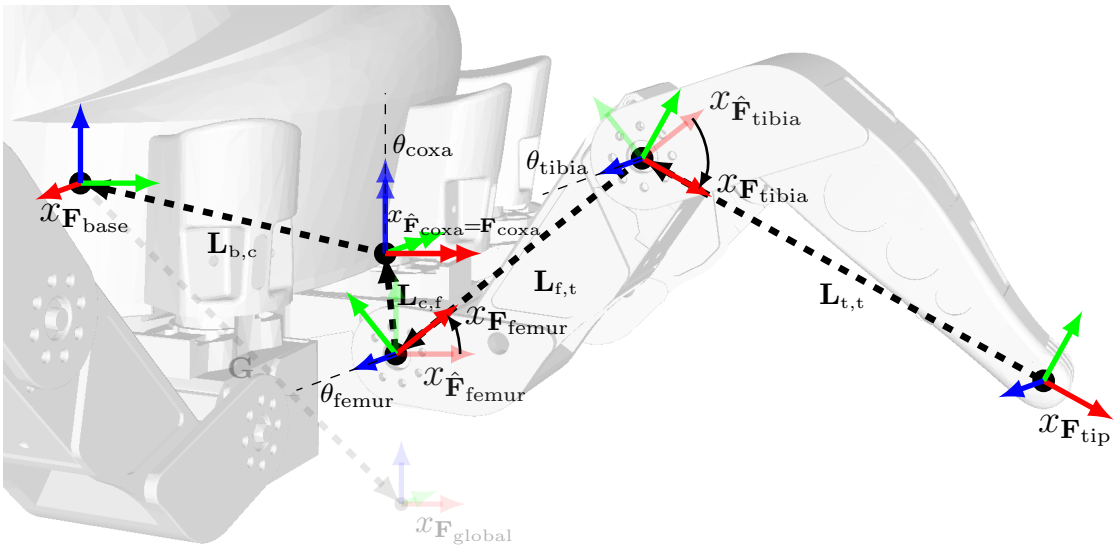


Figure 6: Schematics of a serial kinematic chain of a legged robot leg. The angle of the coxa joint θ_{coxa} is zero in the visualized configuration, thus the coordinate frames $\hat{\mathbf{F}}_{\text{coxa}}$ and $\mathbf{F}_{\text{coxa}}$ are aligned in the figure.

Each vertex/joint is associated with two coordinate frames $\mathbf{F}_l$ and $\hat{\mathbf{F}}_l$, where $\mathbf{F}_l$ is the coordinate frame of the link l, and $\hat{\mathbf{F}}_l$ is the pose of the joint connecting l to its parent link. For the actuated joints, the transformation between $\hat{\mathbf{F}}_{l_2}$ and $\mathbf{F}_{l_1}$ corresponds to a single CDOF of the robot. For the fixed joints, the coordinate frames are equal. Each edge/joint is associated with an isomorphism $\mathbf{L}_{l_1,l_2}$ composed of the rotation $\mathbf{R}_{l_1,l_2}$ and translation $\mathbf{t}_{l_1,l_2}$

$$\mathbf{L}_{l_1,l_2} = \begin{bmatrix} \mathbf{R}_{l_1,l_2} & \mathbf{t}_{l_1,l_2} \\ \mathbf{0}^{1\times 3} & 1 \end{bmatrix}, \tag{4}$$

where $\mathbf{L}_{l_1,l_2}$ describes the isomorphism from $\hat{\mathbf{F}}_{l_2}$ to $\mathbf{F}_{l_1}$, the position of the joint connecting l_2 to l_1 within the coordinate system of l_2; i.e, the edge is oriented from l_2 to l_1

Using the Denavit-Hartenberg convention, $\hat{\mathbf{F}}_{l_2}$ is the coordinate frame of the base of the joint connecting l_1 and l_2. Our robots consist only of revolute joints associated with an angle θ_l, where the subscript l is used for the joint angles because each joint corresponds to a link. By convention, the revolute joint connecting the link l rotates about the z-axis of the reference frame $\hat{\mathbf{F}}_l$. Thus the isomorphism from F_l to $\hat{F}_l$ can be expressed as

$$\mathbf{Z}_l(\theta_l) = \begin{bmatrix} \cos\theta_l & -\sin(\theta_l) & 0 & 0 \\ \sin\theta_l & \cos(\theta_l) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

Let $l_n, l_{n-1}, \ldots, l_2, l_1$ be an oriented path in a tree. The isomorphism $\mathbf{M}_{l_1,l_n}(q)$ is the transformation from the coordinate frame F_{l_n} to F_{l_1} with respect to (w.r.t) the robot configuration q that can be expressed as

$$\mathbf{M}_{l_1,l_n}(q) = \mathbf{L}_{l_1,l_2}\, \mathbf{Z}_{l_2}(\theta_{l_2}) \mathbf{L}_{l_2,l_3}\, \mathbf{Z}_{l_3}(\theta_{l_3}) \mathbf{L}_{l_3,l_4}\, \ldots\, \mathbf{Z}_{l_{n-1}}(\theta_{l_{n-1}}) \mathbf{L}_{l_{n-1},l_n}\, \mathbf{Z}_{l_n}(\theta_{l_n}) \tag{6}$$

where θ_{l_i} is the set angle of the joint connecting l_i at the configuration q.

A point $\mathbf{x}_{l_n}$ is transformed from $\mathbf{F}_{l_n}$ to $\mathbf{F}_{l_1}$ as

$$\hat{\mathbf{x}}_{l_1} = \mathbf{M}_{l_1,l_n}(q) \cdot \hat{\mathbf{x}}_{l_n} \tag{7}$$

where $\hat{\mathbf{x}}$ is the homogeneous coordinate representation of $\mathbf{x}$

$$\hat{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} . \tag{8}$$

Let $\mathbf{p}$ be a point in the coordinate frame $\mathbf{F}_l$. The following steps are performed to find its coordinates $\mathbf{p}(q)$ in the global coordinate frame at a given configuration $q \in \mathcal{C}$.

1. Find an oriented path $l, \ldots, l_{\text{root}}$ from the link to which $\mathbf{p}$ is fixed to the base link.
2. Compute $\mathbf{M}_{l_{\text{root}},l}$ as in (6).
3. Compute $hat\mathbf{p}(q) = \mathbf{G} \cdot \mathbf{M}_{l_{\text{root}},l} \cdot \hat{\mathbf{p}}$, where $\mathbf{G}$ is the isomorphism performing the transformation from $F_{l_{\text{root}}}$ to the global coordinate frame.

Let $\mathbf{p}_l$ be a position in the coordinate frame $\mathbf{F}_l$. Jacobian $\mathbf{J}_{p_l}^{(\theta_l)}$ of $\mathbf{p}_l$ w.r.t. the angle θ_l can be computed as [15].

$$\mathbf{J}_{\mathbf{p}_l}^{(\theta_l)} = \hat{\mathbf{k}}_l \times \mathbf{p}_l \, , \tag{9}$$

where $\hat{\mathbf{k}}_l$ is the unit vector pointing in the direction of the z-axis of the coordinate frame F_l. Note that Jacobian $\mathbf{J}_{\mathbf{p}_l}^{(\theta_l)}$ is in the coordinate frame F_l.

If $\mathbf{J}_{\mathbf{p}_l(q)}^{(\theta_l)}$ is Jacobian of $\mathbf{p}_l(q)$ w.r.t. θ_l in the coordinate frame $\mathbf{F}_l$ at a configuration q, we can compute Jacobian $\mathbf{J}_{\mathbf{p}_l(q)}^{(\theta_l)}(q)$ w.r.t. θ_l in the global reference frame at the configuration q in the following manner.

1. Obtain $\mathbf{M}_{l_{\text{root}},l}(q)$ in the same way as in (6).
2. Set $\mathbf{R}_{l_{\text{root}},l}(q)$ as the rotation component of $\mathbf{M}_{l_1,l}(q)$.
3. Compute $\mathbf{J}_{\mathbf{p}_l(q)}^{(\theta_l)}(q)$ as:
$$\mathbf{J}_{\mathbf{p}_l(q)}^{(\theta_l)}(q) = \mathbf{R}_G \cdot \mathbf{R}_{l_{\text{root}},l}(q) \cdot \mathbf{J}_{\mathbf{p}_l(q)}^{(\theta_l)} \, , \tag{10}$$
where $\mathbf{R}_G$ is the rotation component of $\mathbf{G}$.

Jacobian $\mathbf{J}_{\mathbf{p}}^{(\boldsymbol{\theta})}(q)$ of a point $\mathbf{p}$ fixed to link l w.r.t. the vector $\boldsymbol{\theta}$ of all joint angles can be determined as follows.

1. Find the oriented path $l, \ldots, l_{\text{root}}$ the link $\mathbf{p}$ is fixed to from the base link.
2. Compute $\mathbf{p}_{l_i}(q)$ (position of $\mathbf{p}$ in $\mathbf{F}_{l_i}$) for each link l_i along the path. Set the column corresponding to θ_{l_i} to $\mathbf{J}_{\mathbf{p}_l(q)}^{(\theta_l)}(q)$ computed as in (10).
3. The other columns are zero.

The complete Jacobian matrix $\mathbf{J}_\mathbf{p}(q)$ of a point $\mathbf{p}$ fixed to the link l w.r.t. to the configuration q is

$$\mathbf{J}_\mathbf{p}(q) = \left[\mathbf{J}_\mathbf{p}^{(\mathbf{G})}(q) \quad \mathbf{J}_\mathbf{p}^{(\boldsymbol{\theta})}(q)\right], \tag{11}$$

where $\mathbf{J}_\mathbf{p}^{(\mathbf{G})}(q)$ is Jacobian w.r.t. the body pose $\mathbf{G}$ that can be expressed as

$$\mathbf{J}_\mathbf{p}^{(\mathbf{G})}(q) = \left[\mathbf{E}^{3\times 3} \quad (\mathbf{R}_z\mathbf{R}_y\hat{\mathbf{i}}) \times (\mathbf{p}_0(q) - \mathbf{t}) \quad (\mathbf{R}_z\hat{\mathbf{j}}) \times (\mathbf{p}_0(q) - \mathbf{t}) \quad \hat{\mathbf{k}} \times (\mathbf{p}_0(q) - \mathbf{t})\right], \tag{12}$$

where $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, and $\hat{\mathbf{k}}$ are the unit vectors aligned with the x, y, and z axis, respectively, and $\mathbf{R}_z$, $\mathbf{R}_y$, $\mathbf{R}_x$ are the rotations about the x, y, z axes that define the rotation component of $\mathbf{G}$

$$\mathbf{R}_G = \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x\,, \tag{13}$$

and $\mathbf{p}_0(q)$ is the position of the point $\mathbf{p}$ in the global frame at the configuration q.

2.4 Bézier curve

Bézier curve is a description of a general curve in $\mathbb{R}^d$ defined by the polynomial expression. Bézier curve $\mathbf{B}(t) : [0,1] \to \mathbb{R}^d$ with the degree n is defined by $n+1$ control points $\mathbf{p}_0, \mathbf{p}_2, \ldots, \mathbf{p}_n$ where each $\mathbf{p}_i \in \mathbb{R}^d$. A point on the curve for a given value of t is computed as a weighted sum of the control points. The vector of weights for the given t is defined as $\mathbf{B}_n(t) = [B_{i,n}(t)]_{i\in 0\ldots n}$ where

$$B_{i,n}(t) = \binom{n}{i} t^i\,(1-t)^{n-i}\,. \tag{14}$$

A point on the curve can be expressed as

$$\mathbf{B}(t) = \sum_{i=0}^{n} B_{i,n}(t)\mathbf{P}_i \tag{15}$$

$$= \sum_{i=0}^{n} \binom{n}{i} t^i\,(1-t)^{n-i}\,\mathbf{P}_i\,. \tag{16}$$

Let us express Bézier curve as a multiplication of the matrix $\mathbf{P}$ whose columns are the control points and vector of the weights $\mathbf{B}_n(t)$ are expressed as a function of t

$$\mathbf{B}(t) = \mathbf{P}^{(n)} \cdot \mathbf{B}_n(t) = \begin{bmatrix}\mathbf{p}_0 & \mathbf{p}_2 & \ldots & \mathbf{p}_n\end{bmatrix} \cdot \begin{bmatrix} B_{0,n}(t) \\ B_{1,n}(t) \\ \vdots \\ B_{n,n}(t)\end{bmatrix}. \tag{17}$$

Note that $\mathbf{B}(0) = \mathbf{p}_0$ and $\mathbf{B}(1) = \mathbf{p}_1$ holds.

Chapter 3
Related Work

This chapter provides an overview of the most related motion planning work, together with examples of existing methods for legged locomotion planning. The approaches for rough terrain traversal without planning are also described to provide an overview of the complementary methods. Besides, methods for constraints evaluation are summarized in Section 3.5. Summarizing discussion is presented in Section 3.6.

3.1 Motion Planning

An analytic approach to motion planning is to formulate the planning task using a *potential function* [16, 17] in the configuration space. The virtual force field attracts the robot towards the goal and pushes it away from the obstacles/infeasible regions. If we construct such a function, we can find a valid path by following its gradient, which is the reason why such functions are also called *navigation function* [18]. However, the potential function might have multiple local minima in structured environments, and the method may get stuck. Although methods for dealing with local minima in the potential field exist [17, 19], such as using a harmonic potential function, the approach might still suffer from numerical issues [20].

An approach related to the potential function method is path optimization. The path of the robot is parametrized as a polynomial or piecewise polynomial curve. It is optimized w.r.t a constraint penalty function and optimality criterion, e.g., [21]. The difference from the potential function is that the constraint function does not define the solution but is only used to evaluate possible solutions.

Discrete planning tasks, e.g., planning on a grid or in a graph, are being solved by graph-search algorithms such as the A* algorithm and its variants [22–24]. Applying those to continuous planning problems requires a discretization of the configuration space. Grid-like discretization does not scale for more complex systems [12], and randomized sampling-based methods show to be a robust alternative [25].

3.2 Randomized Sampling-based Planning

Sampling-based planning algorithms solve the problem of discretizing high-dimensional continuous space by sampling the configurations randomly. A graph (also called a planning roadmap) is built from samples to approximate the connectivity of the configuration space. Two general variants of sampling methods can be distinguished [12]:

- **Multi-query (graph-based)** – incrementally constructs a graph by randomly sampling $\mathcal{C}_{\text{feasible}}$ and attempts to connect the sample to nearby vertices. A graph-search algorithm can then search the resulting graph. The approach is referred to as *multi-query* because the constructed graph can be reused for the following queries. A representative is the Probabilistic Roadmap (PRM) [26].

- **Single-query (tree-based)** – incrementally constructs a tree by randomly sampling $\mathcal{C}_{\text{feasible}}$ and connecting the samples to a single neighbor in the tree. The approach is called *single-query* since the resulting tree is not so readily usable for different queries. However, it has to evaluate

fewer connections, which may improve the performance on a single query. A representative is the Rapidly-exploring Random Tree (RRT) [27].

The sampling-based algorithms are designed to be *probabilistically complete* and possible also *probabilistically optimal* that can be described as a property that with the increasing number of sampled configurations grows, the probability of finding a feasible and eventually also the optimal path is going to one [28]. There are many variants of the PRM and RRT based methods that can be found in the literature, e.g., [29–33] to list few. Many of them can be found as implementation using the Open Motion Planning Library [34], which helps to extend and compare various approaches.

Motion planning under constraints is discussed in the survey on sampling-based motion planning by Kingston, *et al.* [12]. For the unconstrained case, the dimensionality of $\mathcal{C}_{\text{feasible}}$ is the same as $\mathcal{C}$. In the constrained case, $\mathcal{C}_{\text{feasible}}$ is restricted by an equality constraint expressed by a constraint function $f : \mathcal{C} \to \mathbb{R}^k$ such that $\mathcal{C}_{\text{feasible}} = \{q \in \mathcal{C} \mid f(q) = \mathbf{0}\}$; hence, the dimensionality of $\mathcal{C}_{\text{feasible}}$ is reduced by k (see Section 2.2). The set $\mathcal{C}_{\text{feasible}}$ has a zero measure in $\mathcal{C}$, and thus, valid configurations cannot be found by random sampling alone. Kingston, *et al.* further differentiate ways of dealing with the sampling and planning under the equality constraints as follows.

- **Reparametrization** – define a new unconstrained configuration space that fully describes the robot's state. The configurations in the new configuration space have to map to valid configurations smoothly. An example is a legged robot with 3DOF legs where a leg's foot tip is fixed to a point on the terrain. In such a case, the joint angles are uniquely defined (under mild assumptions) by the pose of the robot's body.
- **Relaxation** – instead of enforcing $f(q) = \mathbf{0}$, we relax the constraint to $\|f(q)\| < \epsilon$. The set satisfying the relaxed constraint has a non-zero measure in the configuration space. Thus, it delegates the need to solve the constraint to the robot's controller precisely. Note that we always work with some finite precision, and therefore, we can say that a method is relaxation-based when $\epsilon \gg \epsilon_M$, where ϵ_M is the machine precision.
- **Tangent space** – in some cases, it is possible to locally approximate the manifold defined by the constraint function using known valid configurations. Hence, the method is viable only for sampling neighborhoods of known valid configurations and constructing local paths. It is somewhat analogous to the potential function methods.
- **Atlas** – the tangent space can be used to define a piecewise-linear approximate representation of the valid set. Atlas is a data structure storing the tangent spaces. The tangent spaces are generated dynamically. Atlas is used in a variant of the RRT called the AtlasRRT [35].
- **Projection** – define a *projection operator* that takes any configuration and projects it to the valid space. In practice, it means numerically solving the set of equations $f(q) = \mathbf{0}$, e.g., by gradient descend or Newton-Raphson method [36]. Some form of projection to the manifold has to be used in all the previously mentioned approaches except for the reparametrization.

3.3 Multi-modal Planning

In some domains, multiple constraint functions are defining multiple intersecting submanifolds of the configuration space, determining multiple *modes* of the system. In the domain of legged locomotion planning, the modes correspond to specific placements of legs on the terrain – *stances*. The modes can be purely discrete (e.g., a finite set of footholds) or accompanied by continuous parameters defining *mode families* (such as a set of contact regions). A graph representation can be used to describe

the rules for transitioning between modes/mode families [13, 37, 38]. Solving a multimodal planning task thus requires finding a sequence of mode transitions and solving the constrained planning subproblems.

Kingston, *et al.* describe a heuristic approach for searching a sequence of modes selected from parametrized mode families [14]. They test the heuristic on a simplified model of a two-limbed robot moving by grasping bars – similarly to Robonaut 2 [39]. In [11], K. Hauser proposes a general probabilistically complete sampling-based algorithm called Incremental Multi-Modal Probabilistic Roadmap – Incremental-MMPRM for multimodal problems with discrete modes.

3.4 Legged Locomotion

The task of navigating a legged robot in a rough/structured environment can be solved by several approaches, each considering and relaxing different constraints. Legged robots have two main pronounced abilities compared to their wheeled/tracked counterparts.

- **Ability to position the body without changing terrain contacts** – For most legged robots, the body essentially forms a parallel over-actuated platform. Such additional freedom of movement allows the legged robot to avoid obstacles better and redistribute its weight.
- **Ability to select terrain contacts** – Legged robots interact with the environment via a set of footholds. The footholds can be carefully selected and precisely reached, and maintained by the robot's legs. Legged robots can thus navigate environments with very little supportive terrain and control the forces applied to the environment.

While body position and terrain contacts mutually constrain each other, they are not as tightly linked as is the case for wheeled or tracked robots. These extra degrees of freedom, combined with the multimodality of the configuration space, make controlling legged robots a non-trivial task. However, it is possible to decouple the combinatorial and continuous planning successfully. We can distinguish two general kinds of approaches to legged locomotion planning.

1. Planning the continuous path for the robot's body and selecting terrain contacts as needed on a step-by-step basis. Ground-reachability constraint needs to be enforced, which roughly corresponds to the relaxation approaches distinguished by Kingston, *et al.* in [12].
2. Finding a feasible sequence of stances (modes) and solving the continuous constrained subproblems to validate the transitions between modes.

Besides, legged robots can locomote without planning that is briefly overviewed in the following section. The body planning and multi-model planning approaches are described in Section 3.4.2 and Section 3.4.3, respectively.

3.4.1 Locomotion without Planning

Legged robots are capable of negotiating quite a rough terrain even without using any deliberative approach. In [40], Mathis, *et al.* report on experimental demonstration (using human volunteers) that it is possible to walk efficiently even with limited visual information (to about the distance of two steps for humans/bipeds). Methods using a purely reactive control scheme with only positional feedback from the servos, e.g., [41], can be quite successful in negotiating uneven terrain. Approaches based on blind locomotion can even deal with unexpected obstacles or missing supports as reported in [42].

Agility Robotics demonstrated the ability of their biped robots (ATRIAS, Cassie) on rough terrain using blind walking [43], where moderate terrain disturbances are handled by the controller, without foothold planning and exteroception.

In the reactive approaches, it is assumed that most parts of the terrain are safe to step on. Because of that, they have trouble with environments such as stairs, where bad foot placement can lead to failure. An approach dealing with shortcomings of the blind methods has been developed by P. Fankhauser, *et al.* from ANYbotics [44]. If the nominal foothold is unreachable or unsafe, a model of the environment built using exteroceptive sensors is used to find an alternative, valid foothold. Pose optimization is used to find a body pose that ensures the foothold is reachable while maintaining stability. Fankhauser, *et al.* uses a *Signed Distance Field* (SDF) for fast collision detection, which is further described in more detail in Section 4.2.4 because it is adopted in the herein proposed approach. The trajectory of the leg is optimized for speed and collision avoidance. For that end, the method described in [21] is used.

3.4.2 Planning for Body

Using the classification [12] of Kingston, *et al.*, the following approaches can be categorized as *relaxation approaches*. Only a simplified description of the legs' workspaces is considered in these methods, and it is assumed that footholds and single-step leg motions can be found later. The primary consideration is for the movement of the robot's body while making sure all legs have some feasible footholds in their respective workspaces. The footholds are selected such that they facilitate the planned motion of the robot's body.

D. Belter, *et al.* use a two-stage planner in [45]. A high-level grid-based planner finds a path of the robot body using estimated terrain traversability cost. Then, a random sampling-based planner finds a precise path to intermediate goals along the rough path found in the first stage. The sampling planner essentially plans in $SE(2)$. Footholds are selected for the given body positions, and the other body coordinates (height, pitch, roll) are optimized based on the footholds and the terrain.

M. A. Arainn, *et al.* proposed to use map preprocessing to create a grid-map of body positions [46]. The cost of those positions is evaluated based on the kinematic model of the robot. A path of the body is found using a graph-search algorithm, and footholds are selected later.

In [9], P. Čížek, D. Masri, and J. Faigl proposed a similar approach but with random sampling as opposed to the grid discretization. They interpret the 2.5D height map as an image and use image processing techniques to construct a ranking of feasible body positions to construct a probabilistic roadmap. Each body position is evaluated according to the number of valid footholds in the workspace of each leg. The planner then uses a pentapod gait to move between stances with six contacts.

N. Perrin, *et al.* define a notion of *weak collision-freeness* in the robot's configuration space [47]. It is essentially a redefinition of the *collisions free space* in the way that ensures it is possible to assign foot contacts to body poses along a continuous path produced by a sampling-based planner – the workspace of the robot's legs has to intersect with feasible contact regions on the terrain. The method is similar to method [9] proposed by Čížek, *et al.* in that it requires a valid foothold to be present in all legs' workspaces. Perrin, *et al.* do not go into too much detail regarding the application of the method on a hexapod robot mentioned in [47]. However, it appears that stances with six fixed contact points are required, and a simplified model of legs' workspaces is utilized.

Tonneau, *et al.* also proposed an approach based on terrain reachability in [48]. The authors focused on selecting the contacts in such a way that a regular gait pattern is not imposed. The algorithm is hand-crafted and works with the predetermined motion of the body.

In [6], R. Buchanan, *et al.* focus on planning in confined spaces where obstacles limit the motion of

the robot's body. First, they plan the robot's body path, avoid obstacles, and then select footholds. The footholds are selected to support the planned path of the body similarly to [44]. Planning the path for the robot's body is important because obstacles constrain the motion.

A. Norby and A. M. Johnson proposed to plan a trajectory of the robot body using a *long-horizon* sampling-based planning under kinodynamic constraints [49]. Ground reachability is ensured by enforcing a minimum distance to the terrain. Nevertheless, the planner also allows *leap phases* during which no legs are touching the terrain. The specific individual footsteps are planned later.

A nonlinear solver is proposed by A. W. Winkler, *et al.* to find a trajectory by solving constraints considering the robot's dynamic and kinematic model in [50]. They manage to avoid decoupling the foothold planning from the body trajectory planning. Duration of the stance and swing phases for individual legs are part of the trajectory parametrization; so, it is the position of footholds for the stance phases. Note that the number of stance and swing phases has to be selected beforehand. The approach can be considered more of a relaxation approach since the position of the footholds on the terrain is mostly unconstrained, and only an approximation of the legs' workspace is used.

V. Tsounis, *et al.* propose a method based on reinforcement learning [51]. They formulate both the motion planning and control problems as Markov decision problems and learn policy functions implemented as artificial neural networks. Convolution layers are used to process the local terrain represented as a heightmap.

3.4.3 Multi-modal Planning

The approaches that consider the multi-modality of the planning task use a graph representation to capture the possible transition between the individual modes [12, 13, 52] or mode families [14]. Using a graph representation has been successfully used on humanoid/biped robots [10, 53–55]. The modes correspond to the position and orientation of the standing leg, and when selecting possible neighboring modes, several sampled foot positions and orientations are considered.

K. Hauser extensively studies motion planning for multi-legged robots and general multi-modal planning in his dissertation thesis [11] and several related papers [37, 52, 56]. The most crucial parts of his work that are relevant for the legged robot domain are summarized in [36]. He approached the problem by delaying the one-step planning after a *candidate sequence* of modes was found. For considering a step between two stances, he only tests whether the intersection of subsets of configurations attainable from the two stances has a non-empty intersection; the existence of such *transition configuration* is an indication a continuous path could be found [11]. Hauser argues that in very-rough terrains with a limited number of usable footholds, the movement of the robot is mostly constrained just as it places/lifts its foot onto/from the terrain [57]. Thus, a sequence of steps is planned first, including the leg's configurations can be planted/lifted. The smooth motion between those *transition configurations* is computed in the second phase. The first phase is realized by a graph-based algorithm where the vertices are modes corresponding to a particular assignment of the robot's legs to footholds on the terrain. The existence/non-existence of edges corresponds to the existence/non-existence of the *transition configurations*. A random sampling algorithm realizes the smooth motions between those configurations after a candidate path in the graph is found.

Hauser builds on the work of T. Bretl, who coauthored some of Hauser's paper [56, 57]. Bretl worked on a multi-step planner for a multi-limbed free-climbing robot LEMUR [13]. He focused on solving the problem of *disconnection proofs* – showing a path does not exist between two intermediate configurations. The motivation is to avoid using a sampling-based planner for *obviously* impossible steps. Since random-sampling algorithms cannot disprove the existence of a valid path in general, much time can potentially be wasted on solving a single subproblem.

Hauser also formulates a general probabilistically complete multi-modal planner [11] based on a probabilistic roadmap. The algorithm balances sampling the transition configurations and configurations in the individual submanifolds. A very similar approach is used by P. Vernaza, *et al.*. They applied a graph-search method similar to Hauser on a quadruped robot [58], but compute the sequence of *stances* using a randomized graph search algorithm R* [23]. A multi-component cost function is defined to avoid infeasible steps. The robot controller handles the one-step motions. Unlike Hauser's more general approach, Vernaza, *et al.* only focus on planning for a quadruped robot.

Z. Kingston, *et al.* proposed a method of informing the search for the sequence of modes by using results from the previous single-mode planning queries [14]. They work with parametrized families of modes corresponding to *grab locations* on a bar handle for a simplified 2D robot and environment similar to Robonaut 2. Planning a sequence of contact modes in continuous mode families for humanoid robots is studied by A. Escande, *et al.* in [8]. The continuous families represent possible positions of an effector contact on a flat surface of the environment.

To the best of our knowledge, regarding quasi-static gait-free planning for a hexapod robot, the only work done is by T. Bretl and K. Hauser.

3.5 Constraint Evaluation

A major computational bottleneck in motion planning is evaluating the related constraints, such as the kinematic constraints, collision constraints (self-collisions and collisions with the terrain), and stability constraints. Besides, friction constraints and joint torque limit constraints are related to stability constraints. Hence, in constrained motion planning, one has to find valid configurations by solving a set of nonlinear equalities and inequalities given by the relevant constraints unless a viable reparametrization of the configuration space exists. A common method for implementing the projection to the valid subset are descent algorithms – gradient descent, Jacobian inverse, or damped least squares [12, 52, 59]. These methods require the constraints expressed as a differentiable function. Particular approaches for constraint evaluation are further discussed in the following paragraphs of the section.

Only kinematic and support polygon constraints are considered by the authors of [52], and the proposed Iterative Constraint Enforcement (ICE) algorithm is restarted with a different initial sample when the other constraints are violated. In the new iteration, the constraint function is extended to account for other constraints violated by the previous sample.

The requirement to be differentiable, imposed on the constraint function by the descent algorithms, is somewhat strict. For some types of constraints, it may not be possible or efficient. Algorithms like the Nelder-Mead simplex method [60] and Particle Swarm Optimization (PSO) [61] can optimize functions without using their Jacobians. The PSO is used for pose optimization by [45, 62], and it thus seems to be a suitable choice.

One of the hardest constraints to evaluate is the collision constraint. General collision detection can be done by mesh interference using a hierarchical 3D model [63]. A disadvantage of mesh interference detection is its low speed. A method of terrain collision detection that provides a good metric of constraint violation and a gradient of the distance-to-terrain function is a *Signed Distance Field* (SDF) used by [6, 21, 44]. Collisions can be quickly evaluated using the SDF for a robot whose shape is approximated by a set of spheres. The SDF can be computed relatively quickly using a distance transform of sampled functions described in [64].

Some researchers also use learned approximate constraint models that are faster to evaluate during planning. The Gaussian mixture model is used by D. Belter, *et al.* to approximate several constraints of legged robots that do not have a simply closed-form solution [62], such as distance to the edge of the

workspace of the legs and collisions between adjacent legs. N. Das, *et al.* addressed the computational complexity of collision detection by using a technique similar to kernel SVM to learn an approximate collision model and speed up collision queries [65]. The learned model improved the runtime of sampling-based algorithms by orders of magnitude.

3.6 Discussion of Approaches for Legged Locomotion

It is possible to traverse rough environments without planning a whole-body motion using a blind locomotion [21, 41–43] or to plan single-step motions using exteroceptive sensing [44]. These results justify planning using relaxation methods focusing on planning body path first, such as [9, 45–47]. In moderately rough terrain with relatively abundant footholds, these approaches can be sufficient. Planning body path/trajectory while relaxing the kinematic constraints lets us more easily consider other constraints like terrain collisions or kinodynamic constraints [6, 49, 50]. There is an underlying assumption in the methods mentioned above. It is assumed that it is possible to find footholds for each robot's leg in the body poses, which can be valid in many cases.

Multi-modal approaches that primarily plan the sequence of *stances*, such as [13, 14, 36, 58], seem to be suited for scenarios where robot motion is mostly constrained by a limited selection of viable footholds and where the available footholds severely restrict the robot's configuration space. However, the combinatorial complexity of the mode sequencing component of these methods makes them impractical for long-horizon planning. In a real-world deployment, the detailed knowledge of the terrain is restricted to the direct neighborhood of the robot. Hence, planning footholds is mostly restricted to creating short-range plans in practice. The goals for the short-term planner can be set by a long-term planner working with a simplified configuration space and a traversability heuristic.

In scenarios, such as the *obstacle race* briefly introduced in Section 1.1, with very sparse footholds planning, the sequence of stances is crucial for finding a viable solution. The assumption that footholds can be selected after a viable motion for the robot's body is planned is false. Therefore, in this thesis, we focus on the motion planning solution in such scenarios with the assumption of a relatively short-term trajectory.

Chapter 4
Problem Statement

In the context of this thesis, a legged robot is a mammal-like quadruped or an insect-like hexapod robot with a rigid body, bilaterally or radially symmetrically placed identical legs with three to six joints per leg. The present approach is focused on hexapod walking robots with 3DOF on each leg; however, most can be generalized for any legged robot. The following is assumed.

- **Quasi-static motion** – superior static stability is the main advantage of the hexapod platform. A safe traversal of rough and possibly unstable terrain is desired; thus, the robot should move deliberately. At any moment, the robot should be able to stop its motion and backtrack. Any dynamic motions such as jumps or steps that require active balancing are therefore undesirable.

  Formally it means ignoring the first and second-order terms in the Lagrangian equation, which leaves (26).

- **Rigid environment** – in scenarios such as exploring collapsed buildings, disturbing the environment is dangerous; therefore, any deformable terrain should be avoided. A model of a non-rigid environment can be included later. It is expected to restrict the set of viable footholds and requires more careful consideration of the forces applied to the environment. Hence, the environment is assumed to be rigid.

- **Finite set of discrete footholds** – In highly structured terrain, locally optimal footholds such as small concavities might be desirable. Also, due to the physical dimensions of the foot tip, the contacts with the environment cannot be selected completely arbitrarily. Any sliding of the robot's effectors on the terrain is also undesirable as it might disturb the environment. Therefore, we work with a finite set of footholds selected before planning.

In the rest of the chapter, the configuration space of a statically stable-legged robot is described. The planning task in the described configuration space is defined. Finally, based on the early results of preliminary solutions, observations about the nature of the solution of the planning task are presented to support the selected design choice of the proposed free-gait planning.

4.1 Configuration Space

A statically stable configuration q of a legged robot with N legs with M degrees of freedom consists of a body pose and joint configurations. The body pose is an isomorphism $\mathcal{P} \in SE(3)$ and the angle of the j-th joint of the i-th leg is a planar rotation $\theta_{i,j} \in SO(2)$. The whole configuration space is then $\mathcal{C} = SE(3) \times SO(2)^{NM}$ with the dimensionality $6 + NM$. That is NM CDOF and $6 + NM$ total DOF. Therefore, a hexapod robot with 3DOF legs has 24 total DOF, and 18 of them are controllable.

Only some allowed parts of the robot, the robot's *effectors*, can contact the terrain. Other contacts with the terrain are considered a collision. Slippage of effectors is undesirable, and once placed on the terrain, their position is fixed in the global coordinate frame. The effectors in contact with the terrain are the only way the robot can exert forces on the environment. The effectors used to exert the forces are in a *stance state* while the effectors not in the stance state are in a *swing state*. The state of the effector is considered to change instantaneously.

4.1 Configuration Space

A *stance* σ is a partial mapping $\sigma : \mathcal{E} \to \mathcal{H}$, where $\mathcal{E}$ is a set of the robot *effectors* and $\mathcal{H} \subseteq \mathbb{R}^3$ is a set of viable footholds on the terrain. Let us denote Σ the set of all stances. The domain of σ, $\text{dom}(\sigma)$ are the effectors for which the mapping is defined. We say that effectors in $\text{dom}(\sigma)$ are *fixed* by σ. The image of σ, $\text{im}(\sigma) \subseteq \mathcal{H}$ is the subset of footholds to which some effector is mapped by σ.

For each valid configuration $q \in \mathcal{C}$, there exists a stance that fixes the contacts allowing the robot to exert the forces necessary to maintain its stability. There might be multiple stances associated with a given configuration q. Indeed, multiple effectors can be touching the terrain in a given configuration, but only a smaller subset of effectors might be considered in a stance state and used to exert forces.

A stance defines a *contact constraint*. A configuration satisfies the contact constraints of σ *iff* all effectors fixed by σ are in the position they are assigned by σ. The set of all configurations satisfying the contact constraint of σ is called the *contact submanifold* of σ, denoted $\mathcal{C}_\sigma$.

The stance σ is critical for defining constraints and the subset of valid configurations $\mathcal{C}_{\text{feasible}}$. It is therefore helpful to include σ in the description of the configuration. Therefore, we can define $\mathcal{C}_\Sigma$ as

$$\mathcal{C}_\Sigma = SE(3) \times SO(2)^{NM} \times \Sigma \,. \tag{18}$$

Here, Σ is a part of the configuration space definition and the stance σ associated with a given configuration is used to determine if the stance-configuration $q_\sigma = (q, \sigma)$ is valid, i.e., $q_\sigma \in \mathcal{C}_{\Sigma,\text{feasible}}$.

Beside the contact constraint, the following restrict the configuration space.

- **Force/torque, friction** – The robot must be able to hold the configuration q such that the sum of forces acting on the terrain equals the force of gravity. The forces must not exceed the limits imposed by the terrain. The torques at the robot's joints cannot exceed the limit of the actuators. Note that the joint torques could be made a part of the configuration space. However, it may be easier to restrict configurations where it is impossible to achieve the necessary forces without exceeding the torque limits, then increasing the dimensionality of the configuration space.

  The stance is important in defining these constraints because it specifies effectors that may exert forces on the environment.

- **Terrain collisions** – All parts of the robot must be at a safe distance from the environment. The stance defines the allowed contact with the environment. It is necessary to allow the robot effectors to contact the terrain near the footholds or even intersect the terrain to account for effector compliability.

- **Joint position limits and self collision** – The rotation angles $\theta_{i,j}$ must not exceed the limits given by the construction of the robot and the properties of the actuators. Any two links of the robot must not intersect each other. These limitations constraint the allowable configurations of the robot's joints.

Let us define $\mathcal{F}_\sigma \subseteq \mathcal{C}_\sigma$, where $\mathcal{F}_\sigma$ are all configurations q such that q_σ satisfies all other constraints in addition to the contact constraint of σ. The set $\mathcal{F}_\sigma$ is called *stance region*[3] [11]. While the contact constraint is an equality constraint, the other constraints are inequality constraints and do not reduce the topological dimension of $\mathcal{C}_\sigma$.

The subset of valid configurations $\mathcal{C}_{\text{feasible}}$ is the union of $\mathcal{F}_\sigma \subseteq \mathcal{C}_\sigma$,

$$\mathcal{C}_{\text{feasible}} = \bigcup_{\sigma \in \Sigma} \mathcal{F}_\sigma \,, \tag{19}$$

[3]Hauser calls it the feasible space but we find that less intuitive.

and therefore,

$$\mathcal{C}_{\Sigma,\text{feasible}} = \bigcup_{\sigma\in\Sigma} (\mathcal{F}_\sigma \times \{\sigma\}) \,. \tag{20}$$

4.2 Constraints

In this section, the individual constraints of the robot's configuration space are discussed. The description provided here is general, and the exact definition of the constraints is implementation-dependent. For the exact formulation of the considered constraints, the reader is referred to Section 5.2.

4.2.1 Contact constraint

The contact constraint is defined w.r.t. a given configuration q and a stance σ. It is a set of equality constraints

$$\forall e \; \text{FK}_q(e) - \sigma(e) = \mathbf{0} \; ; \; e \in \text{dom}(\sigma) \,, \tag{21}$$

where $\text{FK}_q(e)$ is the forward kinematic solution for the effector e in the configuration q. For $\sigma \in \Sigma$, let $\mathcal{C}_\sigma \subseteq \mathcal{C}$ be the set of all configurations $q \in \mathcal{C}$ that satisfy the contact constraint of σ. The set $\mathcal{C}_\sigma$ forms a submanifold of $\mathcal{C}$. Since the position of the foothold has 3DOF, each foothold fixed by σ reduces the topological dimension of $\mathcal{C}_\sigma$ by three compared to $\mathcal{C}$. Let $|\sigma|$ be the number of contact points fixed by σ, then it holds that

$$\dim(\mathcal{C}_\sigma) = \dim(\mathcal{C}) - 3|\sigma| \,. \tag{22}$$

4.2.2 Stability, Friction and Torque Limits

A necessary condition for the robot's stability at some configuration q and stance σ is that the position of the center of mass in q projected to the level plane $\hat{\mathbf{p}}_{\text{COM}}(q)$ lays within the support polygon; that is a convex hull of the footholds fixed by σ projected to the level plane. This constraint is easy to evaluate; however, it is not a general condition for stability. The complete formulation needs to consider the forces applied to the footholds.

Let $\sigma(e_1), \sigma(e_2), \ldots, \sigma(e_n)$ be the footholds for the effectors $e_1, e_2, \ldots, e_n$ fixed by σ. Let ν_i, μ_i, f_i for $i = 1, 2, \ldots, n$ be the respective terrain normals, static coefficient of friction, and reactive forces acting on the robot. The robot is in the static equilibrium if the following constraint are satisfied.

- The net force action on the robot is zero

$$m\mathbf{g} + \sum_{i=1}^{n} f_i = \mathbf{0} \,, \tag{23}$$

  where $\mathbf{g}$ is the gravity vector.

- The net torque acting on the robot is zero

$$\mathbf{p}_{\text{com}}(q) \times m\mathbf{g} + \sum_{i=1}^{n} \sigma(e_i) \times f_i = \mathbf{0} \,, \tag{24}$$

  where $\mathbf{p}_{\text{com}}(q)$ is the position of the robot's center of mass in the configuration q.

- All forces lay in the friction cones of their respective footholds

$$\forall i : \left\| (I - \nu_i \nu_i^T) f_i \right\|_2 \leq \mu_i \nu_i^T f_i \,. \tag{25}$$

- The reaction forces can be achieved without exceeding the maximum torques acting on the joints. Let $\boldsymbol{\tau}$ be the vector of torques

$$\boldsymbol{\tau} = \mathbf{G}(q) - \sum_{i=1}^{n} \left(\mathbf{J}^{(\boldsymbol{\theta})}_{\mathbf{p}_i(q)}(q) \right)^T \cdot f_i \,, \tag{26}$$

  where $\mathbf{G}(q)$ is the generalized gravity

$$\mathbf{G}(q) = \left(\mathbf{J}^{(\boldsymbol{\theta})}_{\mathbf{p}_{\text{COM}}(q)}(q) \right)^T \cdot m\mathbf{g} \,. \tag{27}$$

The torques values can be found as a solution to the Second Order Cone Program (SOCP) [66]. The constraint of the SOCP are expressed in (23) to (27).

4.2.3 Joint Position Limits and Self-collisions

Let us discuss constraints defined only by the joint configuration $\boldsymbol{\theta}$ component of the robot's configuration. Some joint configurations are infeasible because links of the robot may collide with one another and because of the limited range of the joints. Each joint has to respect its maximum and minimum angle limit. This constraint is easy to check and every θ_i of $\boldsymbol{\theta} = [\theta_1 \theta_2 \dots \theta_n]^T$ has to satisfy

$$\theta_{i,\text{MIN}} \leq \theta_i \leq \theta_{i,\text{MAX}} \,. \tag{28}$$

The limits $\theta_{i,\text{MIN}}$ and $\theta_{i,\text{MAX}}$ are given by the range of the robot's actuators and by the construction of the robot to prevent collisions between adjacent links.

A valid joint configuration must not lead to self-collision. A straightforward way to check self-collisions is to use a mesh-interference checker, such as [63]. A problem with the mesh-based collision checker is that it gives only a binary answer. It does not provide a metric to evaluate how much the constraint is violated or how close a given configuration is to violate the constraint. It is also relatively slow to evaluate. A representation using a simplified geometric model of the robot could be faster and sufficiently precise. Using geometric primitives allows for a closed-form expression of the distance between two links. It is possibly faster to compute than mesh intersection and possibly provides more information about the distance to and obstacle.

In some cases, it is possible to use knowledge about the robot to simplify the representation of the collision constraint. For instance, the collisions between adjacent links can be handled by limiting the range of the motion of the joint connecting them. In fact, we can eliminate the risk of self-collision all together by imposing strict bounds on the joint angle. A less restrictive approach would be to determine which body parts may not collide and only check collisions between the pairs that can. For example, it may be enough for a legged robot to check collisions between neighboring legs only.

4.2.4 Terrain Collisions

Except for the effectors, no part of the robot may intersect the environment. This constraint must be considered when walking on rough terrain and in confined spaces. A mesh-interference checker can be used when checking for self-collisions. In addition to the issues with the mesh-based checker discussed in the previous section, we would need to represent the environment using a mesh or a set of meshes. Here, using geometric primitives such as spheres is not as straightforward as in the case of self-collisions. It is unfeasible to represent a rough environment using geometric primitives. One approach is to approximate only the shape of the robot using geometric primitives, namely spheres.

With a representation of the terrain that would allow computing distance to the obstacle at every point easily, it would be trivial to test for collision with the sphere model [21]. Such a representation can be a look-up table precomputed using a distance transformation, which is the Signed Distance Field (SDF) [64].

4.3 Planning Task

A general planning task $\mathcal{P}$ is defined by the configuration space $\mathcal{C}$, the constraints defining $\mathcal{C}_{\text{feasible}} \subseteq \mathcal{C}$, an initial configuration $q_{\text{init}} \in \mathcal{C}_{\text{feasible}}$, and a goal condition defining a goal set $\mathcal{G} \subseteq \mathcal{C}_{\text{feasible}}$. For example, a goal can be defined by some goal stance σ_{goal} as $\mathcal{F}_{\sigma_{\text{goal}}}$ or any other subset of $\mathcal{C}$ such as all configurations with the given x, y coordinates. A solution to $\mathcal{P}$ is a continuous path $\pi : [0,1] \to \mathcal{C}_{\text{feasible}}$, such that $\pi(0) = q_{\text{init}}$ and $\pi(1) \in \mathcal{G}$.

Because of (20) and since for every t it holds that $\pi(t) \in \mathcal{C}_{\text{feasible}}$, there exists at least one stance σ at each t such that $(\pi(t), \sigma) \in \mathcal{C}_{\Sigma,\text{feasible}}$. Note that there can be multiple such stances that form a valid pair with q. Let us define $\mu : \mathcal{C}_{\text{feasible}} \to \Sigma$ such that $\mu(q) = \operatorname{argmax}_{\{\sigma \mid (q,\sigma) \in \mathcal{C}_{\Sigma,\text{feasible}}\}} |\sigma|$. Each configuration on the path π shall be associated with the largest possible stance. It is assumed that all legs, whose foot tips are in contact with the terrain, are in the stance phase. Thus, $\pi_\Sigma : [0,1] \to \mathcal{C}_{\Sigma,\text{feasible}}$ is defined as

$$\pi_\Sigma(t) = (\pi(t), \mu(\pi(t))) \,. \tag{29}$$

Due to the nature of $\mathcal{C}$ and Σ, the *stance*s switch discretely at $t_1, t_2, \ldots, t_k$. The interval $[0,1]$ is split into subintervals $\mathcal{I}_1 = (0, t_1), \mathcal{I}_2 = (t_1, t_2), \ldots, \mathcal{I}_k + 1 = (t_k, 1)$; those correspond to stances $\sigma_1, \sigma_2, \ldots, \sigma_{k+1}$. At the times of transitions between stances t_i, the constraints of both stances must be satisfied. It must hold that $\pi(\mathcal{I}_i) \subseteq \mathcal{F}_{\sigma i}$ and $\pi(t_i) \in \mathcal{F}_{\sigma i} \cap \mathcal{F}_{\sigma i+1}$. Let, without the loss of generality, $|\sigma_1| > |\sigma_2|$, then the contact constraint of σ_1 and the stability constraint of σ_2 must be both satisfied by all configurations in $\mathcal{F}_{\sigma 1} \cap \mathcal{F}_{\sigma 2}$.

Notice that for each solution to a general planning task, there are two implicit sequences:

- a sequence of stances that the robot uses to support its body during the motion; and
- a sequence of *intermediate configurations* $\pi(t_1), \pi(t_2), \ldots, \pi(t_k)$ at which the discrete switch from one stance to another happens.

Let has a planning task $\mathcal{P} = (\mathcal{C}_{\text{feasible}}, q_{\text{start}}, \mathcal{G})$ and sequence

$$\mathcal{S} = q_0 \; \sigma_1 \; q_1 \; \sigma_2 \; q_2 \; \ldots \; \sigma_{k+1} \; q_{k+1} \tag{30}$$

satisfying the requirements:

$$\sigma_1 = \mu(q_{\text{start}}) \,, \tag{31}$$
$$q_0 = q_{\text{start}} \,, \tag{32}$$
$$q_{k+1} \in \mathcal{G} \,, \tag{33}$$
$$q_{k+1} \in \mathcal{F}_{\sigma k+1} \,, \tag{34}$$
$$q_i \in \mathcal{F}_{\sigma i} \cap \mathcal{F}_{\sigma i+1} \,. \tag{35}$$

The existence of such a sequence is a necessary condition for the existence of a solution to the planning task $\mathcal{P}$. The sequence $\mathcal{S}$ is called a *candidate sequence* of $\mathcal{P}$.

If a candidate sequence for the planning task $\mathcal{P}$ is obtained, the problem decomposes to $k+1$ single-mode constrained planning problems. Let $\mathcal{S} = q_0\ \sigma_1\ q_1\ \sigma_2\ q_2\ \ldots\ \sigma_{k+1}\ q_{k+1}$ be a candidate sequence for $\mathcal{P}$. The solution to $\mathcal{P}$ – π can be obtained by finding $\pi_1, \pi_2, \ldots \pi_{k+1}$ such that

$$\pi_i : (0,1) \to \mathcal{F}_{\sigma i}\ , \tag{36}$$

$$\pi_i(0) = q_{i-1}\ , \tag{37}$$

$$\pi_i(1) = q_i\ . \tag{38}$$

Chapter 5
Proposed Method

In this chapter, a description of the proposed method is provided. First, an overview of the method is outlined, and the used constraint functions are formally introduced. Then, the proposed solution to finding a candidate sequence is presented in Section 5.3. The proposed approach for finding a smooth path is presented in Section 5.4. Finally, the chapter is concluded with the discussion of the transition sampling approach in Section 5.5.

5.1 Overview of Proposed Planning Method

We are given the initial configuration $q \in \mathcal{C}_{\text{start}}$ and an initial stance σ_{start}. The goal is given as an $x - y$ position of the body because specifying the whole body configuration or a specific stance as a goal would be too restrictive. The propose approach follows the methodology used by Hauser [11] and Vernaza, *et al.* [58]. We decompose the planning task into two steps that are visualized in Fig. 7.

1. **Find a candidate sequence of stances (Candidate sequence planning).** The first step is based on the A* graph search algorithm. The vertices of the graph correspond to feasible stances. Because of the assumption of quasi-static motion, a valid stance must fix at least three effectors. An edge exists between two vertices σ_1, σ_2 *iff* $\mathcal{F}_{\sigma 1} \cap \mathcal{F}_{\sigma 2} \neq \emptyset$.

   Since the searched graph is building during the search, we use a simple geometric criterion to select potential neighboring stances when enumerating all neighbors of a vertex, The existence of edges to the potential neighbors has to be verified by finding a configuration satisfying the constraints of both stances. A random configuration close to $\mathcal{F}_{\sigma 1} \cap \mathcal{F}_{\sigma 2}$ is sampled randomly and projected onto the set by gradient descend methods like Newton-Raphson or Levenberg-Marquard algorithm.

2. **Find the final motion (Single-step planning).** In the second step, we need to ensure that the robot can move between the intermediate configurations in the candidate sequence. Here, we assume it is relatively easy for simple-legged robots and that the smooth paths between the intermediate configurations are not complicated. Therefore, we propose to represent the subpaths as Bézier curves in $\mathcal{C}$. A numeric method similar to the one used for finding the intermediate configurations in the first step is used to move the curve close to $\mathcal{F}_\sigma$. A relaxation approach is employed as only limited precision is possible with the polynomial Bézier curve.

The used decomposition does not restrict the set of solvable planning tasks because the existence of a candidate sequence is a necessary condition for the existence of a complete solution. Also, as stated in [11], the motion of the robot is the most constrained near the intersections of the stance regions $\mathcal{F}_\sigma$; therefore, the existence of intermediate configuration is a good indication that a complete solution exists. However, the existence of a candidate sequence does not guarantee that the second step will succeed. If it is impossible to obtain a valid path from a particular candidate sequence, the task may still be solvable. It may be enough to resample the intermediate configurations. It is also possible to restart the planning with an adjusted cost of edges.

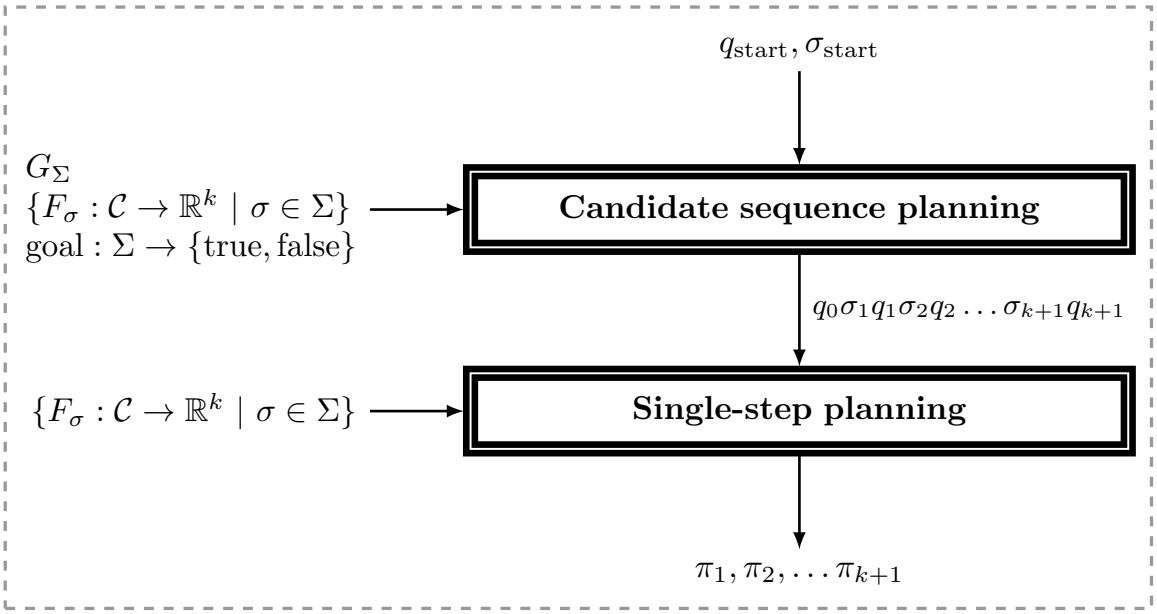


Figure 7: Schema of the planning method

5.2 Constraint Functions

This section presents a formal definition of the constraint function restricting the sets $\mathcal{F}_\sigma$. The constraint function provides a measure of how much a given constraint is violated. The functions are also differentiable to allow for the use of iterative descend methods.

We can numerically solve the set of nonlinear equalities and inequalities

$$f(q) = \mathbf{0}$$
$$g(q) \leq \mathbf{0}$$

where $f : \mathcal{C} \to \mathbb{R}^n$ is the equality constraint penalty function, and $g : \mathcal{C} \to \mathbb{R}^m$ is the inequality constraint penalty function. The inequality constraints are handled using the *active set* method. The combined constraint function are solved as

$$F(q) = \begin{bmatrix} f(q) \\ \hat{g}(q) \end{bmatrix} \tag{39}$$

where $\hat{g}(q)$ are the non-zero values of $g(q)$.

In particular, the following constraint penalty functions are used.

- **Contact constraint** $f_{\mathcal{C}_\sigma}$ – For each leg whose foot tip is fixed by σ, the difference of the position of the foot tip is given by the forward kinematics, body pose, and the target position given by the stance.
- **Support polygon constraint** $g_{\mathcal{S}_\sigma}$ – We use the support polygon and the Center of Mass (COM) position to evaluate stability. The stance also gives the support polygon. As of now, we do not consider the force/torque limits.
- **Joint limits** $g_{\boldsymbol{\theta}}$ – The constraint restricts the range of the robot's joints. We use this constraint to prevent self-collisions. The limits are set such that no parts of the robot can intersect each other in any configuration.

- **Terrain collisions** $g_{\text{SDF}\,\sigma}$ – The constraint penalizes collisions with the terrain using an approximate collision model of the robot and SDF.
- **Goal constraint** $g_{\mathcal{G}}$ – The constraint is added when testing if a configuration is a goal configuration; see Section 5.3.3.

Our constraint function is thus defined as

$$F_{\mathcal{F}_\sigma}(q) = \begin{bmatrix} f_{\mathcal{C}_\sigma}(q) \\ \hat{g}_{\mathcal{S}_\sigma}(q) \\ \hat{g}_{\boldsymbol{\theta}}(q) \\ \hat{g}_{\text{SDF}_\sigma}(q) \end{bmatrix}. \tag{40}$$

The individual constraints are further detailed in the following paragraphs.

5.2.1 Contact Constraint

Let $\text{FK}_q(e_i)$ be the forward kinematic solution for the i-th effector e, fixed by a stance σ at the configuration q. Let $\sigma(e_i)$ be the target position of the i-th effector assigned by the stance σ. The contact constraint penalty function is then

$$f_{\mathcal{C}_\sigma}(q) = \begin{bmatrix} \text{FK}_q(e_1) - \sigma(e_1) \\ \text{FK}_q(e_2) - \sigma(e_2) \\ \vdots \\ \text{FK}_q(e_{|\sigma|}) - \sigma(e_{|\sigma|}) \end{bmatrix}. \tag{41}$$

The Jacobian matrix of the contact constraint is a concatenation of kinematic Jacobians of the individual foot tips w.r.t. the whole robot configuration

$$\mathbf{J}_{f_{\mathcal{C}_\sigma}}(q) = \begin{bmatrix} \mathbf{J}_{e_1}(q) \\ \mathbf{J}_{e_2}(q) \\ \vdots \\ \mathbf{J}_{e_{|\sigma|}}(q) \end{bmatrix}. \tag{42}$$

5.2.2 Support Polygon Constraint

Let σ be the support stance and the support n-gon be the convex hull of the foot tip positions prescribed by σ projected to the x-y plane, the vertices of the support polygon be $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_n, \mathbf{v}_{n+1}$, where $\mathbf{v}_{n+1} = \mathbf{v}_1$, and $\hat{\nu}_i$ be the outward pointing normal of the i-th edge $(\mathbf{v}_{i+1} - \mathbf{v}_i)$ of the support polygon. The position $\mathbf{p}_{\text{COM}}$ of the center of mass of the robot is computed as weighted sum of the COM positions of individual links

$$\mathbf{p}_{\text{COM}}(q) = \frac{1}{M} \sum_{j=1}^{|\mathcal{L}|} m_j \mathbf{p}_{\text{COM},j}(q)\,, \tag{43}$$

where $\mathcal{L}$ is the set of robot's links. The point $\mathbf{p}_{\text{COM},j}(q)$ is the position of the j-th link's COM, m_j is the mass of j-th link, and M is the total mass of the robot.

Let $\hat{\mathbf{p}}_{\text{COM}}$ be the position of the center of mass projected to the x-y plane. The support polygon constraint penalty function is determined using

$$g_{\mathcal{S}_\sigma}(q) = \begin{bmatrix} \hat{\nu}_1 \cdot (\hat{\mathbf{p}}_{\text{COM}}(q) - \mathbf{v}_1) + M_{\text{stab}} \\ \hat{\nu}_2 \cdot (\hat{\mathbf{p}}_{\text{COM}}(q) - \mathbf{v}_2) + M_{\text{stab}} \\ \vdots \\ \hat{\nu}_n \cdot (\hat{\mathbf{p}}_{\text{COM}}(q) - \mathbf{v}_n) + M_{\text{stab}} \end{bmatrix}, \tag{44}$$

where $M_{\text{stab}} \geq 0$ is a stability margin. The Jacobian matrix of the support polygon penalty can be computed as

$$\mathbf{J}_{g_{\mathcal{S}_\sigma}}(q) = \begin{bmatrix} \hat{\nu}_1^T \cdot \mathbf{J}_{\hat{\mathbf{p}}_{\text{COM}}}(q) \\ \hat{\nu}_2^T \cdot \mathbf{J}_{\hat{\mathbf{p}}_{\text{COM}}}(q) \\ \vdots \\ \hat{\nu}_n^T \cdot \mathbf{J}_{\hat{\mathbf{p}}_{\text{COM}}}(q) \end{bmatrix}, \tag{45}$$

where $\mathbf{J}_{\mathbf{p}_{\text{COM}}}(q)$ is defined as

$$\mathbf{J}_{\mathbf{p}_{\text{COM}}}(q) = \frac{1}{M} \sum_{j=1}^{|\mathcal{L}|} m_j \mathbf{J}_{\mathbf{p}_{\text{COM},j}}(q). \tag{46}$$

5.2.3 Joint Limits

Let $\theta_{\text{max},i,j}$ and $\theta_{\text{min},i,j}$ be the maximum and minimum angle of the j-th joint of the i-th leg, respectively. The joint limit penalty is computed as

$$g_{\boldsymbol{\theta}}(q) = \begin{bmatrix} \max(\theta_{\text{min},1,1} - \theta_{1,1}, \theta_{1,1} - \theta_{\text{max},1,1}) \\ \max(\theta_{\text{min},1,2} - \theta_{1,2}, \theta_{1,2} - \theta_{\text{max},1,2}) \\ \vdots \\ \max(\theta_{\text{min},2,1} - \theta_{2,1}, \theta_{2,1} - \theta_{\text{max},2,1}) \\ \max(\theta_{\text{min},2,2} - \theta_{2,2}, \theta_{2,2} - \theta_{\text{max},2,2}) \\ \vdots \\ \max(\theta_{\text{min},n,m} - \theta_{n,m}, \theta_{n,m} - \theta_{\text{max},n,m}) \end{bmatrix}. \tag{47}$$

Jacobian of the joint limit function can be defined as

$$\mathbf{J}_{g_{\boldsymbol{\theta}}}(q) = \begin{bmatrix} \mathbf{0}^{1\times 6} & \partial_{1,1} & 0 & \dots & 0 & 0 & \dots & 0 \\ \mathbf{0}^{1\times 6} & 0 & \partial_{1,2} & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}^{1\times 6} & 0 & 0 & \dots & \partial_{2,1} & 0 & \dots & 0 \\ \mathbf{0}^{1\times 6} & 0 & 0 & \dots & 0 & \partial_{2,2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}^{1\times 6} & 0 & 0 & \dots & 0 & 0 & \dots & \partial_{n,m} \end{bmatrix}, \tag{48}$$

where $\partial_{i,j} = \text{sgn}(2\theta_{i,j} - \theta_{\text{min},i,j} - \theta_{\text{max},i,j})$.

5.2.4 Terrain Collisions

The shapes of the robot's links are approximated by several spheres, where the set of spheres should completely cover all links (see Fig. 8). Let k spheres cover the whole robot, r_i being the radius of

the i-th sphere, and $c_i(q)$ be the position of the center of the i-th sphere at the configuration q. Let $\text{SDF} : \mathbf{F}_{\text{global}} \to \mathbb{R}$ be a function computing the signed distance to terrain for query points in the global coordinate frame. The value of $\text{SDF}(\mathbf{x})$ can be negative if $\mathbf{x}$ is inside the terrain. The function is realized by the Signed Distance Field (SDF), which is precomputed before planning.

Figure 8: Sphere-based robot collision model approximation.

The collision penalty is computed as

$$g_{\text{SDF}\,\sigma}(q) \approx \begin{bmatrix} M_{\text{col}} + r_1 - \text{SDF}(c_1(q)) \\ M_{\text{col}} + r_2 - \text{SDF}(c_2(q)) \\ \vdots \\ M_{\text{col}} + r_k - \text{SDF}(c_k(q)) \end{bmatrix}, \tag{49}$$

where $M_{\text{col}} \geq 0$ is the *collision margin*.

The definition of the collision penalty, in combination with the contact constraint, leads to $\mathcal{F}_\sigma = \emptyset$ for $M_{\text{col}} \gg 0$. Since a large collision margin prevents the foot tips from reaching their target footholds, we propose to locally relax the collision constraint near the footholds similarly to [44]. The *step function* is defined

$$s(x) = \begin{cases} 0 & x \leq r \\ 1 - \frac{1}{b^2}\,(x - r - b)^2 & x \leq r + b \\ 1 & \text{otherwise} \end{cases}. \tag{50}$$

where $r \geq 0$ is a *dead-zone radius* and $b \geq 0$ is *brim size*.

Using the SDF, we can define a *closest point function* that computes the nearest point on the terrain for a given position:

$$\text{CP}(\mathbf{x}) = \mathbf{x} - \text{SDF}(\mathbf{x}) \nabla \text{SDF}(\mathbf{x}) \,. \tag{51}$$

For a stance σ, we can define a *relaxation function*

$$\rho_\sigma(\mathbf{x}) = \prod_{i=1}^{|\sigma|} s(\|\sigma(e_i) - \text{CP}(\mathbf{x})\|_2) \,. \tag{52}$$

Then, the *collision potential function* for σ becomes:

$$p_\sigma(\mathbf{x}, r) = \rho_\sigma(\mathbf{x})\ (-\text{SDF}(\mathbf{x}) + \rho_\sigma(\mathbf{x})\ M_{\text{col}} + r)\ . \tag{53}$$

The collision constraint function is properly defined in terms of the potential function

$$g_{\text{SDF}\,\sigma}(q) = \begin{bmatrix} p_\sigma(c_1(q), r_1) \\ p_\sigma(c_2(q), r_2) \\ \vdots \\ p_\sigma(c_k(q), r_k) \end{bmatrix} \tag{54}$$

and its Jacobian is defined as

$$\mathbf{J}_{g_{\text{SDF}\,\sigma}}(q) = \begin{bmatrix} \nabla p_\sigma(c_1(q), r_1) \\ \nabla p_\sigma(c_2(q), r_2) \\ \vdots \\ \nabla p_\sigma(c_k(q), r_k) \end{bmatrix}. \tag{55}$$

The gradients can be computed as

$$\nabla p_\sigma(\mathbf{x}, r) = -\rho_\sigma(\mathbf{x})\, \nabla \,\text{SDF}(\mathbf{x}) - SDF(\mathbf{x})\, \nabla \rho_\sigma(\mathbf{x}) + 2\, M_{\text{col}}\, \rho_\sigma(\mathbf{x})\, \nabla \rho_\sigma(\mathbf{x}) + r\, \nabla \rho_\sigma(\mathbf{x})\,, \tag{56}$$

where

$$\nabla \rho_\sigma(\mathbf{x}) = \sum_{i=1}^{|\sigma|} \nabla \left(s(\|\sigma(e_i) - \text{CP}(\mathbf{x})\|_2) \right) \prod_{j=0\,;\,j\neq i}^{|\sigma|} s(\|\sigma(e_j) - \text{CP}(\mathbf{x}\|_2) \tag{57}$$

$$\nabla \left(s(\|\sigma(e_i) - \text{CP}(\mathbf{x})\|_2) \right) = -\, s'(\|\sigma(e_i) - \text{CP}(\mathbf{x})\|_2)\, \nabla \,\text{CP}(\mathbf{x}) \frac{\sigma(e_i) - \text{CP}(\mathbf{x})}{\|\sigma(e_i) - \text{CP}(\mathbf{x})\|_2} \tag{58}$$

$$\nabla \,\text{CP}(\mathbf{x}) = (\mathbf{E} - \nabla \text{SDF}(\mathbf{x}) \nabla \text{SDF}(\mathbf{x})^T). \tag{59}$$

A visualization of the collision potential function near a foothold on flat terrain can be seen in Figure 9. The relaxation function creates a funnel in the collision that allows the effector of the robot to contact the terrain. To account for compliant effectors, the collision potential field allows the effectors to penetrate the terrain in the *deadzone*. Notice that for points far enough from the footholds of σ, the collision potential function is the same as the collision function defined in (49).

5.3 Finding a Candidate Sequence

The first step of the proposed planner is the search for a candidate sequence of a given planning task $\mathcal{P} = (\mathcal{C}_{\text{feasible}}, q_{\text{start}}, \mathcal{G})$. We find the sequence using graph-search for which we define a *stance graph* $G_\Sigma = (V, E)$ where $V = \Sigma$ and $E = \{(\sigma_1, \sigma_2) \mid \mathcal{F}_{\sigma 1} \cap \mathcal{F}_{\sigma 2} \neq \emptyset \wedge \text{compatible}(\sigma_1, \sigma_2)\}$. For each edge of the graph (σ_1, σ_2), we can select an intermediate configuration from $\mathcal{F}_{\sigma 1} \cap \mathcal{F}_{\sigma 2}$ – indeed the set $\mathcal{F}_{\sigma 1} \cap \mathcal{F}_{\sigma 2}$ is non-empty. The intersection of the stance regions is called *transition* [11]. Any path from σ_{start} to any $\sigma \in \mathcal{G}$ corresponds to a candidate sequence of $\mathcal{P}$. We say that two stances, σ_1 and σ_2, are *compatible iff* they only differ in one foothold, i.e., σ_2 only removes/adds a foothold from/to σ_1. Note that we do not lose any valid path by restricting the candidate sequence in this way.

For the graph-search, A* with the lazy expansion is employed, see Algorithm 1. Sampling the intersection of two stance regions requires solving a nonlinear system of equations, and it is costly to test the existence of an edge in the stance graph. Therefore, possible candidate neighbours are first generated using simple criteria. The existence of an edge is properly tested only if the neighbour would be added to the open list. The following functions and predicates need to be defined as subroutines of the algorithm:

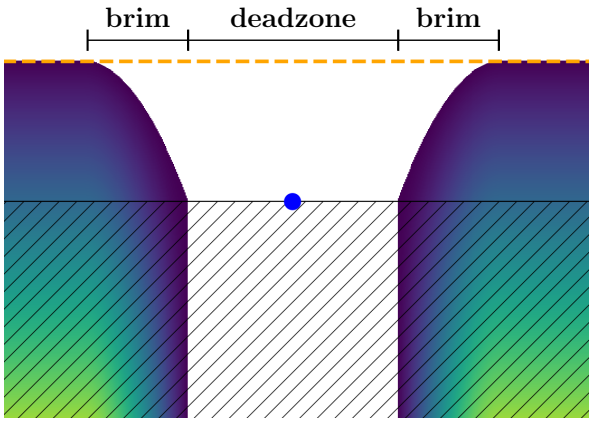


Figure 9: Collision potential function with a collision margin (orange dashed line) around a foothold (blue point) on a flat terrain showing a funnel created by the relaxation function. The *deadzone radius* is the radius of the bottom of the funnel. The *brim* is the difference between the dead zone radius and the radius of the affected neighborhood of the foothold.

- Cost function $\mathrm{f}(\sigma)$
- Heuristic function $\mathrm{h}(\sigma)$
- $\texttt{candidateNeighbours}(\sigma)$
- $\texttt{validEdge}(\sigma_1, \sigma_2)$
- $\texttt{goal}(\sigma)$

With the exception of $\texttt{candidateNeighbours}(\sigma)$ and $\texttt{validEdge}(\sigma_1, \sigma_2)$ the meaning is the same as in the regular A*. Note that if all vertices returned by $\texttt{candidateNeighbours}(\sigma)$ are valid and $\texttt{validEdge}(\sigma_1, \sigma_2)$ always returns `true`, the algorithm is equivalent to the regular A*.

5.3.1 Cost and Heuristic Function

A variety of quality measures about the candidate sequence can be optimized by selecting a suitable cost function. We may want to optimize the path length, path energy consumption, or safety. The heuristic function should be as precise a lower-bound approximation of the cost function as possible if we care about a fast and optimal solution. However, if we mostly care about finding any feasible solution, the cost function is not as important, and the heuristic should severely penalize vertices far from the goal. In our implementation, we use the following cost and heuristic functions.

- **Cost** $f : \Sigma^2 \to \mathbb{R}$ – We set the cost to 1 for each edge. That is in line with the observation that the robot's motion is most restricted when it places or lifts a leg [11]. The maneuver of lifting a foot from the terrain and losing support and the maneuver of placing a foot on the terrain violating collision margin are inherently risky. Thus limiting the number of steps is desirable arguably more than just optimizing the traveled distance.
- **Heuristic** $h : \Sigma \to \mathbb{R}$ – We use a *least squares pose* heuristic by setting

$$\mathrm{h}(\sigma) = D_{\text{scale}} \left\| \hat{\mathrm{CP}}(\sigma) - \hat{\mathbf{p}}_{\text{goal}} \right\|_2 , \tag{60}$$

Algorithm 1: A* with lazy expansion

$C \leftarrow \emptyset$
$\sigma_{\text{start}}.\text{cost} \leftarrow 0$
$O \leftarrow \{\sigma_{\text{start}}\}$
while $(Q - C) \neq \emptyset$ **do**
  $\sigma_{\text{cur}} \leftarrow \text{argmax}_{\sigma \in (O-C)}\ \sigma.\text{cost} + \text{h}(\sigma)$
  $C \leftarrow C \cup \{\sigma_{\text{cur}}\}$
  if `goal`(σ_{cur}) **then**
    END succees
  end if
  for σ_n in `candidateNeighbours`(σ_{cur}) **do**
    if $\sigma_n \notin O$ **or** $\sigma_{\text{cur}}.\text{cost} + \text{c}(\sigma_{\text{cur}}, \sigma_n) < \sigma_n.\text{cost}$ **then**
      if `validEdge`$(\sigma_{\text{cur}}, \sigma_n)$ **then**
        $\sigma_n.\text{cost} \leftarrow \sigma_{\text{cur}}.\text{cost} + \text{c}(\sigma_{\text{cur}}, \sigma_n)$
        $O \leftarrow O \cup \{\sigma_n\}$
      end if
    end if
  end for
end while
END failure

where $\hat{\mathbf{p}}_{\text{goal}}$ is the x-y goal position and $\hat{\text{CP}}(\sigma)$ represents the x-y coordinates of a base link pose $\text{CP}(\sigma)$ minimizing the squared distance of the effectors to footholds assigned by the stance in a *base joint configuration* $\boldsymbol{\theta}_{\text{default}}$ of the robot,

$$\text{CP}(\sigma) = \underset{\mathbf{p}}{\text{argmin}} \sum_{i=1}^{|\sigma|} \left\| \text{IK}_{(\mathbf{p}, \boldsymbol{\theta}_{\text{default}})}(e_i) - \sigma(e_i) \right\| . \tag{61}$$

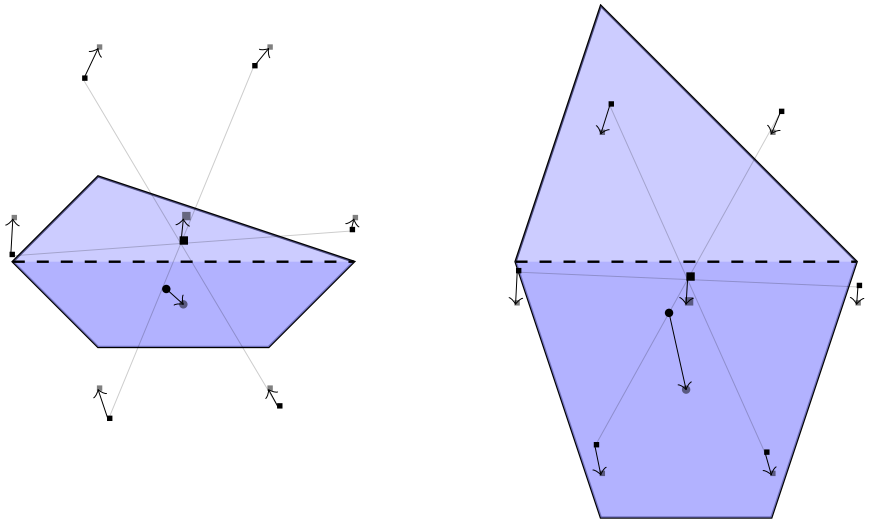
The parameter D_{scale} is a *distance scaling factor*, and with the appropriate value of D_{scale}, the distance is the simplest estimation for the number of steps to reach the goal. If we do not care much about optimality and want any feasible solution, setting large D_{scale} makes the heuristic inadmissible, however, it will drive the algorithm to the goal faster.

A simple heuristic is used by K. Hauser, *et al.* [57] that uses a centroid of the support polygon instead of the least-squares pose. The problem with the *support polygon* heuristic is that it always penalizes lifting the front legs. The heuristic (60) penalizes lifting the front legs in *stretched* stances, where it is desirable to move the rear legs. It actually rewards lifting the front legs in *squished* stances where it is desirable behavior, see Fig. 10.

5.3.2 Generating Candidate Neighbours

We need to determine possible neighboring stances of the current stances during the search for a candidate sequence. Only the stances that differ in a single foothold are considered when selecting possible neighboring stances for the current stance It gives two types of edges.

1. **Foothold removal** – For each leg whose foot tip is fixed by the current stance, we consider a stance with that foothold removed as a neighbor of the current stance. If the current stance has three footholds, we do not consider these edges as they would violate static stability.



(a) squished stance (b) stretched stance

Figure 10: Comparison of support polygon and *least squares pose* heuristics. The diagram shows change in the position of the centroid of the support polygon and the pose of the robot minimizing the sum of least square distances of effectors to footholds at the default joint configuration. Notice that the proportions of the support polygon are exaggerated to enhance the effect. (left) When lifting a front leg in a *squished* stance, the *support polygon* heuristic penalizes this step, although it is desirable. The *least squares pose* heuristic rewards this step. (right) The *least squares pose* heuristic penalizes lifting a leg in a *stretched* stance; this is a desirable behaviour.

2. **Foothold addition** – We find a *least squares pose* $\mathbf{p}$ to the current stance as in (61). For an effector that is not fixed by the current stance, and each foothold on the terrain in a sphere around the position of the effector at the configuration $(\mathbf{p}, \boldsymbol{\theta}_{\text{default}})$, we form a stance with that foothold added for that same effector, see a visual example in Fig. 11.

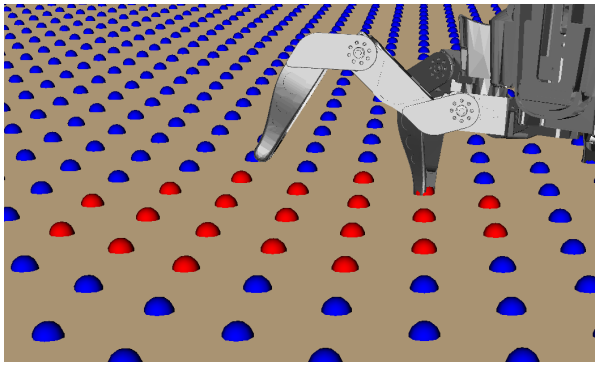


Figure 11: Footholds considered (marked red) for the *swinging* leg.

5.3.3 Transition and Goal Sampling

In order to implement the `validEdge`(σ_1, σ_2) procedure we need to test if $\mathcal{F}_{\sigma 1} \cap \mathcal{F}_{\sigma 2} \neq \emptyset$. We propose to do that by repeatedly randomly sample $\mathcal{F}_{\sigma 1} \cap \mathcal{F}_{\sigma 2}$. However, it is impossible to get such a configuration by random sampling $\mathcal{C}$. Thus, the neighborhood of $\mathcal{F}_{\sigma 1} \cap \mathcal{F}_{\sigma 2}$ is sampled, and a

descend method is used to correct the sampled configuration. If the descending method converges, the sampling was successful. Without the loss of generality, we solve only a sampling of $\mathcal{F}_{\sigma 1} \cap \mathcal{F}_{\sigma 2}$ where $|\sigma_2| = |\sigma_1| + 1$. Indeed, only stances differing in a single foothold are considered, and placing an effector is time-symmetric with the effector lifting. Let us relabel σ_1 as σ_{support} and σ_2 as σ_{target}. The constraint penalty function of $\mathcal{F}_{\sigma\,\text{support}} \cap \mathcal{F}_{\sigma\,\text{target}}$ becomes

$$F_{\mathcal{F}_{\sigma\,\text{support}} \cap \mathcal{F}_{\sigma\,\text{target}}}(q) = \begin{bmatrix} f_{\mathcal{C}_{\sigma\,\text{stance}}}(q) \\ \hat{g}_{\mathcal{S}_{\sigma\,\text{support}}}(q) \\ \hat{g}_{\boldsymbol{\theta}}(q) \\ \hat{g}_{\text{SDF}\,\sigma\,\text{stance}}(q) \end{bmatrix}. \tag{62}$$

We follow Algorithm 2 to obtain a configuration from $\mathcal{F}_{\sigma\,\text{support}} \cap \mathcal{F}_{\sigma\,\text{target}} = \emptyset$ by `SampleTransition()` procedure.

In order to examine we reach the goal, i.e., implement the subroutine goal(σ) of Algorithm 1, we need to test if $\mathcal{F}_\sigma \cap \mathcal{G} = \emptyset$. That is done using essentially the same procedure `SampleTransition()`, just the constraint function is extended by the goal equality constraint function $f_{\mathcal{G}}$ as

$$f_{\mathcal{F}_\sigma \cap \mathcal{G}}(q) = \begin{bmatrix} f_{\mathcal{C}_\sigma}(q) \\ f_{\mathcal{G}}(q) \\ \hat{g}_{\mathcal{S}_\sigma}(q) \\ \hat{g}_{\boldsymbol{\theta}}(q) \\ \hat{g}_{\text{SDF}\,\sigma}(q) \end{bmatrix}. \tag{63}$$

5.4 Finding Smooth Paths

The second step of the proposed planning method is finding smooth path for the determine candidate sequence. Let has a candidate sequence

$$q_0\ \sigma_1\ q_1\ \sigma_2\ q_2\ \ldots\ \sigma_{k+1}\ q_{k+1} \tag{64}$$

We need to find sub-paths $\pi_i : [0, 1] \to \mathcal{F}_{\sigma i}$ connecting intermediate configurations q_{i-1} and q_i to obtain a full path.

Let defines a constraint violation function of the path using the constraint penalty function of $\mathcal{F}_{\sigma i}$ as

$$F_{\mathcal{F}_{\sigma i}, q_{i-1}, q_i}(q) = \begin{bmatrix} f_{\mathcal{C}_{\sigma i}}(q) \\ g_{\mathcal{S}_{\sigma i}}(q) \\ g_{\boldsymbol{\theta}}(q) \\ g_{\text{SDF}\,\sigma i, q_{i-1}, q_i}(q) \end{bmatrix}. \tag{65}$$

Note that the collision function depends on the initial and final configurations q_{i-1} and q_i, respectively, on the stances $\mu(q_{i-1})$ and $\mu(q_i)$. It is needed to relax the collision constraint near footholds of $\mu(q_{i-1})$ and $\mu(q_i)$ that needs to be reached at the endpoints of the path.

Now, let defines the penalty of the path π_i for a given *sampling* S as

$$\mathcal{E}_S(\pi_i) = \begin{bmatrix} F_{\mathcal{F}_{\sigma i}, q_{i-1}, q_i}(\pi_i(0/S)) \\ F_{\mathcal{F}_{\sigma i}, q_{i-1}, q_i}(\pi_i(1/S)) \\ \vdots \\ F_{\mathcal{F}_{\sigma i}, q_{i-1}, q_i}(\pi_i(S/S)) \end{bmatrix}. \tag{66}$$

In order to have a valid path, it must hold that

$$\|\mathcal{E}_S(\pi_i)\|_2 < \sqrt{S}\,\epsilon, \tag{67}$$

Algorithm 2: Sample Transition

SampleTransition(σ_1, σ_2)

$\sigma_{\text{target}} \leftarrow \text{argmax}_{\sigma \in \{\sigma_1, \sigma_2\}} |\sigma|$
$\sigma_{\text{suppport}} \leftarrow \text{argmin}_{\sigma \in \{\sigma_1, \sigma_2\}} |\sigma|$
for $i \in \{1 \dots \texttt{MaxIteration}\}$ **do**
  $q \leftarrow \texttt{SampleNeighbourhood}(\sigma_1, \sigma_2)$
  $q \leftarrow \texttt{SolveNewtonRaphson}(q, F_{\mathcal{F}_\sigma[\text{support}] \cap \mathcal{F}_\sigma[\text{target}]})$
  if $f_{\mathcal{F}_\sigma[\text{support}] \cap \mathcal{F}_\sigma[\text{target}]}(q) \leq \epsilon$ **then**
    END SUCCESS
  end if
end for
END FAILURE

SolveLevenbergMarquard(q, f)

for $i \in \{1 \dots \texttt{MaxIteration}\}$ **do**
  if $\|f(q)\|_2 \leq \epsilon$ **then**
    RETURN q
  end if
  $\mathbf{v} \leftarrow f(q)$
  $\boldsymbol{\delta} \leftarrow (\mathbf{J}_f{}^T \mathbf{J}_f - \lambda E)^{-1} \mathbf{J}_f{}^T \mathbf{v}$
  if $\|f(q + \boldsymbol{\delta})\|_2 < \|f(q)\|_2$ **then**
    $q \leftarrow q - \boldsymbol{\delta}$
    $\lambda \leftarrow \lambda_{\text{drop}} \cdot \lambda$
  else
    $\lambda \leftarrow \lambda_{\text{boost}} \cdot \lambda$
  end if
end for
END FAILURE

SampleNeighbourhood(σ_1, σ_2)

$\sigma_{\text{target}} \leftarrow \text{argmax}_{\sigma \in \{\sigma_1, \sigma_2\}} |\sigma|$
$\boldsymbol{\theta}_0 \leftarrow$ Nominal Joint Positions
$\mathbf{p}_0 \leftarrow \text{argmin}_{p \in SE(3)} f_{\mathcal{C}_{\sigma_{\text{target}}}}((p, \boldsymbol{\theta}_0))$
$\boldsymbol{\epsilon} \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{W})$
RETURN $(\mathbf{p}_0, \boldsymbol{\theta}_0) + \boldsymbol{\epsilon}$

where ϵ is the desired precision. Note that $\sqrt{S}\,\epsilon$ is the value of l^2-norm of $\mathcal{E}_S(\pi_i)$ if all S samples on π_i violate the constraint by ϵ.

An alternative path cost functions considers the worst sample along the curve

$$\mathcal{E}_{S,\max}(\pi_i) = \max_{t\in\{0/S,1/S,\ldots,S/S\}} F_{\mathcal{F}_{\sigma i},q_{i-1},q_i}(\pi_i(t)) \; . \tag{68}$$

In order to have a valid path, it must hold that

$$\|\mathcal{E}_{S,\max}(\pi_i)\|_2 < \epsilon \; . \tag{69}$$

We propose to parameterize $\pi_i : [0,1] \to \mathcal{C}$ using Bézier curve of the degree d in $\mathcal{C} \equiv \mathbb{R}^n$. Thus, the Bézier curve is defined by $d+1$ control points. Let those *control points* be the column vectors of the matrix $\mathbf{P} \in \mathbb{R}^{(n,(d+1))}$. A point on the curve for a given value t can be then computed as

$$\pi_{\mathbf{P}}(t) = \mathbf{P} \cdot \mathbf{B}_d(t) \; , \tag{70}$$

where $\mathbf{B}_d(t)$ is the vector of the weights.

The advantage of Bézier curve-based parametrization is that we can control the path complexity by selecting the number of control points. The single-step motions needed to connect the intermediate configurations in a candidate sequence require a relatively simple motion for the hexapod robot with 3DOF per leg. In our experience, even a simple linear interpolation of the configurations produces a viable path. Hence, by using Bézier curves with a low number of control points, the need to use a more complex search algorithm is avoided, and it is possible to use the same constraint functions and descend methods used for transition sampling. Limiting the motion in this way may also avoid producing unnatural motions produced by sampling-based algorithms that are observed by Hauser in [11].

The proposed path smoothing procedure starts with some initial control points $\mathbf{P}_0$. In our cases, a linear interpolation of q_{i-1} and q_i is utilized. The path constraint (66) is solved using the Levenberg-Marquard algorithm depicted in Algorithm 2.

5.5 Discussion of Transition Sampling Approach

Regarding the solution of the constraints of the stances, two particular approaches have been implemented and examined. At first, only the support polygon constraint and the kinematic constraints have been considered. However, instead of formulating the contact constraint as equality, we took advantage of the fact that the inverse kinematics of the 3DOF legs of a simple hexapod robot has a closed-form solution for which, under mild assumptions, it is possible to define a unique solution. The workspace of a 3DOF leg is also straightforward to model. A closed-form expression for the distance to the boundary of the workspace of the leg was used to formulate an inequality constraint on the position of the robot's body. It is similar to the method used by D. Belter, *et al.* in [45]. Only the 6DOF pose of the robot's body had to be optimized instead of the whole 24DOF configuration. The angles for the *swing* legs were set to a default/retracted position with the legs held close to the body, such that they do not stick out, to prevent them from colliding with the terrain or shifting the center of mass.

Besides, also inspired by Belter, the *Particle Swarm Optimisation* (PSO) was used to solve the constraints. The PSO is a metaheuristic optimization algorithm based on a study of bird flocking behaviour [67]. However, we assumed that the PSO would not scale well to robots with more DOFs per leg and more complex constraint functions. Therefore, it turns out that the final approach is inspired by

Algorithm 3: Particle Swarm Optimization (PSO)

SolvePSO(f)

for $i \in \{1 \dots N\}$ **do**
  $\mathbf{x}_i \leftarrow U(b_{\text{lo}}, b_{\text{up}})$ Uniform random initialization between lower and upper bound
  $\mathbf{p}_i \leftarrow f(\mathbf{x}_i)$
  $\mathbf{v}_i \leftarrow U(-|b_{\text{lo}} - b_{\text{up}}|, |b_{\text{lo}} - b_{\text{up}}|)$
end for
$\mathbf{g} \leftarrow \text{argmin}_{\mathbf{x}_i \,;\, i \in \{1 \dots N\}} \|f(\mathbf{x}_i)\|_2$
while $\|\mathbf{g}\|_2 > \epsilon$ **do**
  for $i \in \{1 \dots N\}$ **do**
    $\mathbf{v}_i \leftarrow \omega\, \mathbf{v}_i + \varphi_p\, U(0,1)\, (\mathbf{p}_i - \mathbf{x}_i) + \varphi_g\, U(0,1)\, (\mathbf{g} - \mathbf{x}_i)$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$
    if $\|f(\mathbf{x}_i)\|_2 < \|f(\mathbf{p}_i)\|_2$ **then**
      $\mathbf{p}_i \leftarrow \mathbf{x}_i$
    end if
    if $\|f(\mathbf{x}_i)\|_2 < \|f(\mathbf{g})\|_2$ **then**
      $\mathbf{g} \leftarrow \mathbf{x}_i$
    end if
  end for
end while

Algorithm 4: Newton-Raphson

SolveNewtonRaphson(q, f)

for $i \in \{1 \dots \texttt{MaxIteration}\}$ **do**
  $\mathbf{v} \leftarrow f(q)$
  $\boldsymbol{\delta} \leftarrow \mathbf{J}_f{}^{\dagger} \cdot \mathbf{v}$
  $\alpha \leftarrow 1$
  while $\|f(q - \alpha\boldsymbol{\delta})\|_2 > \|f(q)\|_2$ **do**
    $\alpha \leftarrow \alpha/2$
  end while
  $q \leftarrow q - \alpha\boldsymbol{\delta}$
  if $\|f(q)\|_2 \leq \epsilon$ **then**
    RETURN q
  end if
end for
END failure

the Iterative Constraint Enforcement (ICE) method [52] and the optimization is based on the Newton-Raphson method (depicted in Algorithm 4) to find a solution to the whole 24DOF configuration.

Comparing the performance of the PSO and Newton-Raphson method, it has been observed that the Newton-Raphson method is faster than the PSO with the 6DOF pose when the contact constraint and support polygon constraint were considered, Admittedly, it might be caused by the need to tune PSO much more than the Newton-Raphson method. Hence, it might be possible to tune the PSO in future work as it could allow incorporating constraints that are not differentiable.

In the final revision of the implementation, the Newton-Raphson method has been replaced by the Levenberg-Marquard algorithm (Algorithm 2). Although it is necessary to tune the damping parameters λ, λ_{boost}, and λ_{drop}, it converges for more initial samples and in fewer iterations. Thus, it provides overall the best performance among the implemented optimization algorithms.

Chapter 6
HexaPlanner Framework

The proposed planning method is implemented as a C++ framework for legged robot motion planning. Modules of the framework reflect the abstractions used in motion planning. In this chapter, the most important parts of the implementation are briefly described to provide an overview of framework architecture and usage. A dependency diagram of the framework is depicted in Fig. 12.

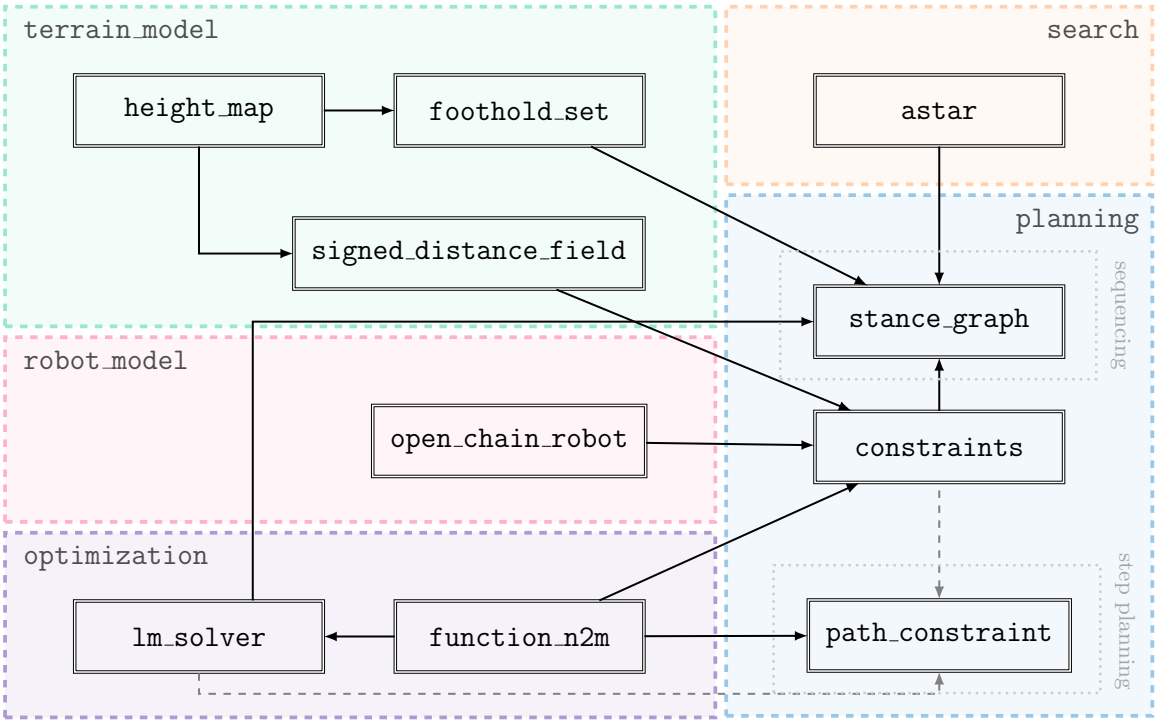


Figure 12: Dependency diagram of the HexaPlanner framework.

The framework facilitates an approach inspired by [36, 58]; however, the abstractions are general enough to be used in future research. The code and documentation are available at the GitLab repository[4]. Besides, a version of the repository as-is at the time of submission of the thesis is uploaded as an attachment of the digital submission, and it is also included on the DVD accompanying the physical copies. The framework heavily utilizes the `Eigen`[5] library for representation of vectors and matrices and its linear solvers.

[4] `https://gitlab.fel.cvut.cz/valoudav/hexaplanner`
[5] `https://eigen.tuxfamily.org`

6.1 Parts of the Developed HexaPlanner Framework

6.1.1 Terrain Model

The terrain is represented by a height map stored as a 2D grid. The used implementation has an interface for querying a height at the given x-y coordinates within the map. Bilinear interpolation is used to calculate the result of the query.

For the used Signed Distance Field (SDF), described in Sections 4.2.4 and 5.2, we used the `grid_map` library by ANYbotics [68]. The library is implemented as a package for the ROS[6]. Since we intend to use only the core module and the SDF implementation from the grid_map library; a fork of the ANYbotics's repository[7] is created, and the `grid_map_core` and `grid_map_sdf` modules were extracted to keep the implementation compact and without unnecessary dependencies.

Selecting the set of footholds is a separate problem in motion planning, and it is out of the scope of this thesis. Therefore, a straightforward implementation distributing the footholds in a grid is used. The implementation of the foothold set provides nearest neighbor and radius search capabilities. For that purpose, the `nanoflann` library implementation of the KD-Trees is employed [69].

6.1.2 Robot Model

The framework implements the `OpenChainRobot` class that represents a description of the robot comprised of open kinematic chains. It is realized as an array of `Links`. The links are stored in an order given by the topological ordering of the oriented tree isomorphic to the structure of the robot described in Section 2.3), where the *base link* is being the first. Each `Link` stores information about the index of its parent link and the isomorphism describing its pose in the frame of the parent link. The representation allows for transformations from the coordinates frames of all links at a given configuration to the global coordinate frame that can be computed in a single pass over the links in the topological order.

Additional data are stored for each link, such as mass, the position of the center of mass (COM), and the collision model. Based on the used data representation, the position of COM of each sub-tree can also be computed in a single pass.

6.1.3 Search

Templated implementation of the A* algorithm with lazy edge evaluation described in Algorithm 1 is developed. The template parameters specify the type of the graph vertex and a *graph class* implementing the subroutines of the A* algorithm listed in Section 5.3.

6.1.4 Optimization

A `RawMatrix` class is implemented for storing the value and Jacobian of a general function $f : \mathbb{R}^n \to \mathbb{R}^m$. The `Eigen::Matrix` class does not provide a way of appending matrices, which is required by the implementation. Therefore, `RawMatrix` stores the matrix as an array of variables of the `Eigen::Triplet` type. The `Eigen::Triplet` structure is used to store matrix coefficients along with their row and column indices. In `Eigen`, it is used as a precursor representation to construct sparse matrices. The representation allows adding coefficients into the matrix easily. `RawMatrix`

[6]Robot Operating System `https://ros.org`
[7]`https://github.com/ANYbotics/grid_map`

is meant as an intermediate representation for a large matrix being constructed. Before it is used in further computation, a dense matrix representation or a sparse matrix representation is constructed from the `Triplets`.

A `FunctionN2M` abstract class representing a function $f : \mathbb{R}^n \to \mathbb{R}^m$ is implemented An abstract method `append(Eigen::Vector& argument, RawMatrix& value, RawMatrix* jacobian)` is defined by `FunctionN2M`. It appends a value of the function for a given argument to a `RawMatrix` representation of the output vector. If the optional `Jacobian` argument is provided, Jacobian for the argument value is appended.

Two implementations of the Levenberg-Marquard algorithm (see Algorithm 2) are implemented. They are initialized with reference to `FunctionN2M` representing the set of solved equalities.

The first implementation is using a dense matrix representation and the `Eigen::PartialPivLU` linear solver. The second one is using a sparse matrix representation and the `Eigen:SparseQR` linear solver.

6.1.5 Planning

A `StanceGraph` class implementing the subroutines of the A* algorithm (Section 6.1.3) is provided. A `Stance` structure is used to represent the vertices of the *stance graph*. More specifically, the vertices are stored as `std::shared_ptr<Stace>`.

The constraints as described in Section 5.2 are realized as implementations of the `FunctionN2M` interface.

The path constraint (66) from Section 5.4 is in an implementation of the `FunctionN2M` interface. It is initialized with a reference to an instance of `FunctionN2M` implementing the constraint function of the mode (65).

Chapter 7
Results

The empirical evaluation of the proposed free-gait planning for the hexapod walking robot has been performed in a testing scenario motivated by [9] that is depicted in Fig. 4a for a comparison. In that setup, the robot has to cross a gap using two *beams* with *holes* in them. However, the scenario is solvable using a *terrain-aware* crawling gait as demonstrated in [9]. In this thesis, we target more challenging scenarios where not all legs can find a foothold in the gait-defined stance and support phases. Therefore, a more challenging virtual experimental setup has been created, where we replace one of the beams with a *stepping stone* with only a single foothold. The problem requires precise motion planning and a solution of the sequencing part to determine the suitable sequence of stance and swing phases. Furthermore, we modify the scenario to create an even more challenging planning problem with a wider gap to be passed.

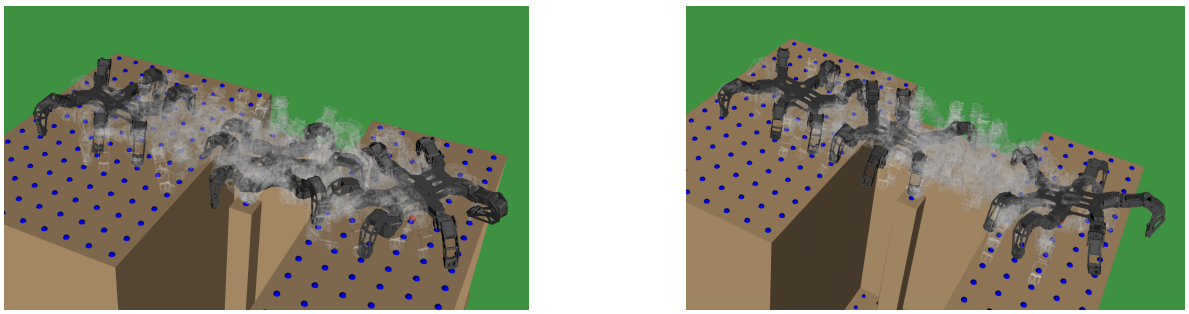
The herein presented results consist of the overview of the initial solution implemented to validate the early ideas of the proposed method. In Section 7.2, the results of the proposed method as described in Chapter5 are reported to demonstrate the feasibility and computational requirements of the method further. All the presented results have been achieved with C++ implementation running on the computational environment with the Intel i7-8565U running at 4.6 GHz accompanied with 32 GB RAM.

7.1 Early Results on the Proposed Planner

Early implementation of the proposed planner [70] slightly differs from the final method because later improvements showed to be more efficient. In particular, the following parts differ.

1. **Newton-Raphson method** is used for constraints solving instead Levenberg-Marquard.
2. **Distance from the centroid of support polygon** is used as the heuristic instead of the distance of the *least squares pose* (Section 5.3.1).
3. **Candidate footholds are chosen using 2D criteria** not 3D, thus the footholds in the "*valley*" in the presented scenario are considered, albeit there might not be suitable.
4. **Alternative path constraint** (68) described in Section 5.4 is used.

The employed computational kinematic model of the hexapod walking robot is the same as in [9].



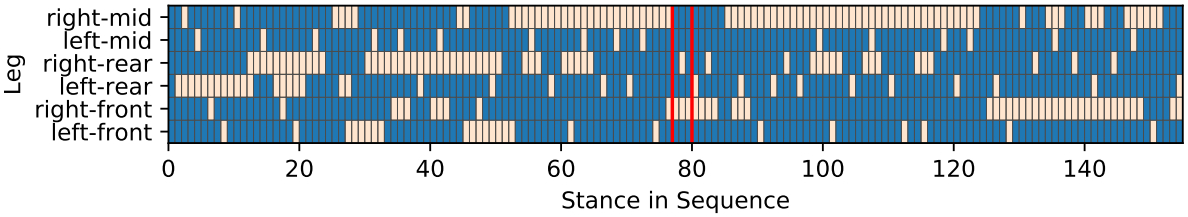
(a) Narrow gap scenario (b) Wide gap scenario

Figure 13: Visualization of the evaluation scenarios and the respective candidate sequences.

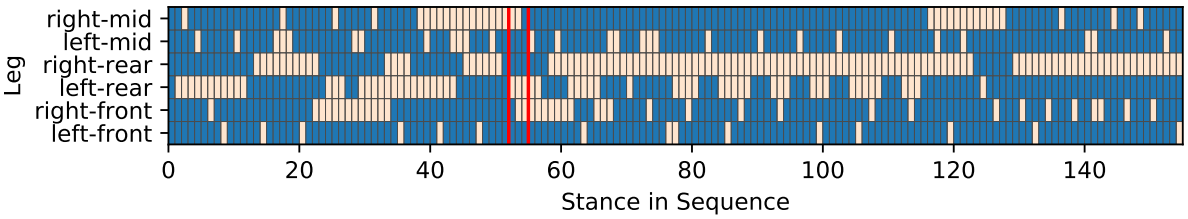
Table 1: Sequencing Results - Early Implementation

Scenario	D_{scale}	**Time** [min]	**No. of Stance Expansions**	**Sequence Length**
Narrow	1000	55	4677	155
Wide	1000	31	3375	155

A summary of the results on the sequencing part of the planning is depicted in Table 1. It lists the planning time, the number of expanded stances, and the length of the resulting sequence. The heuristic scaling factor $D_{\text{scale}} = 1000$ is used in both cases (narrow and wide scenarios). The contact diagrams representing which legs are assigned to a foothold by individual stances in the found sequences are visualized in Fig. 14.

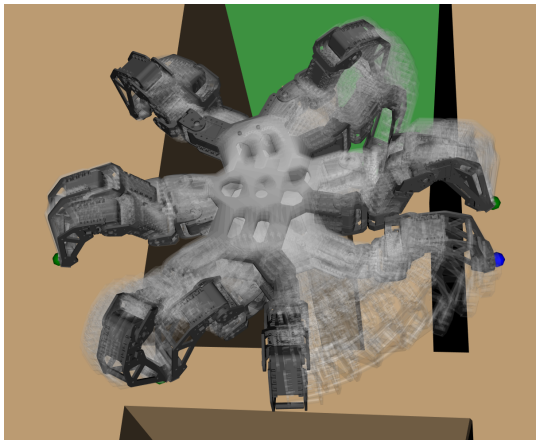


(a) Narrow gap scenario

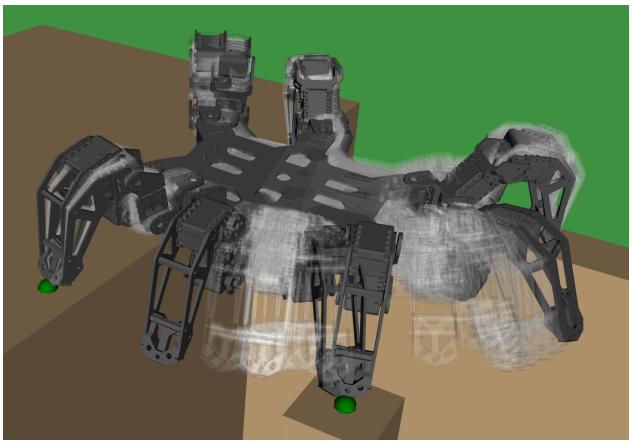


(b) Wide gap scenario

Figure 14: Leg contact diagram in narrow and wide gap scenarios. Blue cells signify the leg is in the stance state.



(a) Single step in the narrow gap scenario



(b) Single step in the wide gap scenario

Figure 15: Single step parametrized as a Bézier curve in the configuration (joint) space.

One “interesting” step is selected in each sequence to demonstrate the proposed step planning with the Bézier curve parametrization. The stances at which the step is taken and the two adjacent stances

are marked in the contact diagram in Fig. 14. In the narrow gap scenario, the step is chosen for the situation where the right-rear leg is moved between two distant footholds, see Fig. 15a. In the wide gap scenario, the chosen step is when the robot switches the right-front leg for the right-middle leg on the stepping stone, shown in Fig. 15b. The planning results for the two described steps are summarized in Table 2 showing the maximal degree of the Bézier curve, the number of iterations of the Newton-Raphson algorithm, the planning time, and the used tolerance ϵ.

Table 2: Step Planning Results - Early Implementation

Scenario	**Max Degree**	**No. of Iterations**	**Time** [sec]	ϵ [m]
Narrow	5	747	4.0	0.001
Wide	6	1465	8.3	0.001

Discussion

The feasibility of the proposed planning has been validated in the evaluation scenarios, successfully showing it is possible to execute motion without an explicitly prescribed gait. The gait-free property can be observed in the diagram of the individual legs' contact with the ground. Notably, in the scenario with the wide gap, the right-middle leg is placed on the *stepping stone* for a significant portion of the sequence. The right-rear leg is not in any foothold for most of the sequence, demonstrating the gait-free capability of the hexapod robot. The sequencing part is computationally demanding because each sequence induces expensive sampling of the intermediate configurations needed to validate the graph edges. However, the performance can be improved by a more informed heuristic and employing fast rejection of infeasible edges. On the other hand, the single-step planning using low-capacity path parametrization shows to be a suitable choice. Even the steps with a substantial difference between the initial and the final configurations have been successfully determined up to the specified precision.

The achieved results motivated us to improve the implementation further to reduce the computational burden in the sequencing part of the planning. The results of the improved implementation are reported in the following section.

7.2 Results on the Improved Implementation of the Planner

Based on the early results, we improved the implementation of the identified parts of the proposed planning to decreasing the computational burden of the demanding parts. The most notable improvements are as follows.

1. **Levenberg-Marquard method** shows to be more efficient for solving the constraints than the Newton-Raphson method. Levenberg-Marquard method requires setting the damping strategy [71]. However, it converges much faster than Newton-Raphson and makes the convergence less sensitive to the initial sample. As a result, we can use lower the maximum allowed iterations and the maximum number of samples tested before an edge is rejected in Algorithm 2, without discarding too many feasible edges.

2. **Least squares pose** is used as the heuristic (Section 5.3.1).

3. **Candidate footholds are chosen based on the radius search in the 3D** and thus, some infeasible edges in the stance graph are no longer tested.

4. **Path constraint** (66) is used (Section 5.4).

The computational kinematic model is of the SCARAB II hexapod walking robot [72] with the same morphology the robot used in Section 7.1; thus, it does not affect the computational requirements. In addition to the *narrow gap* scenario and the *wide gap* scenario, the planner was tested on an artificial cave floor (Fig. 16). This demonstrates the ability to plan on non-flat terrain.

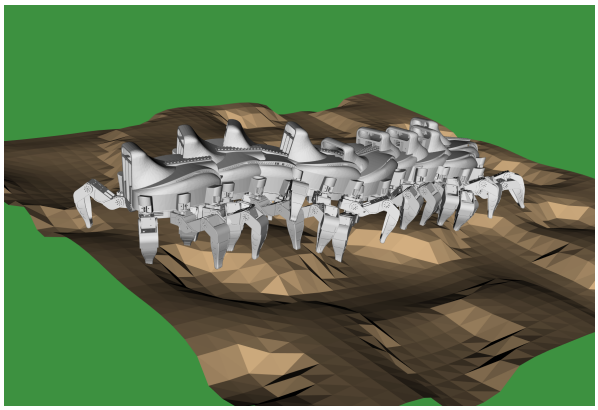


Figure 16: Artificial cave floor scenario, with the planned sequence visualized.

The used heuristic and the strategy for candidate neighbor selection are the parts that most likely significantly influence the combinatorial complexity of the sequencing part. Therefore, the sequencing has been tested with both the heuristic based on the centroid of the *Support Polygon* (refereed SP) and the new implementation using the *Least Squares Pose* (denoted LSP) heuristic. A summary of the results is presented in Table 3 that lists the planning time, the number of expanded stances, and the length of the resulting sequence. The heuristic scaling factor $D_{\text{scale}} = 1000$ is used in both cases.

Table 3: Sequencing Results - Improved Implementation

Scenario	**Heuristic**	D_{scale}	**Time** [sec]	**No. of Stance Expansions**	**Sequence Length**
Narrow	SP	1000	74	1811	122
Narrow	LSP	1000	25	579	101
Wide	SP	1000	204	2341	121
Wide	LSP	1000	68	773	107
Cave	SP	1000	107	1551	98
Cave	LSP	1000	35	615	79

The reported results in Table 3 show significant computational improvements in comparison to Table 1, and the implemented improvements yield the expected results. The LSP heuristic improves the performance, and the results support the hypothesis about its properties presented in Section 5.3.1.

Chapter 8
Discussion of Further Extensions

The herein addressed precise motion planning can be considered a challenging problem where the computational requirements are increased because of the sequencing part of the free-gait locomotion. The presented results show that the proposed approach is feasible. Furthermore, the improved implementation also yields a significant reduction of the computational requirements. Based on the achieved results, several directions for further extensions can be identified. In addition to improvements in consolidating the developed C++-based HexaPlanner framework, there are possible generalizations towards constraints, path optimization, and possible further speedups. Selected ideas are briefly discussed in the following paragraphs.

Optimization of Joints Force/Torque

So far, the force/torque limits are not considered during planning. Since the configuration does not uniquely define the ground reaction forces resulting in static stability, they are difficult to compute and incorporate into our approach using the differentiable constraints. However, considering those constraints would allow us to plan using footholds on inclined or even vertical surfaces, e.g., scenarios like climbing in a vertical shaft using the friction of the effectors. Thus, generalizations of the constraints to respect the force/torque constraints during the optimization would solve such scenarios.

Generalized Stance Graph

In the stance graph definition presented in Section 5.3, we restricted ourselves to consider two stances *compatible* when they differ in a single foothold. This definition is simple and does not discard any solution. However, we can define a more general stance graph, where vertices still correspond to stances $V = \Sigma$, but we redefine the set of edges in the following way.

The edge (σ_1, σ_2) is in E if two conditions are satisfied.

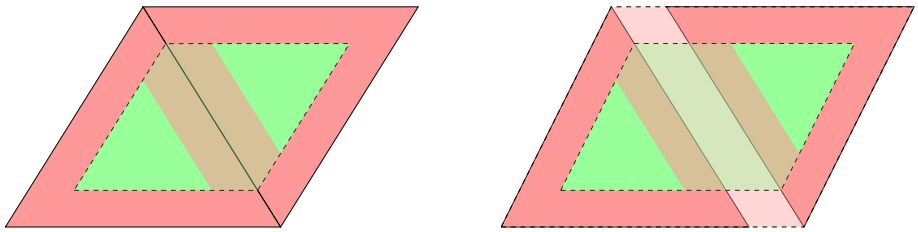
1. For each effector e it holds that if If $\sigma_1(e)$ and $\sigma_2(e)$ are defined then$\sigma_1(e) = \sigma_2(e)$, which allows us to define the *stance combination operator* $\sigma_1 \oplus \sigma_2$:

$$(\sigma_1 \oplus \sigma_2)(c) = \begin{cases} \sigma_1(c) & \sigma_1(c) \text{ is defined} \\ \sigma_2(c) & \text{otherwise} \end{cases} .$$

2. There exists $q_1, q_2 \in \mathcal{C}_{\text{feasible}}$ such that

$$q_1 \in \mathcal{F}_{\sigma 1} \cap \mathcal{F}_{\sigma_1 \oplus \sigma_2}$$
$$q_2 \in \mathcal{F}_{\sigma_1 \oplus \sigma_2} \cap \mathcal{F}_{\sigma 2} .$$

Note that if (without loss of generality) σ_2 only adds footholds to σ_1 then $\sigma_1 \oplus \sigma_2 = \sigma_2$. In such a case, we only need to find $q \in \mathcal{F}_{\sigma 1} \cap \mathcal{F}_{\sigma 2}$. So far, we are considering edges between stances differing in a single foothold (Section 5.3). Thus, the definition of the stance graph based on [11] is a special case of a more general definition.



(a) adjoint (b) disjoint

Figure 17: (left) Two support polygons with adjoint stability regions but with disjoint stability regions when considering stability margin. (right) Two support polygons with disjoint stability regions.

The advantage of the generalized stance graph is demonstrated in Fig. 17. Two intermediate configurations q_1 and q_2 are checked, because it is easily possible for two stances σ_1 and σ_2 to have disjoint stability regions, but those respective stability regions are both included in the stability region of the combination $\sigma_1 \oplus \sigma_2$.

Path Optimization using Stance Graph Expansion and n-convexity

As discussed in Section 5.4), the motion needed to execute a single step is relatively simple and can be successfully approximated by a low-degree polynomial curve. Using the restricted parametrization of the path, we can produce a *sensible* motion without the need for post-processing; as noted by Hauser in [11], where sampling-based methods can produce unnatural, *jerky* motions. So far, in preliminary evaluations, using randomized sampling produces the intermediate configurations and yields more *natural-looking* poses by restricting the joint ranges. Thus, it might be interesting to compute a single-step path (Section 5.4) and sample the transition (Section 5.3.3) at the same time. It is expected that a more sparse sampling of the curve would be needed, and the endpoint of the curve would need to be constrained to lay in the transition between two stances. The resulting intermediate configuration might be more "*reasonable*", as it would be an endpoint of the smooth motion starting from a "*reasonable*' configuration.

We also find it interesting to elaborate more on the mathematical properties of the sets that can be navigated using low-degree polynomial curves. To this end, we propose a notion of $\mathbf{n}$-convexity as follows. A set A is n-convex *iff* every pair of points $\mathbf{a}_1, \mathbf{a}_2 \in A$ is connected by a polynomial curve of the degree n such that the whole curve lays in A. Note that for $n = 1$, the definition gives us a regular convexity. The assumption about the simplicity of the single-step motion use in the presented approach is an assumption that the stance regions $\mathcal{F}_\sigma$ can be approximated by an n-convex set.

Potential Search Speedup

It is known that one of the fastest statically stable motions possible with a hexapod robot can be achieved using the *tripod gait* that might be used for speeding up the search policy. In the restricted support graph setting, where it is only possible to lift/place a single leg, one step of a tripod gait would correspond to six edges in the graph.[8] At the start and the end of such a sequence, the robot is at the stance with three stance legs. The support polygon is a triangle in that case. Now, let us label the stance at the start of the step σ_1 and the stance at the end of the step σ_2; notice that the stance of the size 6 in the *middle* of the sequence is $\sigma_1 \oplus \sigma_2$. Therefore, one step of the tripod gait corresponds

[8] Placement of the three swinging legs one-by-one and lift the next three swinging legs one-by-one.

to a single edge in the generalized stance graph, compared to six edges in the restricted stance graph currently used in the proposed planner. Let us consider the time complexity of the graph-search query in the generalized stance graph compared to the restricted stance graph.

First, let examines the case of the restricted stance graph as a baseline reference. The time complexity of a graph search depends on the *branching factor* and the *solution depth* as

$$T_{\text{restrict}} \in \mathcal{O}\left(B^D\right).$$

Note that the branching factor is proportional to the number of available footholds in the robot's vicinity at a given stance.

Let us express the branching factor of the generalized stance graph and the solution depth in the generalized graph. In any edge $(\sigma, \sigma_{\text{new}})$ leading from σ, we have to assign footholds to at most three legs. Since no leg assigned as stance in σ can be assigned a new foothold in σ_{new}, and each stance has to assign footholds to at least three legs to remain statically stable. Given there are about B available footholds in the neighborhood of the robot, there are $\mathcal{O}(B^3)$ potential σ_{new}. For the case of the tripod gait, the solution depth may be up to six times lower than in the restricted graph. Let the depth be $\mathcal{O}(\frac{D}{R})$, where R is a reduction factor between one to six. Time complexity for the generalized graph can be expressed as

$$T_{\text{general}} \in \mathcal{O}\left(\left(B^3\right)^{\left(\frac{D}{R}\right)}\right) = \mathcal{O}\left(B^{\left(3\frac{D}{R}\right)}\right).$$

Now let us express the ratio of T_{general} and T_{restrict}:

$$\frac{T_{\text{restrict}}}{T_{\text{general}}} \in \mathcal{O}\left(\frac{B^D}{B^{\left(3\frac{D}{R}\right)}}\right) = \mathcal{O}\left(B^{D\left(1-\frac{3}{R}\right)}\right).$$

Thus, if the reduction R is greater than three, $T_{\text{general}}/T_{\text{restrict}}$ would grow to ∞ polynomially with B and even exponentially with D. Hence, we can expect improvements in the computational requirements.

Chapter 9
Conclusion

In this thesis, we present a planning framework for hexapod walking robots based on a decomposition into the sequence determination and sequence validation parts. The developed locomotion planning does not rely on a specific locomotion gait prescribing how the legs are alternating in swings. It provides the ability to exploit the advantage of the multi-legged platforms to traverse environments with only a few footholds. In the sequencing stage, a candidate sequence of stances with intermediate configurations provides a necessary condition heuristic on feasible paths connecting the stances in the sequence. In the sequence validation stage, finding smooth paths connecting the intermediate configurations is attempted considering the robot's motion constraints. A feasible solution is provided if a complete path is found that can be directly executed by the robot effectors.

The proposed path parametrization based on Bézier curves used in single-step planning showed to be a viable approach. The curve optimization finds a path even for steps with a substantial difference between the initial and final configuration. Besides, the realization of the path yields motion that is smooth and natural-looking. On the other hand, we found out that convergence of the numeric solver for the path constraint exhibit inconsistencies that deserve further investigation because the focus of the thesis is on the graph-based sequencing of the robot's stances.

The computational requirements of the developed sequencing procedure have been significantly lower in comparison to the early implementation. The solver provides candidate sequences needed in planning while considering all the constraints, including collisions with the terrain. Levenberg-Marquard algorithm is proposed to solve the constraints that showed to be a more robust descend algorithm than the Newton-Raphson method. The search has been accelerated by a more refined heuristic function and an improved strategy for selecting candidate edges. The used heuristic function exhibit improved performance compared to heuristics used in similar approaches found in related work. Since the computational requirements are relatively low, we can consider the developed precise motion planner as a free-gait locomotion planner because it allows to use arbitrary sequences of the legs swings.

The proposed planner is implemented in C++ as the modular and extendable framework called Hexa-Planner. In addition to further generalizations and possible speedup of the planning, the natural next step is to validate the found paths using real hexapod walking robots.

References

[1] A. Orlando, "After disaster strikes, a robot might save your life," 2020, cited on 2021-04-12. [Online]. Available: https://www.discovermagazine.com/technology/after-disaster-strikes-a-robot-might-save-your-life

[2] "The Center for Robot-Assisted Search and Rescue (CRASAR)," cited on 2021-08-11. [Online]. Available: http://crasar.org

[3] "Humanitarian Robotics and AI Laboratory at Texas A&M University," cited on 2021-08-11. [Online]. Available: http://hrail.crasar.org/

[4] T. Rouček, M. Pecka, P. Čížek, J. Bayer, V. Šalanský, D. Heřt, M. Petrlík, T. Báča, V. Spurný, F. Pomerleau, V. Kubelka, J. Faigl, K. Zimmermann, M. Saska, T. Svoboda, and T. Krajník, "Darpa subterranean challenge: Multi-robotic exploration of underground environments," in *2019 Modelling and Simulation for Autonomous Systems (MESAS)*, 2020, pp. 274–290.

[5] A. Bouman, M. F. Ginting, N. Alatur, M. Palieri, D. D. Fan, T. Touma, T. Pailevanian, S.-K. Kim, K. Otsu, J. Burdick *et al.*, "Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2518–2525.

[6] R. Buchanan, L. Wellhausen, M. Bjelonic, T. Bandyopadhyay, N. Kottege, and M. Hutter, "Perceptive whole-body planning for multilegged robots in confined spaces," *Journal of Field Robotics*, vol. 38, no. 1, pp. 68–84, 2021.

[7] G. Wiedebach, S. Bertrand, T. Wu, L. Fiorio, S. McCrory, R. Griffin, F. Nori, and J. Pratt, "Walking on partial footholds including line contacts with the humanoid robot atlas," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 1312–1319.

[8] A. Escande, A. Kheddar, and S. Miossec, "Planning contact points for humanoid robots," *Robotics and Autonomous Systems*, vol. 61, no. 5, pp. 428–442, 2013.

[9] P. Čížek, D. Masri, and J. Faigl, "Foothold placement planning with a hexapod crawling robot," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4096–4101.

[10] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, "Footstep planning for autonomous walking over rough terrain," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2019, pp. 9–16.

[11] K. Hauser, "Motion planning for legged and humanoid robots," Ph.D. dissertation, University of Illinois, 2008.

[12] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual review of control, robotics, and autonomous systems*, vol. 1, pp. 159–185, 2018.

[13] T. Bretl, S. Lall, J.-C. Latombe, and S. Rock, "Multi-step motion planning for free-climbing robots," in *Algorithmic Foundations of Robotics VI*. Springer, 2004, pp. 59–74.

[14] Z. Kingston, A. M. Wells, M. Moll, and L. E. Kavraki, "Informing multi-modal planning with synergistic discrete leads," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3199–3205.

[15] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 2008.

[16] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[17] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *Departmental Papers (ESE)*, p. 323, 1992.

[18] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[19] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[20] J. Faigl and J. Mačák, "Multi-goal path planning using self-organizing map with navigation functions," in *European Symposium on Artificial Neural Networks (ESANN)*, 2011, pp. 41–46.

[21] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[22] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic a," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.

[23] M. Likhachev and A. Stentz, "R* search," in *AAAI Conference on Artificial Intelligence*, 2008, pp. 344–350.

[24] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[25] S. M. LaValle, *Planning Algorithms*. Cambridge university press, 2006.

[26] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[27] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep. 98-11, 1998.

[28] A. M. Ladd and L. E. Kavraki, "Measure theoretic analysis of probabilistic path planning," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 229–242, 2004.

[29] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *2000 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2000, pp. 521–528.

[30] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[31] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (bit*): Informed asymptotically optimal anytime search," *The International Journal of Robotics Research*, vol. 39, no. 5, pp. 543–567, 2020.

[32] R. Luna, M. Moll, J. Badger, and L. E. Kavraki, "A scalable motion planner for high-dimensional kinematic systems," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 361–388, 2020.

[33] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[34] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[35] L. Jaillet and J. M. Porta, "Path planning under kinematic constraints by rapidly exploring manifolds," *IEEE Transactions on Robotics*, vol. 29, no. 1, pp. 105–117, 2012.

[36] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1325–1349, 2008.

[37] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 897–915, 2010.

[38] J. Mirabel, S. Tonneau, P. Fernbach, A.-K. Seppälä, M. Campana, N. Mansard, and F. Lamiraux, "Hpp: A new software for constrained motion planning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 383–389.

[39] M. A. Diftler, J. Mehling, M. E. Abdallah, N. A. Radford, L. B. Bridgwater, A. M. Sanders, R. S. Askew, D. M. Linn, J. D. Yamokoski, F. Permenter *et al.*, "Robonaut 2-the first humanoid robot in space," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 2178–2183.

[40] J. S. Matthis and B. R. Fajen, "Humans exploit the biomechanics of bipedal gait during visually guided walking over complex terrain," *Proceedings of the Royal Society B: Biological Sciences*, vol. 280, no. 1762, p. 20130700, 2013.

[41] J. Faigl and P. Čížek, "Adaptive locomotion control of hexapod walking robot for traversing rough terrains with position feedback only," *Robotics and Autonomous Systems*, vol. 116, pp. 136–147, 2019.

[42] R. Szadkowski, M. Prágr, and J. Faigl, "Self-learning event mistiming detector based on central pattern generator," *Frontiers in Neurorobotics*, vol. 15, p. 5, 2021.

[43] J. Hurst, "Walk this way: To be useful around people, robots need to learn how to move like we do," *IEEE Spectrum*, vol. 56, no. 3, pp. 30–51, 2019.

[44] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[45] D. Belter, P. Łabęcki, and P. Skrzypczyński, "Adaptive motion planning for autonomous rough terrain traversal with a walking robot," *Journal of Field Robotics*, vol. 33, no. 3, pp. 337–370, 2016.

[46] M. A. Arain, I. Havoutis, C. Semini, J. Buchli, and D. G. Caldwell, "A comparison of search-based planners for a legged robot," in *9th International Workshop on Robot Motion and Control*. IEEE, 2013, pp. 104–109.

[47] N. Perrin, C. Ott, J. Englsberger, O. Stasse, F. Lamiraux, and D. G. Caldwell, "Continuous legged locomotion planning," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 234–239, 2016.

[48] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, 2018.

[49] J. Norby and A. M. Johnson, "Fast global motion planning for dynamic legged robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3829–3836.

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

[51] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[52] K. Hauser, T. Bretl, and J.-C. Latombe, "Non-gaited humanoid locomotion planning," in *2005 5th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2005)*. IEEE, 2005, pp. 7–12.

[53] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," in *2005 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2005, pp. 629–634.

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

[55] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz, "Anytime search-based footstep planning with suboptimality bounds," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 2012, pp. 674–679.

[56] K. Hauser, T. Bretl, J.-C. Latombe, and B. Wilcox, "Motion planning for a six-legged lunar robot," in *The Seventh International Workshop on the Algorithmic Foundations of Robotics*, vol. 7. Citeseer, 2006, pp. 16–18.

[57] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1325–1349, 2008.

[58] P. Vernaza, M. Likhachev, S. Bhattacharya, S. Chitta, A. Kushleyev, and D. D. Lee, "Search-based planning for a legged robot over rough terrain," in *2009 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 2380–2387.

[59] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[60] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[61] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks (ICNN)*, vol. 4. IEEE, 1995, pp. 1942–1948.

[62] D. Belter, "Efficient modeling and evaluation of constraints in path planning for multi-legged walking robots," *IEEE Access*, vol. 7, pp. 107 845–107 862, 2019.

[63] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtree: A hierarchical structure for rapid interference detection," in *23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 171–180.

[64] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of computing*, vol. 8, no. 1, pp. 415–428, 2012.

[65] N. Das and M. Yip, "Learning-based proxy collision detection for robot motion planning applications," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1096–1114, 2020.

[66] F. Alizadeh and D. Goldfarb, "Second-order cone programming," *Mathematical programming*, vol. 95, no. 1, pp. 3–51, 2003.

[67] S. Sengupta, S. Basak, and R. A. Peters, "Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 157–191, 2019.

[68] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, A. Koubaa, Ed., 2016, ch. 5.

[69] J. L. Blanco and P. K. Rai, "nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees," https://github.com/jlblancoc/nanoflann, 2014, cited on 2021-08-12.

[70] D. Valouch and J. Faigl, "Gait-free planning for hexapod walking robot," in *European Conference on Mobile Robots (ECMR)*, 2021, (to appear).

[71] M. Lampton, "Damping–undamping strategies for the levenberg–marquardt nonlinear least-squares method," *Computers in Physics*, vol. 11, no. 1, pp. 110–115, 1997.

[72] M. Forouhar, P. Čížek, and J. Faigl, "SCARAB II: A Small Versatile Six-legged Walking Robot," in *5th Full-Day Workshop on Legged Robots, ICRA 2021*, 2021.

Appendix A
Attachements

The following is the list of attachements to this thesis:

1. DVD containing a `.pdf` of this thesis and a `.zip` with the `HexaPlanner` repository.