DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics

Digital Nermo - Development of a Digital Twin for the Exploration & Evaluation of Tendon Driven Compliant Legs & Spine on the Locomotion Behaviour of the Bio-Inspired Robotic Mouse Nermo

Alex Rohregger

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics

Digital Nermo - Development of a Digital Twin for the Exploration & Evaluation of Tendon Driven Compliant Legs & Spine on the Locomotion Behaviour of the Bio-Inspired Robotic Mouse Nermo

Digital Nermo - Entwicklung eines digitalen Zwillings für die Erforschung und Bewertung von sehnengetriebenen, nachgiebigen Beinen und der Wirbelsäule auf das Fortbewegungsverhalten der bioinspirierten Robotermaus Nermo

Author:	Alex Rohregger
Supervisor:	Prof. Dr.-Ing. habil. Alois Knoll
Advisor:	Dr. rer. nat. Zhenshan Bing
Submission Date:	December 3, 2021

I confirm that this master's thesis in robotics is my own work and I have documented all sources and material used.

Munich, December 3, 2021 Alex Rohregger

Acknowledgments

First of all, I want to thank Prof. Dr.-Ing. habil. Alois Knoll for his supervision and providing this exciting opportunity to write my master thesis at the Chair of Robotics, Artificial Intelligence and Real-time Systems. Additionally, I want to express my utmost gratitude to Dr. rer. nat. Zhenshan Bing for his support throughout every aspect of this thesis: starting with the selection and direction of the topic down to the last details of the robot experiments and thesis submission. Dr. rer. nat. Zhenshan Bing continuously provided invaluable feedback and was always willing to share his time and effort to help. He also continuously motivated me to work harder and helped me challenge my assumptions to achieve the best possible results. Thank you so much for everything - this thesis could not have been possible without you. I also want to thank Yuhong Huang for his continued help during the in-person robot tests.

Finally, I want to thank my parents for their unconditional support throughout my academic journey. Their support has not only allowed me to pursue my passions but kept me going when things got tough. Thank you!

Abstract

Biomimetic robotic systems help provide valuable information about the creatures they are derived from. In particular, biological research relying on rodent models has benefited from the development of biomimetic rodent robots. However, current rodent robots are limited in two key aspects. First, they do not incorporate multiple spine functions within a unified legged control architecture. Second, to the best of our knowledge, no simulation of rodent robots exists to support research and development especially given the growing popularity and benefits of simulators in learning-based algorithms.

This thesis introduces a novel control architecture that combines current legged control architectures with a spine controller. The spine controller handles three spine-based functions that enhance the capabilities of bio-inspired rodent robots beyond classical legged locomotion: 1) spine-based gait extension that increases the effective stride length, 2) spine-based turning, and 3) spine balance compensation for static stability. In addition, we present a high-fidelity simulation (digital twin) of the bio-inspired robotic rodent Nermo. We evaluate the performance of the novel control architecture, focusing on the spine functions, on four test case scenarios using the digital twin. In addition, we compare the simulation results against a subset of experiments conducted with the robot rodent Nermo. We successfully demonstrate that: 1) the spine-based functions increase the gait and turning speed of Nermo compared with naïve legged locomotion, and 2) the high-fidelity digital twin accurately simulates the behaviour of the real-world robot rodent. The results enable the future use of the high-fidelity digital twin as a training platform for novel learning-based algorithms and bio-inspired sensors.

Contents

1. Introduction

Biological inspiration is frequently used in robotics to produce biomimetic components and new control models for biomorphic robotic platforms [1], [2]. Conversely, bio-inspired approaches in robot design are opening up new possibilities for robotics applications in biological research. In other words, biomimetic robotic systems can help provide valuable information about the creatures they are derived from.

Neuroscience is an example of a field of research that uses biomimetic robotic systems for research, mainly to understand the neural systems that drive motor activity, navigation, perception, and social behavior [3]. For this purpose, rodents, particularly mice and rats, have emerged as helpful animal models [4], [5]. In this context, robotic systems with biomimetic mechanisms can be used as the physical models to test neuroscience hypotheses - and many times in scenarios not possible with animal models [3], [6]. For instance, researchers have been studying the interaction between robotic and biological rodents to understand rodent behavior with imitation learning [7], [8], explore analogies between optical flow and the dynamical aspects of tactile sensing of whiskers, and test models of vertebrate navigation [9]–[11]. Therefore, it can be concluded that there is a demand for bio-inspired robotic systems to support research in the field of neuroscience and neurobiology.

Researchers have been conducting studies on biologically inspired rodent robots to provide state-of-the-art robotic platforms to support neuroscience and neurobiological research [7], [8], [12]–[15]. Much of the current research has focused on developing rodent robots that mimic the biological features of rodents individually or in isolation. For example, the rodent robots in [7], [14] focus only on legged locomotion with rigid bodies, while the rodent robots in [12], [16] focus on the rodent spine but use wheels for locomotion. However, rodent bodies are compliant, none-rigid, and rely on their full range of biological features for locomotion and behavior. Treating the rodent features in isolation or abstraction helps simplify the robotic system design but removes the crucial constraints and functionalities that influence the response and behavior of the biological system that the robotic system tries to mimic. Therefore, bio-inspired rodent robots should attempt to mimic the complexity of rodents closely.

This gap in bio-inspired rodent robotics has been filled by previous work conducted at our lab with the development of the bio-inspired robotic rodent Nermo [17]. Nermo is, to the best of our knowledge, the only robot that mimics all major biological features of rodents due to several unique design features: 1) the use of compliant joints instead of discrete joints, 2) 2 DOF front and rear legs (underactuated compared with biological rodents), 3) tendon driven knee joints, spine, and tail, and 4) a multi-jointed compliant spine capable of actuating in the sagittal and frontal plane.

Complimentary to the use of real robots, simulation engines play an ever-increasing role in robotics-based research [18], [19]. Simulators allow for a fast and low-cost approach to testing what is not possible with physical robots. Furthermore, simulators provide an excellent platform for speeding up the development of robots by factors of 100 to 1000 [18]. The use of simulators for accurate physical simulations has been made possible by the continuous improvement of computing performance [18], [19]. Modern simulators are capable of modeling complex biological components like muscles and tendons. However, none of the researchers working on bio-inspired rodents have also implemented a high-fidelity digital twin of their robots.

In this thesis, we fill the gap of simulation models for robotic rodents by developing a complete software stack for the robotic rodent Nermo. First, we develop a novel model-based control architecture that combines legged controllers and a spine controller to take advantage of the flexible spine. The spine controller will handle three spine-based functions inspired by nature: 1) spine-based stride extension, 2) spine-based turning, and 3) spine-based balance compensation. Second, we develop a high-fidelity, modular digital twin of Nermo. Third, we test the novel controller using the high-fidelity digital twin. Lastly, we validate the digital twin results by conducting comparative real-world experiments on a modified version of Nermo.

We have organized the thesis as follows. In Chapter 2 we cover the foundations of rodent gaits and the use of the spine. Then we provide an overview of quadruped robotics with a focus on spine-based and rodent-like quadrupeds. We end the chapter with a look at robotic simulation. In Chapter 4 we provide a short overview of the three core components of Nermo: 1) the mechanical design, 2) the actuation system, and 3) the electrical design. We detail the design process of the digital twin in Chapter 5. The chapter begins with the derivation of the mathematical models that describe Nermo. We end the chapter by presenting a novel control architecture that combines a legged locomotion controller with a spine controller. We evaluate the performance of our models, controller, and digital twin in Chapter 6. Chapter 6 also includes comparative experiments between the simulation and the real-world robotic rodent Nermo. In Chapters 6 and 7 we discuss the results and their implications. Finally, we conclude the thesis and outline our ambitions for future work in Chapter 8.

2. Background

This chapter provides the reader with the background information on the three topics core to developing the novel control architecture and digital twin of the rodent robot. First, we cover the related biological knowledge of rodent locomotion, including their gaits and the functionalities of the spine. Second, we introduce the field of quadruped robotics with a specific focus on the design and control of small quadruped and semi-quadruped robots with flexible spines. In the last section, we look at modern robotic simulators and their advantages and challenges.

2.1. Biological Foundation of Rodents

A fundamental skill of rodents is coordinated legged locomotion, which can be described by unique gait cycles and gait patterns. Rodents use all four limbs for locomotion which we classify as the left forelimb (LF), right forelimb (RF), right hindlimb (RH), and left hindlimb (HL). During an ideal gait cycle, the limbs alternate between stance state (in contact with the ground) and swing state (not in contact with the ground). Gait cycles are categorized into gait types depending on the leg states: a) walking gaits by a single leg in swing state, b) trot gaits by two diagonal leg pairs in swing state, and c) bound gaits by three or four legs in swing state [20]. Consequently, the gait cycles of each limb have a phase offset relative to the other limbs that is unique to the gait type as illustrated in the Hildebrand [21] style gait patterns in Figure 2.1. The ratio of stance phase time to the total gait cycle time is defined as the duty factor [21]. Gait cycles with duty factors greater than 0.5 are classified as walking gaits [22], [23], while gaits with duty factors less than 0.5 are classified as running gaits [24]. Therefore, the duty factor and phase offset (illustrated in Figure 2.1) are functional parameters to define different gait types mathematically.

Rodents change their locomotion speed by adjusting the gait frequency and stride length, which leads to the different gait duty factors and leg phase offsets. Walking gaits (lateral sequence walk or walking trot) naturally occur between gait speeds of 20 cm/s to 50 cm/s and running gaits (running trot/bound) at gait speeds between 50 cm/s to 70 cm/s [22], [23]. An experimental gait analysis conducted by [22], [25] shows that the gait speed depends on the gait frequency. The gait frequency ranges from 2.4 Hz (slower gaits) to 8.0 Hz (faster gaits), with the mean gait frequency of 3.7 Hz occurring for walking gaits. The switch from walking to trot happens abruptly, and rodents often switch between the gait types during walking [6]. Furthermore, [22] and [23] show that the gait speed increases with increasing stride length. It can be concluded that rodents

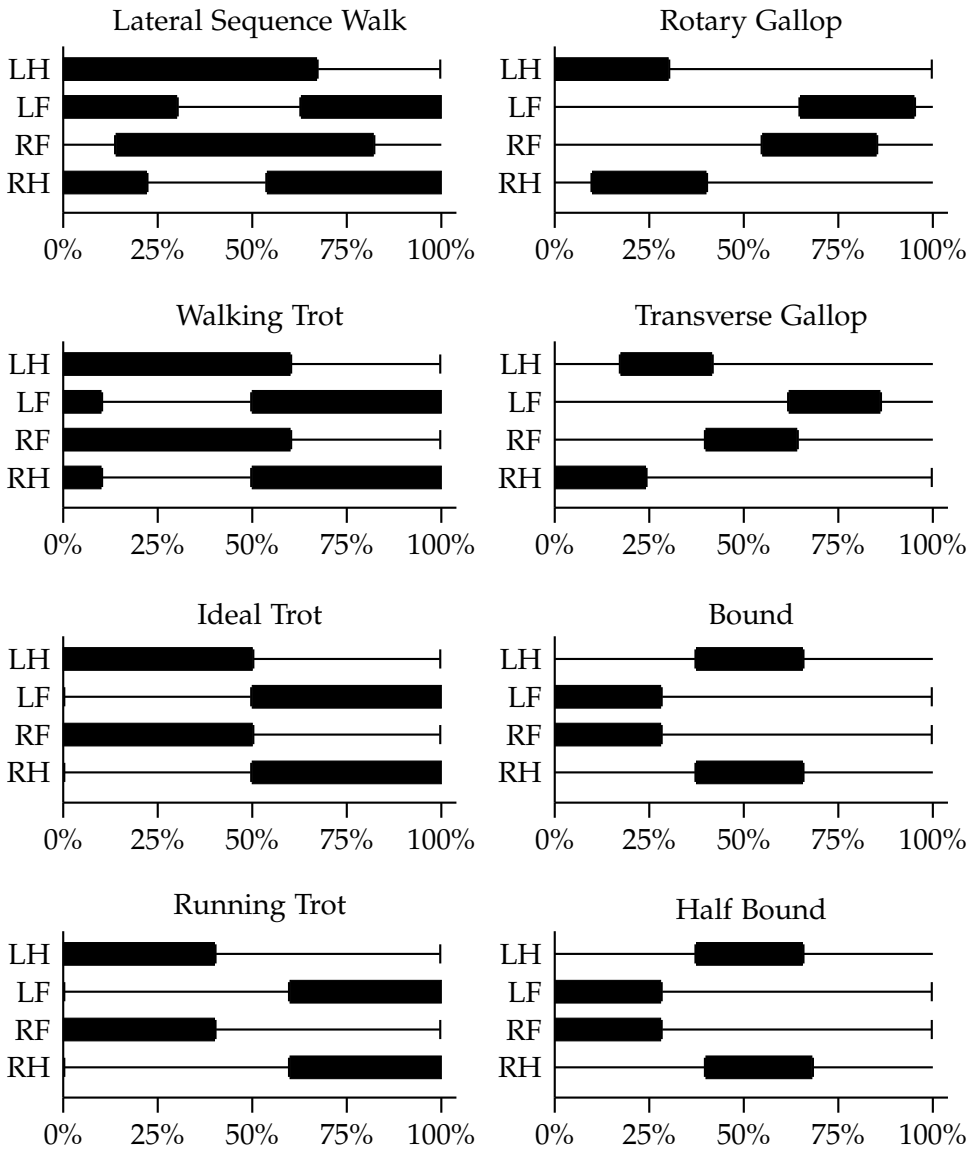


Figure 2.1.: Overview of animal gaits in the style of Hildebrand [21]. The solid blocks indicate the stance phase, and the lines indicate the swing phase. The gaits on the left are symmetrical, the gaits on the right are asymmetrical. Rodents do not exhibit gallop type gaits.

have two main mechanisms for gait speed regulation, gait frequency (at lower speeds) and stride length (at higher speeds), which naturally results in a change of the duty factor and leg phase offset [23], [24].

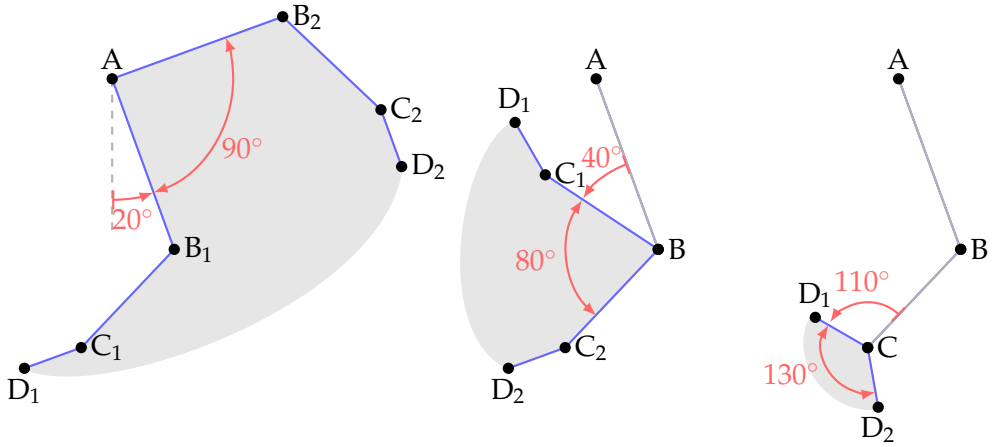
Rodents can perform a wide variety of gait types due to the range of motion of their limbs and body weight distribution. The angular range of motion of the forelimbs is greater than that of the hindlimbs. Table 2.1 lists the typical ranges of motion of the forelimbs and hindlimbs of rodents for walking and trot gait types. Figure 2.2 supplements Table 2.1 with a visualization of the ranges of motion of the forelimbs and hindlimbs. Similarly, the duty cycles between the forelimbs and hindlimbs are different; the duty factor of the hindlimbs is greater than that of the forelimbs [22]. However, rodents transfer more contact force onto their forelimbs during stance, such that the total contact force transmitted during a complete gait cycle is approximately equal across the forelimbs and hindlimbs. Therefore, from the range of motion and weight distribution data, it can be concluded that rodents use the hindlimbs to support the body during gait, and the forelimbs provide the necessary contact dexterity.

Table 2.1.: Overview of the angular range of motion of the fore- and hindlimbs in rodents for symmetrical gaits (walking/trot). (A) marks the hip/shoulder joint, (B) the knee/elbow joint, (C) the ankle/wrist joint, and (D) the foot-end point. The segment AB is the upper leg, BC the lower leg, and CD the foot. Data for the range of motion adapted from [6], [26], [27].

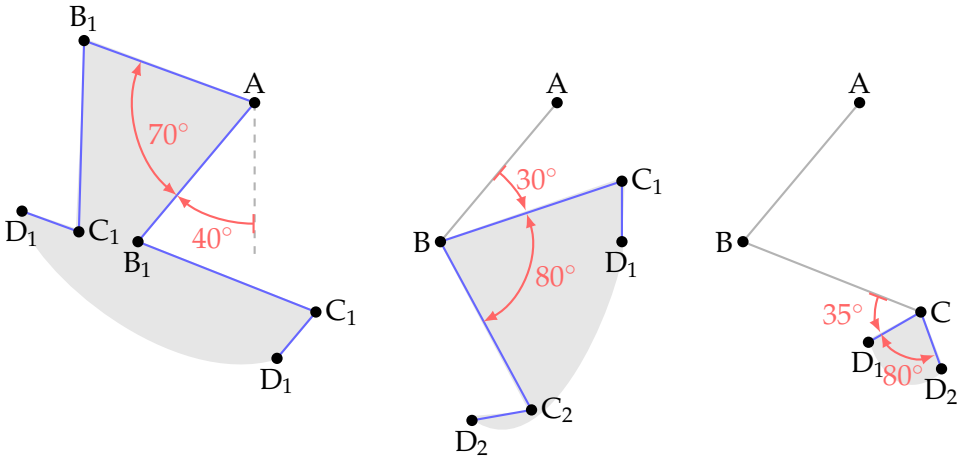
Joint	Min [°]	Max [°]	Range [°]
Hip	40	110	70
Knee	30	110	80
Ankle	35	125	90
Shoulder	20	110	90
Elbow	40	120	80
Wrist	110	240	130

The contact dexterity of the forelimbs is a result of forelimb movement along the frontal plane around the rotation axis of the shoulder, and in the lower forelimb, which improves gait performance [27]. In particular, the non-planar scapula movement contributes to gait balance and stride extension that would not be possible with only the motion of the upper and lower forelimb elements. Further motion data analysis suggests that long-axis rotation of the lower forelimb enables rodents to fine-tune their foot placement which improves the contact stability [27]. Therefore, realistic rodent movement of the forelimbs cannot be replicated by only considering the two-dimensional motion along the sagittal plane.

Similar to the non-planar movement of the scapula, rodents depend on the movement of the pelvis for forward locomotion. The pelvis movement is caused by flexion of the spine along the sagittal plane with joint amplitudes between 3° to 12° [26]. Furthermore,



(a) Forelimb range of motion of rodents for walking and trot gaits.



(b) Hindlimb range of motion of rodents for walking and trot gaits.

Figure 2.2.: Illustration of the forelimb and hindlimb range of motion in rodents.

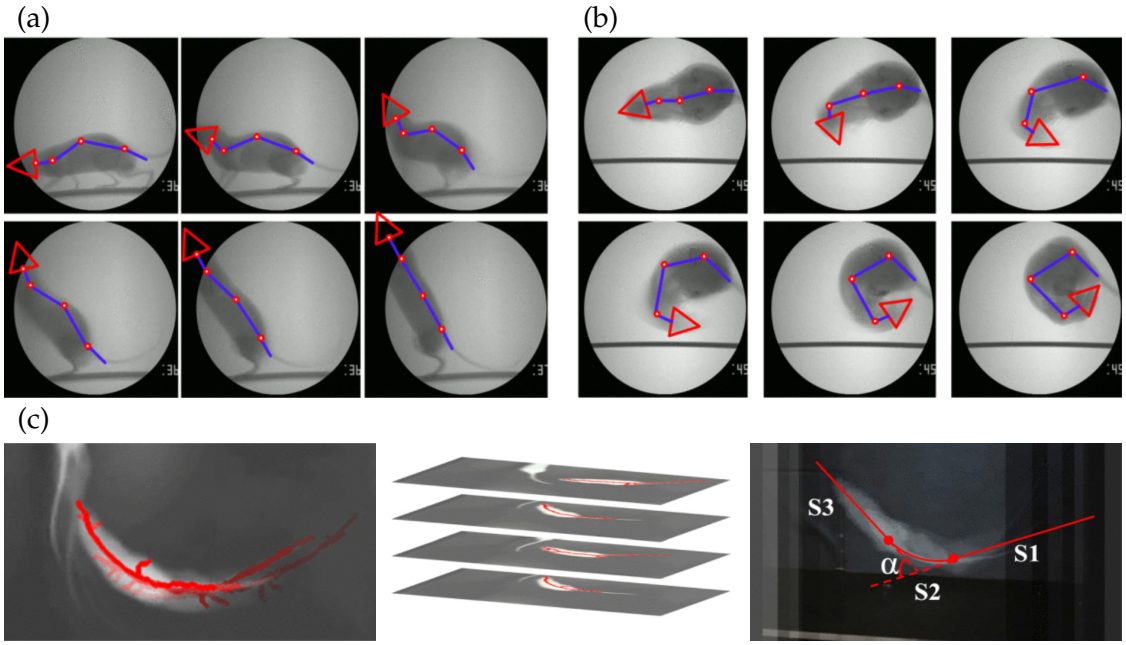


Figure 2.3.: Image analysis of rodents and the use of their spines. (a) Rodent using spine to stand up and extend their bodies. To approximate sagittal plane spine movement, a spine model would require four joints. (b) In-place turning behavior of rodents. The turning behavior can be approximated by three joints enabling full lateral flexion. (c) Shows the analysis of walking turning behavior across four time steps. $S1$ denotes the starting orientation vector, $S3$ the end orientation vector, and α the turning angle. All figures have been adapted from [12], [16].

Fischer et al. [26] discovered that the sagittal plane spine movement contributes approximately half of the total forward movement in the faster asymmetric gait types (bounding, galloping). It can be concluded that rodents use the sagittal plane spine movement as an effective mechanism for stride length extension to increase the locomotion speed in the faster asymmetrical gait types.

In developing their rodent robots, Shi et al. [12] discovered, from video analysis, that rodents also rely on their spines to achieve vertical movement and increase reach (see Figure 2.3). The video analysis confirms that rodents use the vertical movement of the spine for forward motion by extending the effective stride length. In addition, rodents flex their spine along the sagittal plane to climb obstacles and increase their reachable height by up to 1.6 times their body-length [12]. One body-length unit refers to the distance from the mouse snout to the hip in a relaxed state. Shi et al. [12] concluded that the sagittal plane movement of the rodent spine could be accurately approximated, for example, in a robot design, as a series connection of four joints (see Figure 2.3(a).

The spine of a rodent not only facilitates vertical motion, but the added flexibility of the spine is crucial for the turning motion in rodents [12], [16]. In a standstill position, rodents can touch their hips with their snouts with full lateral spine flexion in the frontal plane, which corresponds to a minimal bending distance of 0 body-length and turning radius of 0.2 body-length [12]. During forward motion, the body of a rodent follows a smooth line of action between the starting orientation vector $S1$ until the target orientation vector $S3$, facilitated by the spine [16] (see Figure 2.3(c)). Similar to the four joint approximation of the vertical motion of the spine, the turning movement of the spine can be approximated by a series connection of three joints [12]. In conclusion, the flexibility of their spine allows rodents to achieve small turning radii and efficient turning during forward motion.

2.2. Quadruped Robotics

Inspired by legged animals in nature, research into quadruped robotics has been a growing field of research over the last decades. One reason for the growth in the research of this field comes from the breakthroughs in legged locomotion and control achieved by Raibert et al. [28], [29]. Another interest in quadruped robots is their theoretical ability to traverse human-made terrain, such as stairs, and rough, elevated terrain. In practice, such capabilities are not trivial to achieve for legged robotics with current state-of-the-art research aiming to achieve robustness in traversing rough and unknown terrain [30], [31]. Nevertheless, the advancements in legged robotics have made it possible to develop increasingly performant bio-inspired legged robots.

2.2.1. Mechanical And Actuator Design

The extensive research into larger quadruped robots, typically inspired by canines, has resulted in the standardized rigid body quadruped robot design seen in the MIT Cheetah [31], ANYmal [32], and Boston Dynamic Spot [33], to name a few examples (see Figure 2.4). The limbs of these quadrupeds have three active degrees of freedom (DOF) that allow: 1) hip abduction and adduction, 2) hip flexion/extension, and 3) knee flexion/extension. The legs are actuated by direct-drive or series-elastic actuators that contain a brushless DC motor and gearbox. While all the actuators in [31] and [33] are located at the hip and drive the knee joint using a second gearbox, the knee actuator in [32] is located directly at the knee, allowing for a greater range of motion compared with the robots in [31], [33].

Compared with the larger quadruped robots, the design of smaller quadrupeds is challenging due to the limited availability of compact and performant actuators and the constrained form factor. Nevertheless, there have been successful attempts to develop smaller quadruped robots [7], [14] with compromises made in the leg design and power density of the actuators. A simplified leg design such as in [7], which has two active DOF for the hip and knee, requires less design because only two actuators are needed per

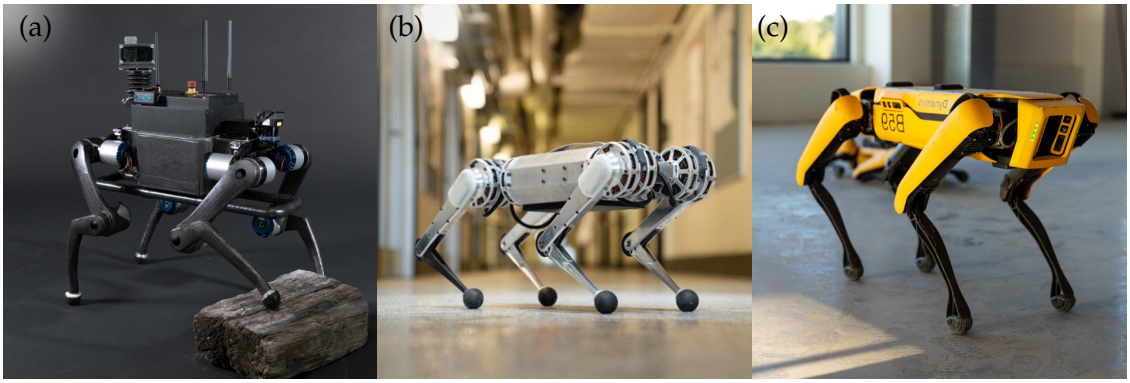
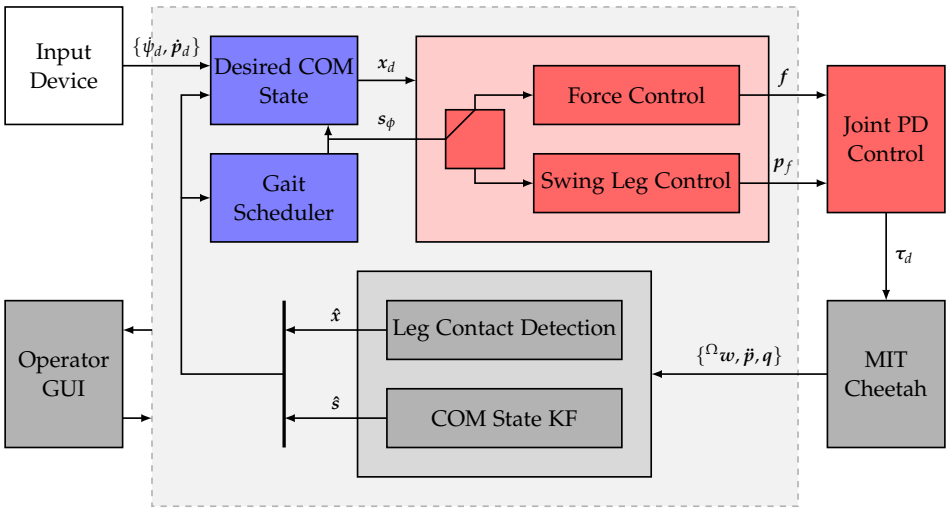


Figure 2.4.: (a) Depiction of ANYmal quadruped robot [32], (b) depiction of the MIT Cheetah quadruped [31], and (c) depiction of the Boston Dynamic Spot quadruped [33].

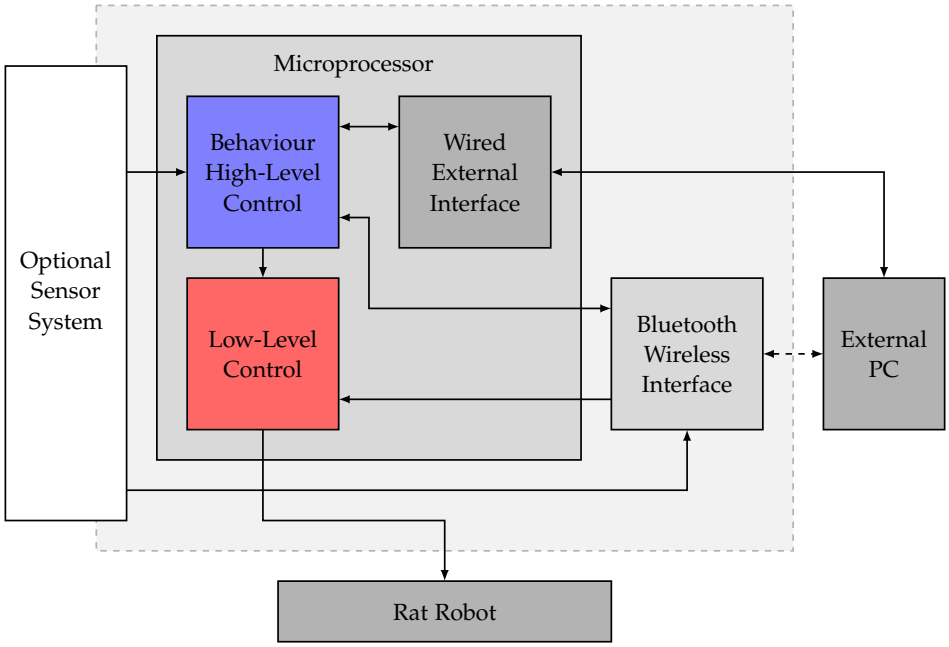
leg. However, compared with the three active DOF per leg of the larger quadrupeds, the simplified leg design is underactuated because it lacks the third active DOF responsible for hip abduction and adduction. The rodent robot in [14] has three active DOF per leg. However, it uses actuators with lower power density to fit within the design space, which is reflected in the gait performance compared with the larger quadrupeds. Therefore, small quadruped robots require design compromises, such as underactuated leg designs or lower performance actuators, resulting in decreased agility compared with larger quadrupeds.

2.2.2. Controller Architecture

Research in recent years [7], [31], [34] has aligned on a control architecture consisting of two core control modules (see Figure 2.5(a)): 1) a high-level controller, and 2) a low-level controller. The high-level controller is responsible for behavior control, such as gait selection and global position control. In scenarios with complex terrains, the high-level commands and trajectories are usually calculated offline [30], while for more straightforward terrain scenarios, such as flat surface traversal, the commands are computed online. The high-level controller commands are sent to the low-level controller responsible for foot-end positioning, leg state handling, and motor control. Although often integrated within the low-level controller, there exist control architectures with a separate hardware-level controller that performs motor PD-control, force control, and inverse dynamics [30], [34]. The hardware-level controller is typically a real-time capable microprocessor capable of running control loops in real-time and at greater frequencies than the microprocessors used in the high and low-level controller [34]. To conclude, modern control architectures divide the core control functionalities across different modules to modularize and improve the efficiency of the controllers.



(a) Control architecture of the MIT Cheetah



(b) Control architecture of the rat robot in [7]. This controller outsources major compute to an external computer.

Figure 2.5.: Comparison of two control architectures. The blue functions blocks are high-level controller elements and the red function blocks are the low-level controller elements.

Additionally, modern control architectures rely on sensors for feedback on the outcome of the control actions on the environment and quadruped position to optimize the next set of control actions. This form of control architecture is known as closed-loop control and improves the robustness of control architectures against external and unexpected environmental noise. Typical sensors of feedback loops include the inertial mass unit for body pose estimation performed by the high-level controller and positional encoders in the legs for motor control performed by the low-level controller. Kalakrishnan et al. [30] also propose, in addition to the inertial mass unit, the use of an external motion capture system to better predict robot pose and position. However, such systems do not scale well beyond experimental lab setups. Therefore, for complex scenarios with physical effects that are difficult to model, consider, or outright unknown, closed-loop control provides a design tool for improving the robustness of the control architecture.

The classical control architectures discussed in this section are model-based. In essence, kinematic and dynamic models of the agent are used to compute the actuator control values. An alternative to model-based control is the black-box learning-based control approach. However, due to the black-box characteristics of the learning-based algorithms, these controllers provide limited insights into the kinematics and dynamics of the systems they control, making troubleshooting difficult and resulting in unrealistic control actions. There have also been attempts to combine model and learning-based control in a hybrid-control architecture [35]. A third control variant for legged robots is a central pattern generator (CPG). CPG-based gait control is a bio-inspired control approach that simulates the extensor and flexor neurons to periodically output servo control values [36]. CPG control also does not require kinematic and dynamic models for control. Therefore, within the scope of this thesis, we focus on model-based control to fully understand the kinematic and physical properties of the to-be-controlled agent.

2.2.3. Electrical Design

Current legged robotics can be categorized into two electrical design philosophies: fully integrated computing and outsourced computing. The larger quadrupeds integrate all the electronics and main compute within the robot [30]–[32]. For example, the quadruped ANYmal has a total of three Intel NUC compute units that perform the computations for robot control [32]. In contrast, Laschi et al. [7] and Patane et al. [8] outsourced the major compute operations to an external computer that communicated over a wireless protocol with the rodent robot to achieve a smaller design footprint (see Figure 2.5(b)). Concretely, the external computer calculates high-level control commands while the onboard microcontroller controls the low-level hardware components. Therefore, outsourcing control operations to an external computer is a practical design choice for achieving smaller form factor quadrupeds at the disadvantage of increased latency from wireless communication.

2.2.4. Quadruped Robots with Flexible Spines

The quadruped robots covered up until now use rigid or semi-rigid actuator systems to drive the legs. Furthermore, these quadrupeds are limited to leg-based actuation. However, as pointed out by [6], [12], [26], rodents and other animals use the flexibility of their spine to support dynamic motion and achieve faster gaits. With these benefits in mind, there is also a field of robotics designing and investigating robots with compliant legs and articulated spines which more closely mimic their bio-inspired counterparts. Robots with articulated spines leverage the extra DOF to increase the performance of turning and locomotion. The quadruped robots that fall under this section are most relevant to the bio-inspired quadruped Nermo. We, therefore, cover some of these small quadruped robots in the following sections.

Spine Actuation for Turning

One class of small quadrupeds uses a single direct-drive motor that replicates a flexible spine to mimic the turning behavior of rodents. The Waseda Rat 2 (WR-2) [13] (see Figure 2.6(a)), and the evolution of the rat robot in [7] with a flexible spine by [8] (see Figure 2.6(c)) are two examples of rodent robots with a single direct-drive spine. Table 2.2 provides a short overview of the size, weight, and DOF of the different rat robots. The single joint direct-drive spine leads to a turning behavior like articulated steering in automobiles and heavy construction equipment than natural rodents. However, even such a simple design helps to drastically improve the ability for rodent robots to navigate small enclosures [8].

One method to improve spine turning to mimic the behavior in biological rodents better and improve the performance is the addition of a second actuator. The spine of the WR-5M rodent robot in [12], in contrast to the rodent robots in [14]-[8], consists of two direct-drive, series-connected servos for lateral spine flexion. The additional DOF of the WR-5M spine enables the robot to perform more complex turning motions and achieve a turning radius of 0.2 body-length, which the authors also validated with real-world tests. Therefore, using additional actuators and joints in the spine, like in the WR-5M, allows rodent robots to achieve a turning behavior closer to that of natural rodents than the abstracted articulated steering.

The increased complexity of the WR-5M spine also leads to increased modeling and control complexity. The WR-5M spine is modeled as a series connection of revolute joints (similar to a multi-joint manipulator), which is actuated such that the rodent robot achieves a target turning velocity and turning angle α. The turning angle α is the angle between the starting orientation S_1 of the rodent and target orientation vector S_3 (refer back to Figure 2.3(c)). The authors introduce the bio-inspired phase-shift turning strategy to control the multi-joint spine [16]. The phase-shift turning strategy is an angular interpolation technique to ensure that the joint displacement between two spine links allows for a sequential joint transition between the $S1$ and $S3$ orientation given a target turning velocity. In theory, the multi-link manipulator can theoretically achieve

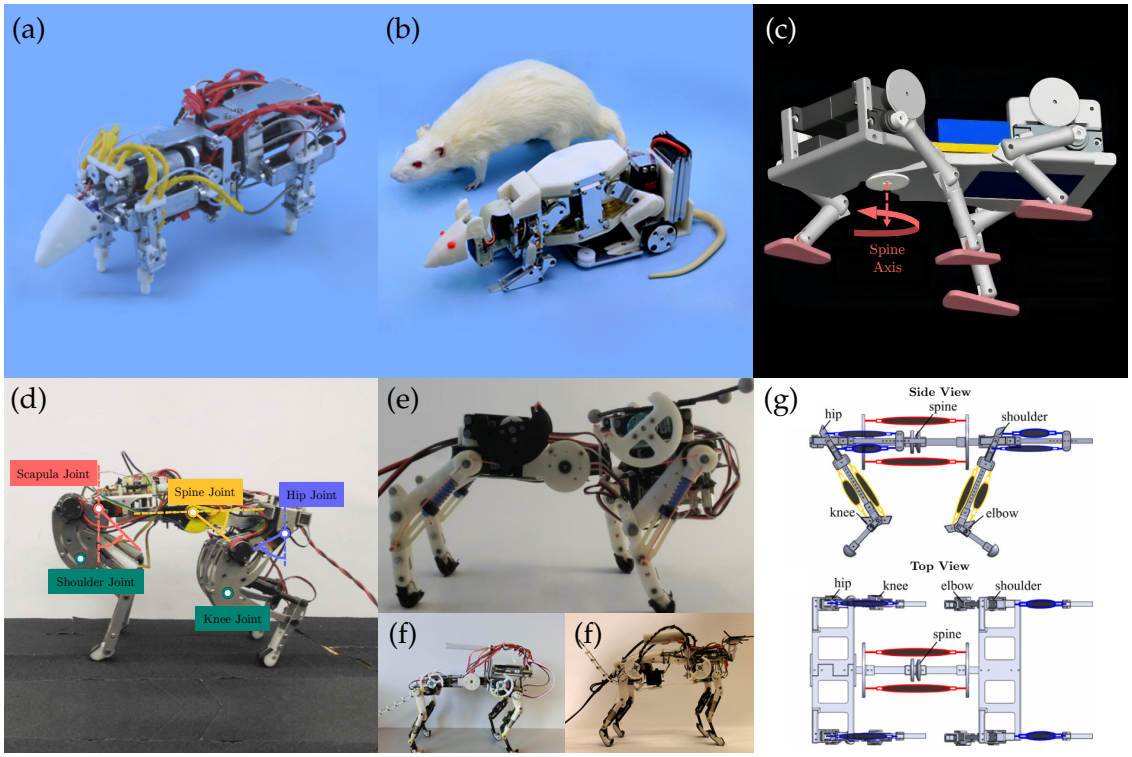


Figure 2.6.: Depiction of small quadruped robots with flexible spine. (a) WR-2 [13] quadruped robot with one DOF in lateral flexion. (b) WR-5M [12] leg-wheel hybrid robot with multi-DOF spine. (c) Robot rat by [8] with single spine joint for yaw turning. (d) Quadruped with single tendon driven spine joint for vertical flexion [37]. (e) Bobcat quadruped with direct drive vertical spine flexion [38], and (f) Lynx with multi-joint tendon driven vertical flexion spine [39] (both from EPFL). (g) Schematic of quadruped with flexible spine and pneumatic muscle actuators [40].

Table 2.2.: Overview of the general mechanical parameters of the WR- 2 [13], WR-5M [12], and rat robot in [8].

Metric	WR-2	WR-5M	Rat Robot[8]
Size LxWxH [mm]	240x70x90	210x70x80	174x68x55
Total DOF	15	13	9
DOF of Spine	2	5	1
DOF of Legs	3	2	2
Weight [g]	850	600	370
Battery [mAh]	1320	1320	NA

similar turning radii to those of biological rodents. However, due to mechanical limits of the joints, the spine joints cannot always achieve the target angle α and are instead limited by a threshold angle α_T. In scenarios where $\alpha > \alpha_T$, the authors propose an N-Step ($N \geq 2$) turning strategy that subdivides the turning angle α into N single steps. Figure 2.7 visualizes the turning behavior of the mouse for a single and a dual-step turning scenario. The two-step turning process enabled the mouse to achieve turning angles of 94° with a standard deviation of 8°. Therefore, a multi-step control algorithm is one viable solution for the control of a multi-actuator robotic spine.

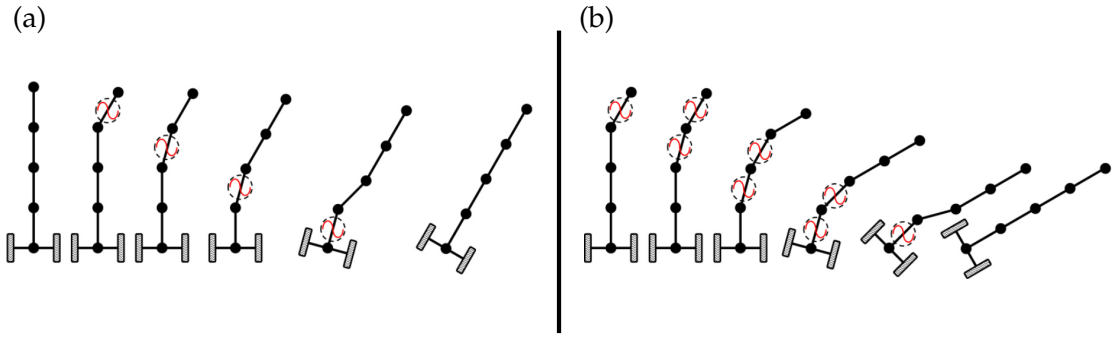


Figure 2.7.: Depiction of the bio-inspired phase-shift turning method. (a) Phase-shift turning for a single step. (b) Phase-shift turning for a two step process. Figure has been adapted from [16].

One drawback of the phase-shift turning strategy in [16] is that it does not consider the leg kinematics for control. The reason for this is that the WR-5M is a wheel-leg hybrid, where wheels replace the hindlimbs to provide active support and locomotion. Therefore, the phase-shift turning strategy did not need to consider leg kinematics. However, when applied to a similar legged rodent robot design [36], the spine and controller did not provide similar performance values. The quadruped version could walk at a maximum speed of 0.1 m/s and achieve a turning radius of 0.5 body-length (0.12 m) compared with the WR-5M, which achieved a maximum speed of 1.5 m/s and turning radius of 0.2 body-length. Therefore, we observe that the performance of the spine with two active DOF and phase-shift turning strategy cannot achieve the same wheel-leg hybrid performance with a quadruped rodent robot.

Spine Actuation for Forward Locomotion

The flexible spine in animals, including rodents, improves forward locomotion, which has prompted several researchers to replicate the spine behavior for similar benefits in small quadruped robots. For example, Kawasaki et al. [37] (see Figure 2.6(d)), Khoramshahi et al. [38] (see Figure 2.6(e)), and Eckert et al. [39] (see Figure 2.6(f)) leverage a flexible spine, inspired predominately by felines, to improve the gait per-

formance of small quadruped robots in four ways: 1) by extending the stride length, 2) by accelerating the swing phase of the forelimbs, 3) by increase the ground contact force of the hindlimbs during kick-off, and 4) by contracting the robot body to position the hindlimbs further forward during touch-down. Therefore, the flexible spine is an additional mechanism for extending the effective stride length per gait cycle.

The designs by [37], [38] have a central revolute joint acting as the spine that connects the front and rear bodies of the quadruped robot. The spine in [38] is actuated by a direct drive, while the spine in [37] has an integrated torsion spring at the joint and is actuated by a pulley system that mimics the muscle actuation behavior of the latissimus dorsi and psoas major in felines. The spine design in [39] has three series-connected revolute joints enabling vertical flexion. The joints in [39] are connected by a single leaf spring instead of a torsion spring and actuated similarly to the design in [37]. The use of leaf or torsion spring elements at the joints is a common design trend for efficient energy storage and release of the spine during the gait cycle.

Concretely, the flexible spine in quadruped robots increases the jumping distance and gait speed performance. The authors in [37] conducted jumping experiments and observed that with the flexible spine enabled: 1) the ground impulse during a jump was 4.5 % greater, and 2) the robot was able to land with the hindlimbs further forward. The combination of both spine-based effects increased the jumping distance of the base robot by 15 %. Khoramshahi et al. [38] showed that the use of an actuated spine increased the gait speed from 0.62 m/s (rigid-spine) to 0.78 m/s (26 % increase). However, tests conducted by [39] showed that a quadruped robot with a multi-joint spine was 25 % slower than the single joint Bobcat [38]. Eckert et al. [39] attribute the difference in speed to difficulties in tuning the multi-joint spine. The authors in [39] concluded that the multi-joint spine underperformed the single joint spine but exhibited gait patterns closer to those found in the literature (especially with flight-phase footfall).

Researchers have also explored the effects of passive flexible spines for improving gait performance. Zhao et al. [40] designed a feline-inspired quadruped walker, named Renny, that has pairs of antagonistic pneumatic artificial muscles to actuate the legs and 1 DOF compliant spine. Due to the actuator and spine design, Zhao et al. [40] was able to conduct experiments with three spine morphologies: 1) a rigid spine (on DOF), 2) a passive spine (1 DOF, no active actuation), and 3) an active spine (1 DOF, actuation). The experimental results in [40] showed similar gait speed performance trends as in [37], with the active spine increasing the gait speed (1.18 m/s) by 40 % compared with the rigid spine. More importantly, Zhao et al. [40] showed that a passive spine, which stores potential energy during the gait, increased the gait speed by 17 % compared with the rigid spine. Therefore, even a passive spine improves the gait performance compared with a fully rigid spine.

The quadrupeds in [37]–[39] are controlled with a central pattern generator. The spine flexion is parametrized with an amplitude and phase offset in the central pattern generator. Both parameters need to be tuned for specific gait types to maximize the straight-line speed performance. Eckert et al. [39] outlined in their results that a poorly

tuned CPG resulted in zero or negative gait velocities. It can be concluded that, in general, CPG control is a common control strategy for small quadrupeds with flexible spines. Furthermore, to the best of our knowledge, no authors have yet proposed a complete model-based control architecture for quadruped robots with flexible spine control, particularly a spine controller that handles multiple spine functions.

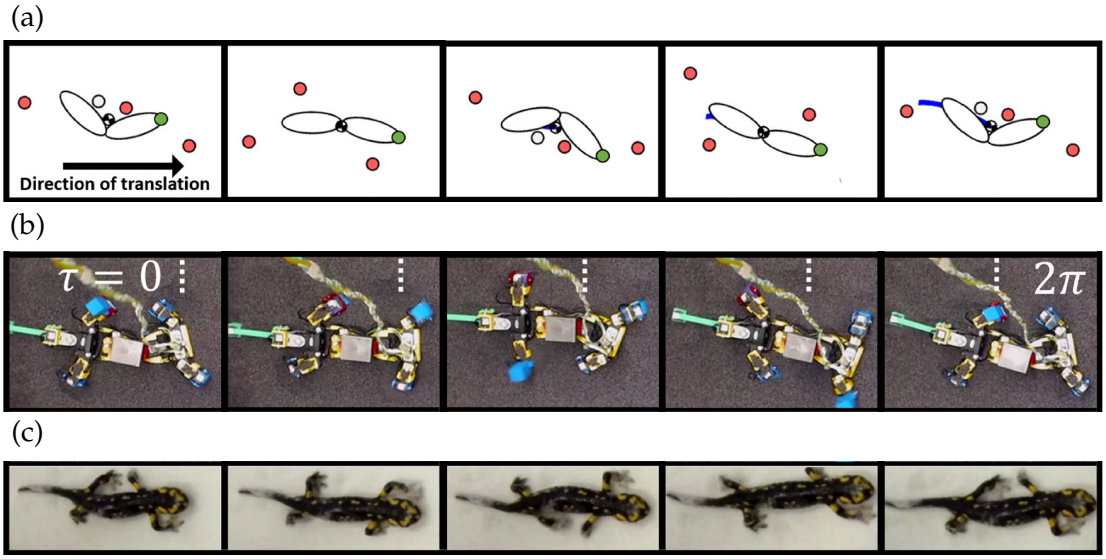


Figure 2.8.: Gecko robot developed by [41]. (a) Illustration of the gecko gait. The green dot denotes the head, the red dots denote feet in contact with the floor, and the white dots denote feet in the swing phase. The blue line depicts the trajectory of the center of mass. (b) Timestamp shots of the gecko robot in gait. (c) Comparative images of animal experiments. Figure has been adapted from [41].

Before we conclude the exploration into legged robotics, we want to cover one last robot example that uses lateral spine flexion for stride length extension and turning. The Gecko robot in [41], depicted in Figure 2.8, is inspired by the gait of reptilian quadrupeds (such as salamanders). The flexible spine consists of a single direct-drive joint enabling flexion in the frontal plane. Figure 2.8(b) depicts a gait of the gecko robot with spine flexion enabled, which extends the effective stride length for symmetric gaits by maximizing the distance between the hip and shoulder of the swing leg pair. Trivially, the distance maximization also minimizes the diagonal leg pair distance during the stance phase. Furthermore, [41] demonstrated the use of the spine to help in turning, with the turning radius r_{turn} of the robot computed as

$$r_{turn}(\theta) = \frac{l_s \cdot \sin(\frac{\pi-\theta}{2})}{\sin(\theta)}, \tag{2.1}$$

where l_s is the stride length and θ the stride rotation per gait cycle. To the best of our

knowledge, this is one of the few robotic systems combining two spine functions for locomotion.

2.3. Simulation Frameworks

The use of physics engines and simulators is synonymous with modern robotics research. Simulators provide an efficient and low-cost approach to testing new robotic design, control, and algorithmic concepts using a virtual robot (digital twin) in virtual environments before transitioning to real-world tests. Modern-day robotic simulators require strong modeling capabilities, API integration (with C++ and Python), and graphical interface options for end-users [19]. Erez et al. [42] note the importance of distinguishing between visually plausible simulators (often used in gaming engines and visual effects) and physically accurate simulators. The former trades accuracy for performance and stability, while the latter leverages efficient recursive algorithms in joint and velocity space to compute contact dynamics accurately [42].

Within the scope of this thesis, we focus on physically accurate robotic simulators engines relevant for mobile robotics. These simulators offer several benefits for robotic development [18], [19], [42], [43] summarized below:

- **Improving robot development:** Using a simulator avoids bottlenecks through purchase and sourcing of hardware. The simulator also eliminates the risk of damaging expensive and fragile components. Furthermore, parameters and test scenarios can be covered that would otherwise not be possible.

- **Accelerate interactive learning:** A simulator provides significant benefits in learning-based tasks because the number of learning iterations required is not practical for robot experiments. The virtual learning epochs can be executed within the simulation at rates exceeding 100 to 1000 times those of real-world tests. Furthermore, simulators offer the possibility to quickly reset virtual environments and build these scenes far more quickly.

- **Enabling collaborative research:** Simulation environments, particularly with open-source simulators, can be easily and quickly shared across multiple researchers and research teams to speed up development drastically.

Several different simulators have been developed to meet the growing demand for mobile robotics and reinforcement learning research. A list of the most relevant simulators is provided in Table 2.3. Gazebo and CoppeliaSim are very popular simulators in robotics because of their ROS integration, extensive sensor packages, and rigid-body capabilities. It should be noted that Gazebo supports multiple physics backends. PyBullet has similar strengths to Gazebo and CoppeliaSim and can handle soft-body contacts like MuJoCo and Chrono. The latter two are the only robotic simulators in the list capable of handling bio-inspired tendon-driven mechanisms, with MuJoCo providing the most

support in this area [42]. Therefore, MuJoCo and Project Chrono offer the greatest modeling freedom, including the modeling of bio-inspired mechanisms.

Two crucial metrics for evaluating simulators are the accuracy-to-performance ratio and ease of integration for reinforcement learning. A study conducted by [42] across four different test cases - grasping, humanoid walker, planar chains, and 27 capsule contact test - showed that MuJoCo was the fastest and most accurate simulator for constrained robotic systems (compared with PyBullet, PhysX, ODE, Havok). In contrast, all of the simulators in Table 2.3 provide tools and integration for reinforcement. Nevertheless, one of the most popular [19] reinforcement learning toolkits, OpenAI Gym, implements the environments used for baseline testing of reinforcement learning algorithms in MuJoCo. Therefore, MuJoCo is rated highly amongst reinforcement learning researchers [19], [42] for its simulation accuracy and existing integration with reinforcement learning toolkits.

Table 2.3.: Overview of modern robotic simulators used throughout research. This overview has been compiled and summarized from [19].

Simulator	RGBD & LiDAR	Force Sensor	Tendon Actuator	Soft-Body Contacts	Real Rendering	IK
CoppeliaSim	✓	✓	✗	✗	✗	✓
Gazebo	✓	✓	✗	✗	✗	✓
MuJoCo	✓	✓	✓	✓	✗	✗
PyBullet	✓	✓	✗	✓	✗	✓
Chrono	✓	✓	✓	✓	✓	✓

While the use of robotic simulators offers many benefits, [18], [19] highlight several critical challenges to consider when transferring simulation results to real-world applications. The first challenge is replicating real-world environments, robots, and physics without drastically increasing compute. Next, validating and debugging simulation models is not always straightforward and is best achieved by cross-validation with real-world tests [18]. For model-based simulations, the reality gap occurs due to numerical and model-based errors. The numerical errors can be solved by adjusting and tuning the simulation's integration time step and are typically limited by available hardware. Meanwhile, model errors rest with the designers and can also be adequately addressed by improving the model fidelity. In contrast, due to the black-box approach, learning-based algorithms suffer from simulation to reality performance deficits. Several solution strategies, such as environment varying and domain randomization, exist to reduce the reality gap, with modern simulators also offering integrated support for these strategies. However, an in-depth exploration into this area is beyond the scope of this thesis.

3. Problem Statement

In the previous chapter, we covered the foundations of rodent gait geometries and metrics critical for the implementation of Nermo's digital twin. We then covered four-legged gait locomotion, from foundations to the control methodologies. Lastly, we looked at the current research exploring the use of a spine in various aspects to influence the gait. The digital twin we are developing in this thesis is based on the bio-inspired robotic mouse Nermo. Nermo has several unique design features that separate it from other researched quadrupeds, which we cover in detail in Chapter 4. Based on the current research on spine-based locomotion and the unique design features of Nermo, we aim to develop a novel digital twin and spine-focused control architecture for Nermo. We then aim to use the digital twin and control architecture to explore five main research questions.

- To what extent can a multi-joint compliant spine improve the straight-line gait speed and stability in underactuated quadrupeds?
- To what extent can a multi-joint compliant spine improve the turning speed and minimum turning radius in underactuated quadrupeds?
- To what extent can a multi-joint compliant spine improve the static stability of underactuated quadrupeds?
- To what extent can a multi-joint compliant spine improve navigation speed and agility through a complex environment in underactuated quadrupeds?
- To what extent can the digital twin accurately predict the behavior of Nermo in the real world?

4. System Design

In this chapter, we provide an overview of Nermo before describing the methodology in Chapter 5. First, we introduce the general specifications of Nermo before covering the state-of-the-art skeleton and actuation system of Nermo. We end the chapter by outlining the details of Nermo's electrical system.

4.1. General Specifications

Nermo is a quadruped robot that mimics common rodents' anatomy, appearance, and motion. In this thesis, we are using version 4.1 of Nermo as shown in Figure 4.1. Nermo is a unique quadruped robot for several reasons: 1) Nermo has a highly complex compliant skeleton with a flexible ribcage that houses the electronics, 2) the fully compliant spine actuatable in the frontal and sagittal plane, 3) the compliant knee and ankle joints of the legs, and 4) the tendon driven actuation system. The primary physical characteristics of Nermo are listed in Table 4.1. Nermo is fully untethered, autonomous capable, and battery-powered, which is state-of-the-art in this form factor.

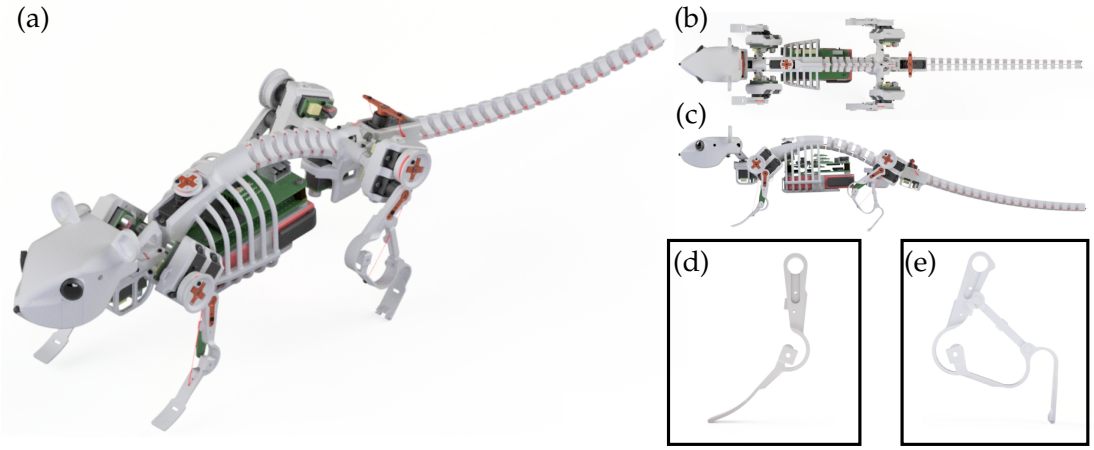


Figure 4.1.: Overview of the Nermo quadruped research platform. (a) Depicts Nermo in isometric view, (b) in top view, and (c) in side view. (d) Depicts the front compliant leg and (e) the rear compliant leg.

Table 4.1.: Overview of mechanical and electrical specifications of Nermo.

Specification	Detail
Length	405 mm
Length Scapula-Pelvis	117 mm
Height	91 mm
Width	90 mm
Weight	275 g
Actuated Degrees of Freedom	13
Battery	7.4V 1000mAh LiPo
Run Time	25 min.
Main Computer	Raspberry Pi Zero
Cost	approx. 1000 EUR

4.2. Mechanical Design

One unique element of Nermo is the skeleton that connects all the components, as shown in Figure 4.1. The skeleton is designed to be lightweight, compliant, and modular. The main load-bearing elements of the skeleton are the following:

- **Compliant Spine**: The spine consists of four compliant joints that enable vertical flexion and four compliant joints that enable horizontal flexion (depicted in Figure 4.2). The joints are arranged in alternating order. Additionally, the vertical compression of the spine is limited by the form of the vertical spine joints. The mass of the spine and main body skeleton is 30 g.
- **Compliant Front Leg**: The compliant front leg depicted in Figure 4.1(d) emulates a three-bar linkage. The knee joint is compliant and has a compartment for a magnetic encoder to measure the knee angle. The lower leg element has several attachments for the tendon to vary the torque applied on the compliant knee joint. Lastly, the lower leg element transitions directly into the foot, providing a pseudo-compliant joint for the ankle. The foot contains a mounting point for a foot-sole to house a ground contact sensor. The mass of each front leg is 2 g.
- **Compliant Rear Leg**: The compliant rear leg depicted in Figure 4.1(e) emulates a four-bar linkage. The knee, pre-ankle, and ankle joints are all compliant joints. The knee joint has a compartment for a magnetic encoder to measure the knee angle. The lower leg has three tendon attachment points to vary the torque on the compliant knee joint. The foot also contains a mounting point for a foot-sole to house a ground contact sensor like the front leg. The mass of each rear leg is 3 g.

In addition to the load-bearing elements, Nermo's skeleton has a fully compliant tail with 18 series-connected compliant joints (similar to the spine). The mass of the tail skeleton is 10 g.

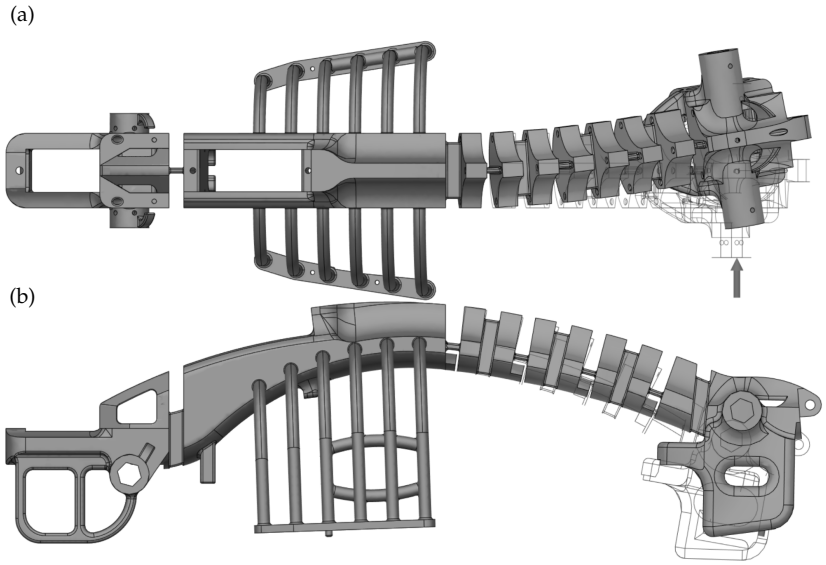


Figure 4.2.: Depiction of the lateral (a) and vertical (b) spine flexion. The wireframe shape depicts the original configuration before flexion.

4.3. Electrical Design

Figure 4.3 shows the electrical actuation system of Nermo. In total, 13 Power HD-DSM44 micro servos [44] (with custom controller boards) actuate the 13 degrees of freedom. Table 4.2 lists the distribution of the servos across the components of Nermo. The computing platform in Nermo is a Raspberry Pi Zero. The Raspberry Pi provides several advantages: 1) the ability to wirelessly communicate with Nermo, 2) run compute on Nermo, including CV applications, via Linux-based OS, and 3) modularity for future sensor and motion upgrades.

Table 4.2.: List of the degrees of freedom of each sub component in Nermo. Each degree of freedom is actuacted by one servo.

Component	DOF	Details
per Leg	2	hip rotation and knee flexion
Spine	2	lateral and lumbar flexion
Tail	1	lateral flexion
Head	2	pan and tilt

Additionally, we have the option to execute compute on a remote machine. To control the servos and address the sensors, we have a custom control board that extends the capabilities of the Raspberry Pi. The custom control board handles power delivery from the connected 7.4 V LiPo battery as the servos run at 6V and the Raspberry Pi at 5 V.

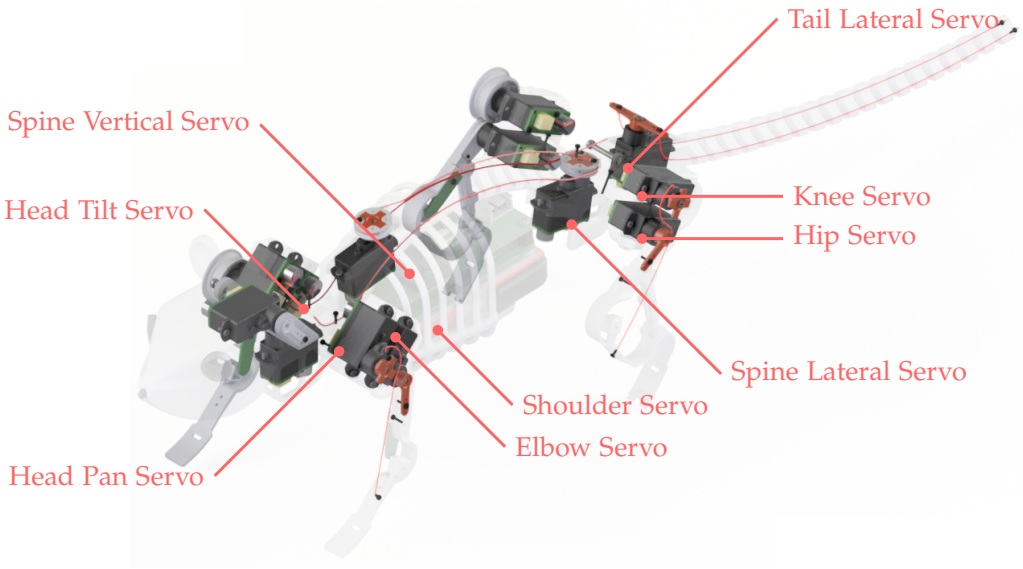


Figure 4.3.: Detail view of the servo and tendon system actuating Nermo. The tendons are fixed at the attachment points using a press-fit with M1 screws.

The servos directly drive the hip, shoulder, head pan, and head tilt, while a tendon pulley system drives spine lateral flexion, lumbar flexion, tail flexion, and knee flexion. The pulley setup for the front leg, rear leg, and spine flexion is shown in Figure 4.3. Of the seven tendon pulley systems, only the spine and tail lateral flexion is designed to emulate an antagonistic muscle pair. For all other tendon pulley systems, the actuation only enables compliant joint compression, while extension comes from the reactive spring force in the compliant joint. We use a 0.35 mm diameter fishing line rated at 12 kg for the tendons. Nermo also has several integrated sensors listed in Table 4.3. Within the scope of this thesis, we make use of the servo and leg sensors.

Table 4.3.: Overview of the sensors in Nermo.

Component	Sensory Capability
Servos	absolute position and current
Legs	absolute knee angle
Feet	ground pressure
Head	2x HD cameras

5. Methodology

In this chapter, we cover the process of developing the digital twin of Nermo. We begin by deriving the models for the compliant front legs, rear legs, and spine. These models form the backbone of the model-based controller. We then explore and derive several functional extensions of the compliant spine: 1) spine offset compensation, 2) spine-based gait extension, and 3) spine balance compensation. After the modeling section, we discuss the model-based control architecture. It should be noted that the term front leg refers to the forelimbs and the term rear leg refers to the hindlimbs. We use these terms interchangeably in the following sections.

5.1. Modeling of the Compliant Legs

This section defines the kinematic models of the compliant legs required for the model-based locomotion control. First, to simplify the modeling approach, we introduce a discretization model for the compliant joints. Second, we cover the open-loop kinematic model and define a set of requirements to formulate the closed-loop kinematic model. Lastly, we compute the inverse kinematic model of the legs and outline the consequences of the different closed-loop leg configurations on the leg control.

5.1.1. Discretization of Compliant Joints

Compliant joints are, in general, complex to model and simulate without substantial compute performance. For example, realistic compliant component behavior is simulated in isolated scenarios using finite element simulations that can, depending on the complexity, take several hours or days to compute. As this approach is not tractable for dynamic robot simulations, we introduce a model to approximate the compliant joints. First, the compliant joint is geometrically approximated as a discrete revolute joint positioned at the center of rotation of the compliant joint. Second, the material and geometric stiffness of the compliant joint is approximated as a virtual torsion spring-damper (with stiffness k_j and damping c_j) at the discrete joint. Figure 5.1(a) shows an example discretization of the compliant joint. The introduced discretized compliant joint model provides a method to approximate the behavior of the compliant joints at a fraction of the compute performance.

It should be noted that the discretization of the compliant joint is only an approximation. The approximation is more accurate in components with small angular deflections of the compliant joint than large angular deflections. Figure 5.1(b) demonstrates the endpoint error between the discretized leg and the compliant leg. The position error

occurs because the compliant joint rotates and compresses around the center of rotation. Therefore, we incur a propagated model error for greater angular deflections due to the additional compression behavior that is not considered in the discrete compliant joint model.

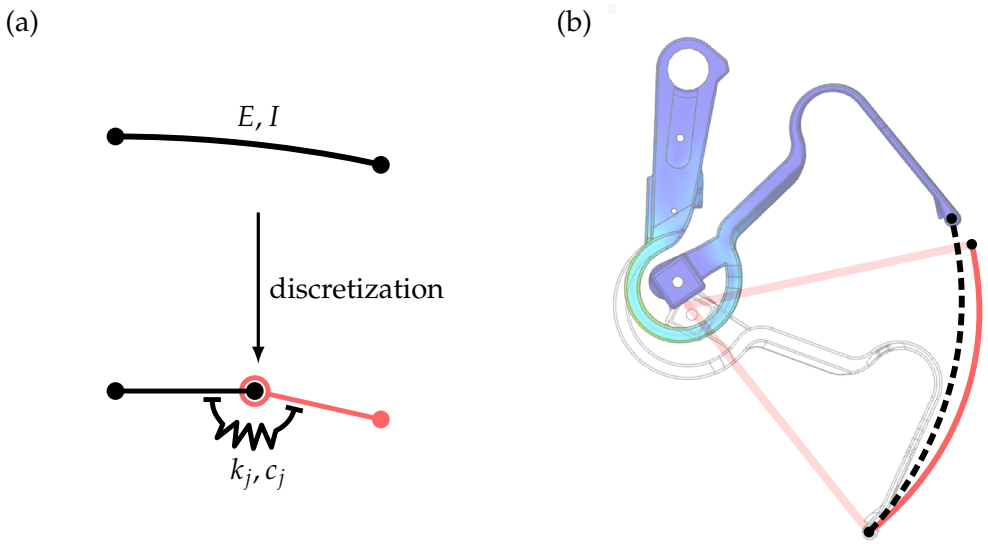


Figure 5.1.: (a) Schematic of the discretization of the compliant joint with stiffness E and moment of inertia I to the discrete revolute joint with virtual spring damper (k_j, c_j). (b) Comparison of foot-end deflection trajectory between compliant joint simulated with FEM (dashed line) and discretized compliant joint (solid line). For the same effective joint angle, the compliant joint exhibits additional non-radial compression and travel.

Lastly, we evaluate the joint stiffness and damping of the virtual spring-damper using finite-element analysis. The analysis process consists of three steps:

1. For each compliant component, we apply torque at the compliant joint.

2. For each case, we measure the angular deflection induced by the torque.

3. Using the relationship $T_j = k_j \theta_j$, where T_j is the applied torque at the compliant joint and θ_j the angular deflection, we solve for the virtual joint stiffness k_j.

The damping parameter c_j is chosen to achieve a stable yet responsive dynamic behavior at the joint. The full results of the finite-element analysis, including the stiffness values, can be found in Appendix A.1.1.

5.1.2. Overview of Compliant Front Leg Models

In this section, we derive the kinematic model of the front legs using the introduced discretized compliant joint model. First, we provide an overview of the open-loop kinematic chain of the leg without considering the tendon pulley system. Second, we introduce the tendon pulley system to explore various closed-loop kinematic configurations of the front legs. Last, we outline the implications of the tendon configuration on the control of the legs.

Front Leg Open-Loop Kinematic Model

The front leg open-loop kinematic model consists of a three-bar linkage with three discrete revolute joints as illustrated in Figure 5.2. To reiterate, the compliant knee and ankle joints have been replaced by the discrete revolute joints introduced with the discretized compliant joint model. The coordinate frame {O} is centered at the shoulder joint. The angles denoted with q_i refer to the servo actuated angles controlled by the controller. All other angles are denoted by θ_i.

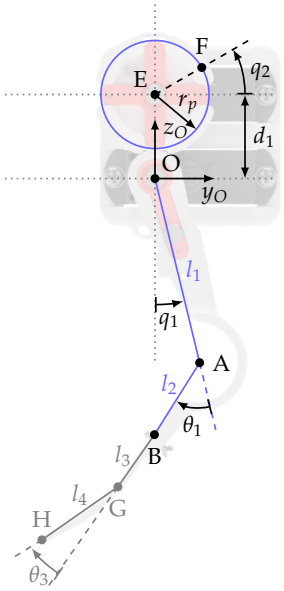


Figure 5.2.: Front leg open-loop kinematic diagram on top of front leg CAD model. The front leg consists of the leg with tendon fixation point (B) and tendon pulley at (E) of radius r_p. The tendon is fixed at point (F) on the pulley and actuated by the servo value q_2.

The parameters shown in Figure 5.2 are defined as follows: d_1 is the distance between shoulder and tendon pulley axis OE, r_p is the radius of the tendon pulley, l_1 is the length of the upper leg segment OA, l_2 is the length of lower the leg segment until the tendon fixation point AB, l_3 is the length from the tendon attachment point to the ankle BG, and l_4 is the length of the foot segment GH. The controllable servo angles are defined

as q_1 for the shoulder servo and q_2 for the tendon pulley servo. The initially unknown parameters are θ_1 as the knee angle between the upper and lower leg, and θ_3 as the ankle angle. The forward kinematics from the base frame {O}, centered at the shoulder axis, to the foot-end {H} are defined as

$$\begin{pmatrix} y_H \\ z_H \end{pmatrix} = \begin{pmatrix} l_1 \cdot \sin(q_1) - l_3 \cdot \sin(\theta_1 - q_1) - l_4 \cdot \sin(\theta_3 + \theta_1 - q_1) \\ -l_1 \cdot \cos(q_1) - l_3 \cdot \cos(\theta_1 - q_1) - l_4 \cdot \cos(\theta_3 + \theta_1 - q_1) \end{pmatrix}, \tag{5.1}$$

where y_H and z_H denote foot-end position (H) with respect to the base frame {0}. For all further considerations we assume the compliant joint of the ankle as sufficiently stiff and thus consider θ_3 as constant.

Overview of Front Leg Closed-Loop Configurations

The forward kinematic model of the open-loop front leg configuration does not define a physical connection between the leg and knee servo pulley at (E) actuated by q_2. As a result, the open-loop kinematic chain does not contain the actuation term q_2 from the pulley servo. Therefore, we introduce a closed-loop kinematic chain by connecting (B) to a new point (D) at the knee servo pulley with an inelastic massless tendon. The tendon pulley connection can be implemented in three design configurations. These three configurations are depicted in Figure 5.3.

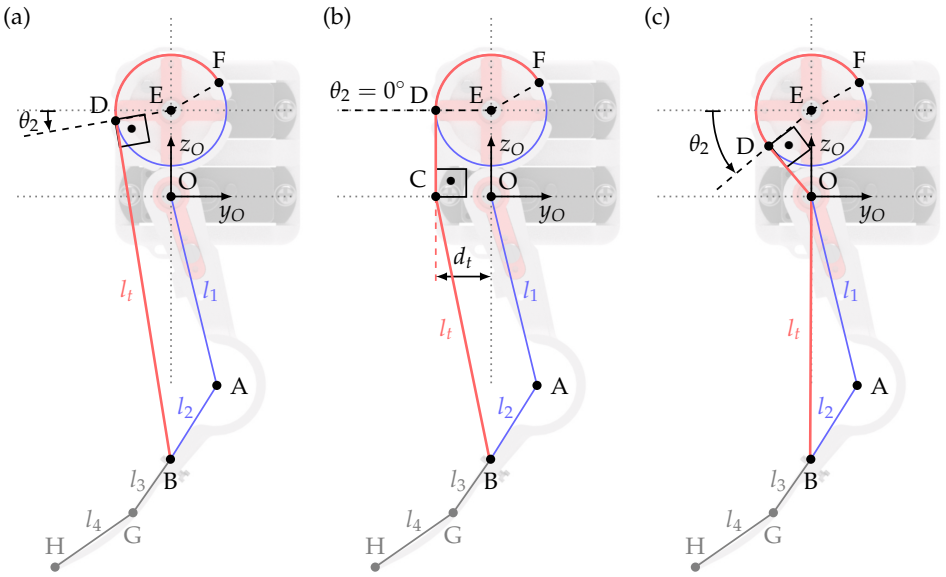


Figure 5.3.: The three closed-loop kinematic tendon configurations. (a) Type 1 is the most general configuration with greatest modeling complexity. (b) Type 2 constrained tendon configuration where $\theta_2 = 0°$. (c) Type 3 is a specific variant of the type 2 configuration with the tendon constrained at the shoulder such that $\theta_2 = const$. (lowest modeling complexity).

We define the three tendon configurations as follows:

1. **Type 1**: This is the most generalized case where the tendon is only constrained at the pulley and the leg attachment point (B). There is a direct connection between the leg knee tendon and the knee servo pulley, which leads to a significant influence of the tendon wrapping on the foot-end position (H).
2. **Type 2**: In this configuration, the tendon is constrained at the pulley (D), the leg attachment point (B), and constrained by a third routing point. In this setup, the routing point ensures that the tendon is always at a fixed tangential angle to the pulley and thus removes the effect of tendon wrapping on the foot-end position (H).
3. **Type 3**: This is a particular version of the type 2 configuration, where the third routing point coincides with the axis of the shoulder at (O).

To describe the closed-loop configurations depicted in Figure 5.3, we introduce two additional parameters: θ_2 is the offset angle defining the tangential wrapping of the tendon at the pulley, and l_t is the tendon length. The tendon length l_t is defined as the length between the tendon fixation point B and the first tendon pickup point; for the type 1 leg at D, the type 2 leg at C and the type 3 leg at O. From observation of Figure 5.3 the tendon length l_t is not constant and defined as

$$l_t(\theta_1(q_1, q_2), q_2) = l_{t0} + r_p(\theta_2 + q_2), \tag{5.2}$$

where l_{t0} is the initial tendon length, BD (type 1 leg) or BC (type 2 leg) or BO (type 3 leg), for the neutral configuration of $q_1 = 0$ and $q_2 = 0$.

Table 5.1.: Overview of important parameters and the difference in formulations resulting from the chosen tendon configurations.

Leg Type	θ_1	θ_2	l_t
1	$f(q_1, q_2)$	$f(q_1, q_2)$	$l_{t0} + r_p(\theta_2 + q_2)$
2	$f(q_1, q_2)$	0	$l_{t0} + r_p q_2$
3	$f(q_2)$	$\text{acos}(r_p/d_1)$	$l_{t0} + r_p q_2$

The different tendon arrangements influence the dependencies of θ_1, θ_2, and l_t with respect to the servo control parameters q_1 and q_2. We observe three major differences between the tendon configurations: 1) the knee angle θ_1 only depends on the servo angle q_2 for the type 3 configuration, 2) θ_2 is constant for the type 2 and 3 configurations, and 3) $l_t = f(q_2)$ for the type 2 and type 3 configurations. These three differences are summarized by a comparison of the modeling simplifications in Table 5.1. Particularly the ability to control the knee angle θ_1 exclusively with the servo angle q_2 makes the type 3 tendon arrangement superior for control. Additional qualitative benefits and drawbacks of each tendon arrangement, regarding modeling, control, and leg design, are outlined in Table 5.2.

Table 5.2.: Qualitative summary and comparison of the three tendon configurations.

Summary	Benefits	Drawbacks
Type 1: Most general tendon configuration model. Leg position depends on tendon wrapping	Considers tendon wrapping. No tendon routing constraints.	Complex modeling. Implicit knee joint solution. Non-linear.
Type 2: Fixed tendon routing configuration. Leg position is no longer dependent on tendon wrapping.	Simplified modeling complexity. Explicit knee joint angle solution.	Higher design effort for tendon routing. Reduced lever arm on the knee.
Type 3: Special case of type 2. Routing of the tendon through the hip/shoulder axis.	Explicit knee joint angle solution. Knee joint independent of servo q_1. Options for compact tendon routing.	Increased design effort for tendon routing. Reduced lever arm on the knee.

5.1.3. Front Leg Closed-Loop Kinematic Models

The newly defined closed-loop leg configurations provide the necessary relationships for deriving the forward and inverse kinematic models with respect to the controllable servo parameters q_1 and q_2. Within the scope of this thesis, we focus on the type 1 and type 3 configuration for two reasons. First, the type 1 configuration is the most general description of the closed-loop configuration. Second, the type 3 configuration provides superior control characteristics and most closely matches the tendon pulley design of Nermo (see Appendix A.1.2). We introduce the following three assumptions to simplify the derivation of the kinematic models: 1) the tendon is inelastic and massless, 2) the knee joint (A) is sufficiently stiff such that the tendon is always under tension, and 3) the tendon attaches to the pulley tangentially at (D). In combination with the introduced closed-loop configurations, these assumptions provide the modeling foundations for solving the forward and inverse kinematic models of the front legs.

Type 1 Leg: Forward Kinematic Model

In the type 1 forward kinematics, expressed by the open-loop equation 5.1, the parameter $\theta_1 = f(q_1, q_2)$ is unknown. We introduce the tendon spiral abstraction to derive a solution for θ_1 in the type 1 configuration. Conceptually, the wrapping of the tendon around the servo pulley can be abstracted as a spiral, which is depicted in Figure 5.4 with the spiral radius r_s and spiral sweep angle θ_s. For example, as the tendon wraps in anti-clockwise direction ($\theta_2 < 0$ and $q_2 = 0$), the spiral radius r_s decreases whilst

the spiral sweep angle θ_s increases. Given this property, the interaction between the tendon and leg can be modeled as the intersection of a spiral wrapped around the spine pulley (E) and tendon fixation point (B). The tendon fixation point is modeled as a circle with radius l_2 around the knee joint (A). Therefore, we derive a solution for the tendon wrapping angle $\theta_2 = f(q_1, q_2)$ using the intersection problem to find a solution for θ_1 using the introduced tendon spiral model.

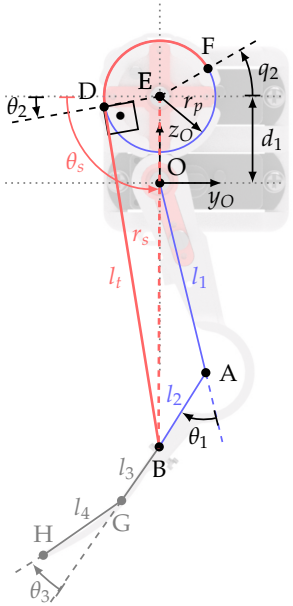


Figure 5.4.: Kinematic diagram of the closed-loop front leg with the introduced spiral of radius r_s and sweep angle θ_s which models the wrapping of the tendon around the knee pulley at (E).

The solution for θ_2 is dependent on the intersection of the tendon spiral and tendon fixation circle. To formally define the intersection problem between the spiral and the fixation point circle, we introduce the equation of the spiral, caused by the tendon wrapping around the servo pulley, in polar form

$$r_s = \sqrt{l_t^2 + r_p^2}, \tag{5.3}$$

$$\theta_s = \operatorname{atan}(l_t / r_p) - \theta_2, \tag{5.4}$$

and in parametric form

$$y_s = -r_s \cdot \cos(\theta_s), \tag{5.5}$$

$$z_s = -r_s \cdot \sin(\theta_s) + d_1, \tag{5.6}$$

where y_s and z_s are the coordinates of any point on the circumference of the spiral. The equation of the circle generated by the range of motion of the leg segment AB is defined as

$$l_2^2 = (y_{cB} - l_1 \cdot \sin(q_1))^2 + (z_{cB} + l_1 \cdot \cos(q_1))^2, \tag{5.7}$$

where y_{cB} and z_{cB} represent the coordinates of any point on the circumference of the circle.

There are three solution cases for the proposed intersection problem: 1) intersection at two points, 2) intersection at one point, and 3) intersection at no point. To simplify the problem, we assume that at least one intersection point exists between the spiral and the circle formed by the leg segment AB within the actuation range of the servo motors. The introduced assumption leads to the following two equalities: 1) $y_s = y_{cB}$ and 2) $z_s = z_{cB}$. By observing the parametric and polar form of the tendon spiral, we deduce that $y_s = f(\theta_2, q_2)$ and $z_s = f(\theta_2, q_2)$. Therefore, by applying the intersection equalities, we can find a solution for θ_2 and θ_1.

We begin by substituting the intersection condition into (5.7) to yield

$$l_2^2 = (y_s - l_1 \cdot \sin(q_1))^2 + (z_s + l_1 \cdot \cos(q_1))^2. \tag{5.8}$$

Next, substituting the values of r_s, θ_s, y_s, and z_s from (5.3), (5.4), (5.5), and (5.6) into (5.8) yields the implicit solution for θ_2

$$l_2{}^2 = (\sigma_1 \cdot \cos(\sigma_2) + l_1 \cdot \sin(q_1))^2 + (d_1 + \sigma_1 \cdot \sin(\sigma_2) + l_1 \cdot \cos(q_1))^2, \tag{5.9}$$

where

$$\sigma_1 = \sqrt{\left(\mathrm{l}_{t0} + q_2\,\mathrm{r_p} + \mathrm{r_p}\,\theta_2\right)^2 + \mathrm{r_p}^2},$$
$$\sigma_2 = \theta_2 - \operatorname{atan}\left(\frac{\mathrm{l}_{t0} + q_2\,\mathrm{r_p} + \mathrm{r_p}\,\theta_2}{\mathrm{r_p}}\right).$$

Equation 5.1.3 contains the implicit solution for θ_2 dependent on q_1 and q_2, which are known parameters. Therefore, using a standard root finding algorithm, we can compute a solution set for θ_2, denoted as θ_2^*.

In the last step, the solution $\theta_2 = \theta_2^*$ is used to compute a solution for θ_1, which is necessary for computing the forward kinematics. To this, we first derive the z-axis component of the tendon fixation point (B) from Figure 5.4 as

$$z_{cB} = -l_1 \cdot cos(q_1) - l_2 \cdot cos(\theta_1 - q_1), \tag{5.10}$$

where $z_{cB} = z_s$ for the intersection case. Applying the intersection equality ($z_{cB} = z_s$) and substituting z_s from (5.6) into (5.10) we obtain the explicit solution for θ_1

$$\theta_1 = \operatorname{acos}\left(\frac{r_s \cdot \sin(\theta_s) - d_1 - l_1 \cdot \cos(q1)}{l_2}\right) + q1, \tag{5.11}$$

where

$$r_s = \sqrt{l_t^2 + r_p^2},$$
$$\theta_s = \operatorname{atan}(l_t / r_p) - \theta_2^*.$$

The explicit solution for θ_1 from (5.11) confirms the original observation that $\theta_1 = f(q_1, q_2)$. The dependence of the knee angle on the tendon wrapping is also visualized in Figure 5.5. The tendon wrapping influence is most prominent in scenarios where $q_1 \neq 0$ and $q_2 = 0$. Figure 5.5 also verifies that the spiral model accurately models the interaction between the tendon and foot-end position for the type 1 tendon configuration.

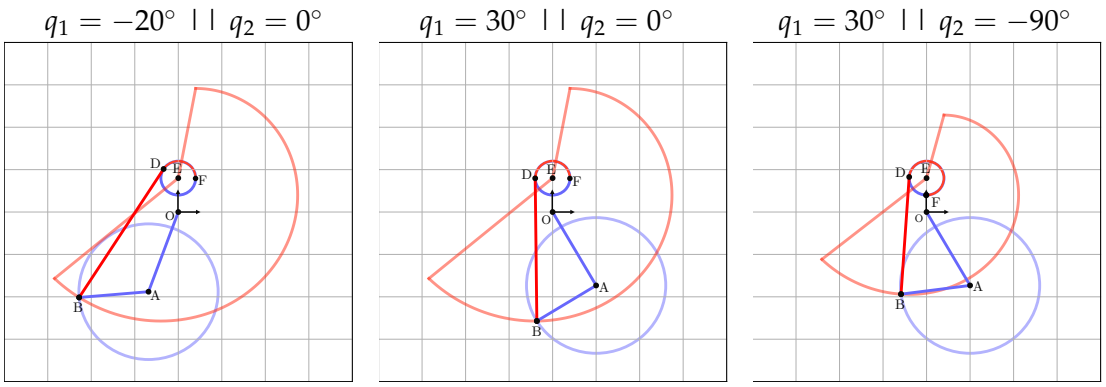


Figure 5.5.: Front leg type 1 kinematic model solutions for various servo control configurations on the tendon fixation point (B).

Type 1 Leg: Inverse Kinematic Model

The forward kinematics describe the foot-end position for a given set of servo control values. However, in the controller, we want to find a set of servo control values $(q_1, q_2)^T$ for a desired foot-end position $\boldsymbol{p_H}$. This is achieved with an inverse kinematic model. We derive the inverse kinematic model in the velocity space using the Jacobian of the forward kinematics and subsequently compute the servo control values q_1 and q_2 through numerical integration.

We begin the derivation of the inverse kinematic model by differentiating the forward kinematics in (5.1) to yield the velocity space kinematics

$$\dot{\boldsymbol{p}}_{\boldsymbol{H}} = \boldsymbol{J} \cdot \dot{\boldsymbol{q}}, \tag{5.12}$$

where $\boldsymbol{J}$ is the Jacobian derived from the forward kinematics, $\dot{\boldsymbol{p}}_{\boldsymbol{H}} = (\dot{y_H}, \dot{z_H})^T$, and $\dot{\boldsymbol{q}} = (\dot{q_1}, \dot{\theta_1})^T$. It should be noted that (5.12) is in the form $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$ and is solved with a standard linear system solver to compute the servo control velocities $\dot{\boldsymbol{q}}$. However, the solution for $\dot{q_2}$ and q_2 is implicitly contained in $\theta_1 = f(q_1, q_2)$. Therefore, we introduce the variable $\dot{\theta_1}$ as part of a two-step process to compute the numerical solutions for $\dot{q_1}$ and $\dot{q_2}$.

The expression for $\dot{\theta_1}$ is derived from the forward kinematics solution of θ_1 in (5.11). The servo control values q_1 and q_2 are implicitly contained in $\theta_1 = f(q_1, q_2, \theta_2)$. Therefore, we apply an implicit differentiation step to (5.11) such that

$$\dot{\theta}_1 = \frac{d\theta_1}{dt} = \sum_i \frac{\partial \theta_1}{\partial q_i} \frac{dq_i}{dt} + \frac{\partial \theta_1}{\partial t}, \tag{5.13}$$

$$\frac{d\theta_1}{dt} = \frac{\partial \theta_1}{\partial q_1} \frac{dq_1}{dt} + \frac{\partial \theta_1}{\partial q_2} \frac{dq_2}{dt} + \frac{\partial \theta_1}{\partial \theta_2} \frac{d\theta_2}{dt} + \frac{\partial \theta_1}{\partial t}. \tag{5.14}$$

However, $\theta_2 = f(q_1(t), q_2(t))$ and requires an additional implicit differentiation step to solve for $\dot{\theta}_2$ such that

$$\frac{d\theta_2}{dt} = \frac{\partial\theta_2}{\partial q_1}\frac{dq_1}{dt} + \frac{\partial\theta_2}{\partial q_2}\frac{dq_2}{dt} + \frac{\partial\theta_2}{\partial t}. \tag{5.15}$$

Substituting the solution from (5.15) into (5.14) yields a solution for $\dot{\theta}_1 = f(\dot{q}_1, \dot{q}_2, q_1, q_2)$. The solution for $\dot{\theta}_1$ is substituted into (5.12) to solve the linear system for $\dot{q}_1$ and $\dot{q}_2$. Lastly, we compute the solutions for the servo control values q_1 and q_2 by means of numerical integration of $\dot{q}_1$ and $\dot{q}_2$. The derived solutions show that the complexity of the type 1 tendon configuration leads to non-trivial, implicit solutions for both the forward and inverse kinematics.

Type 3 Leg: Forward Kinematic Model

In contrast to the type 1 tendon configuration, the type 3 tendon configuration reduces the complexity of the forward and inverse kinematic models. The foot-end position $\boldsymbol{p_H}$ of the type 3 leg is the same as defined for the type 1 leg (see (5.1)). However, unlike the type 1 leg, the solution to θ_1 is trivially derived from the cosine rule of the triangle enclosing ABC (refer to Figure 5.3).

$$\left(l_{t0} + q_2\, r_p\right)^2 = {l_1}^2 + 2 l_1 l_2 \cdot \cos\left(\theta_1\right) + {l_2}^2 \tag{5.16}$$

Rearranging (5.16) for θ_1 leads to the explicit solution

$$\theta_1 = \begin{pmatrix} \pi - \operatorname{acos}\left(\frac{\frac{{l_1}^2}{2} - \frac{\left(l_{t0}+q_2\, r_p\right)^2}{2} + \frac{{l_2}^2}{2}}{l_1\, l_2}\right) \\ \pi + \operatorname{acos}\left(\frac{\frac{{l_1}^2}{2} - \frac{\left(l_{t0}+q_2\, r_p\right)^2}{2} + \frac{{l_2}^2}{2}}{l_1\, l_2}\right) \end{pmatrix}. \tag{5.17}$$

Given that $\operatorname{acos}(x) > 0$ and $0 \leq \theta_1 < \pi$ we deduce the solution for θ_1 as

$$\theta_1 = \pi - \operatorname{acos}\left(\frac{\frac{{l_1}^2}{2} - \frac{\left(l_{t0}+q_2\, r_p\right)^2}{2} + \frac{{l_2}^2}{2}}{l_1\, l_2}\right). \tag{5.18}$$

With the solution of θ_1 in (5.18), we show analytically that the knee angle $\theta_1 = f(q_2)$ for the type 3 leg. This property is superior for leg control, because the knee angle is independent of the shoulder servo q_1. Substituting the solution for θ_1 into (5.1) yields the solution for the forward kinematics in terms of q_1 and q_2.

Type 3 Leg: Inverse Kinematic Model

The type 3 inverse kinematics are also derived in the velocity space using (5.12). However, for the type 3 leg we can directly compute the Jacobian with respect to (q_1, q_2) using the explicit solution for $\theta_1 = f(q_2)$. Therefore, the type 3 configuration avoids the

tedious two-step process of implicit differentiation for computing the servo control values compared with the type 1 configuration.

The Jacobian $\boldsymbol{J}$ of type 3 tendon configuration contains q_1, θ_1, and $\frac{\delta\theta_1}{\delta q_2}$, where

$$\frac{\partial\theta_1}{\partial q_2} = -\frac{r_p\left(\mathrm{l}_{t0} + q_2\, r_p\right)}{l_1\, l_2 \sqrt{1 - \frac{\left(\frac{{l_1}^2}{2} - \frac{\left(\mathrm{l}_{t0} + q_2\, r_p\right)^2}{2} + \frac{{l_2}^2}{2}\right)^2}{{l_1}^2\, {l_2}^2}}} \tag{5.19}$$

We substitute θ_1 with the solution from (5.18), whilst $\frac{\delta\theta_1}{\delta q_2}$ is substituted by (5.19). The result is a system of linear equations of the form $\boldsymbol{A}\cdot\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{x} = \dot{\boldsymbol{q}}$, $\boldsymbol{A} = \boldsymbol{J}$, $\boldsymbol{b} = \dot{\boldsymbol{x}}_{\boldsymbol{H}}$, which can be trivially solved. Lastly, with the solutions for $\dot{q}_1$ and $\dot{q}_2$, we compute the servo control values q_1 and q_2 using standard numerical integration methods. To conclude, the kinematic advantages of the type 3 tendon configuration are the lack of implicit solutions in the forward and inverse kinematics, and the condition that the knee angle θ_1 is controlled exclusively by the servo q_2.

5.1.4. Overview of Compliant Rear Leg Models

In the previous section, we derived the front leg kinematic model and outlined the influence of the closed-loop tendon configuration on the complexity of the kinematic models and leg control. To complete the model-based gait controller, we must repeat the modeling process for the rear legs. Conveniently, the rear legs are modeled using the same methods and considerations as for the front legs. Therefore, in the interest of brevity, we only cover high-level details specific to the modeling of the rear leg in this section.

Open-Loop Rear Leg Model

The rear leg is modeled as a three-bar linkage (see Figure 5.6(a)) and not as the four-bar linkage leg shown in Figure 4.1(e). The three-bar linkage is easier to model and has several mechanical design benefits, which are discussed in Appendix A.2.1. In the kinematic model of the rear leg, the compliant joints at (A) and (G) are replaced with virtual revolute joints using the introduced discretized compliant joint model. The parameter definitions of the kinematic diagram in Figure 5.6(a) are identical to the parameters for the front leg (albeit numerical different). However, the rear leg open-loop forward kinematic chain is slightly different because of directional differences in the definition of θ_1, q_1, and q_2 in the kinematic diagram.

$$\begin{pmatrix} y_H \\ z_H \end{pmatrix} = \begin{pmatrix} l_1 \cdot \sin(q_1) + l_3 \cdot \sin(\theta_1 + q_1) - l_4 \cdot \sin(\theta_3 - \theta_1 - q1) \\ -l_1 \cdot \cos(q_1) - l_3 \cdot \cos(\theta_1 + q_1) - l_4 \cdot \cos(\theta_3 - \theta_1 - q1) \end{pmatrix} \tag{5.20}$$

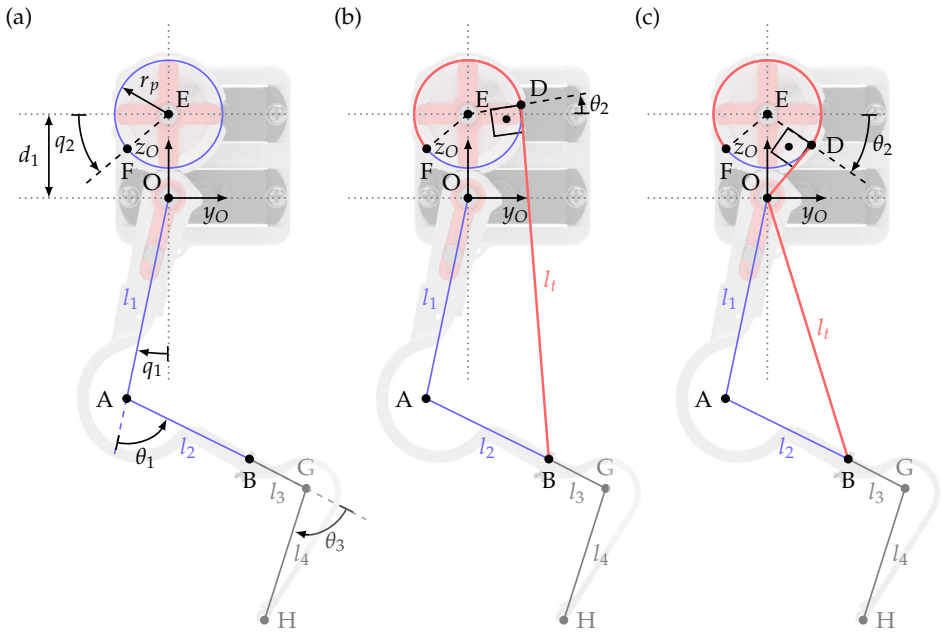


Figure 5.6.: (a) Front leg open-loop kinematic diagram on top of front leg CAD model. The front leg consists of the leg with tendon fixation point (B) and tendon pulley at (E) of radius r_p. The tendon is fixed at point (F) on the pulley and actuated by the servo value q_2. (b) Kinematic diagram of the type 1 tendon configuration. (c) Kinematic diagram of the type 3 tendon configuration. In the type 3 configuration the tendon is routed through the hip-axis at (O) before coming into contact with the pulley at (D).

Overview of Rear Leg Closed-Loop Models

We explained in the previous sections that a closed-loop leg model is required to express the foot-end position in terms of the servo control values q_1 and q_2. Figure 5.6(b) and Figure 5.6(c) depicts the two tendon configurations for the rear legs. The properties of the rear leg tendon configurations are identical to the front leg properties outlined in Table 5.1 and Table 5.2. Therefore, we limit the exploration to the type 1 (general case) and type 3 (superior control and similar to Nermo leg; see Appendix A.1.2) tendon configurations of the rear leg.

One additional advantage of the type 3 tendon configuration is the ability to route the tendon closer to the knee joint before attaching the tendon to point (B) (see Figure 5.7(a)). Routing the tendon closer to the knee has two advantages: 1) the tendon pathway more closely resembles the tendon routing in biological rodents, and 2) the tendon is shielded against possible interference with other components. However, routing the tendon closer to the knee joint reduces the available lever arm and increases the leg design complexity. The adjusted type 3 tendon configuration is also applicable to the front legs.

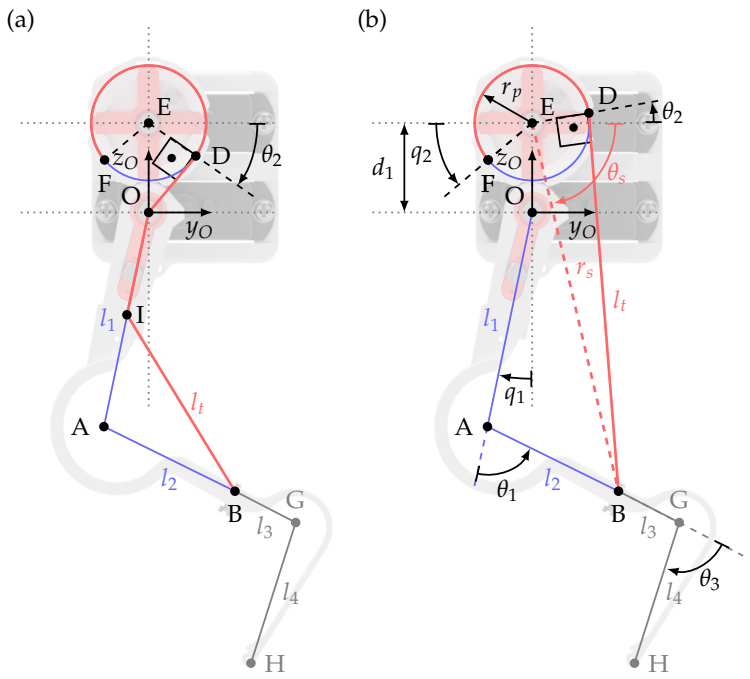


Figure 5.7.: (a) Adjusted type 3 configuration where the tendon is routed further down the upper leg to point (I) before attaching to (B). (b) Kinematic diagram of the type 1 rear leg with spiral of radius r_s and sweep angle θ_s.

5.1.5. Rear Leg Kinematic Model

The kinematic models for the type 1 and type 3 rear legs are derived identically to the front leg models. Therefore, we will not repeat the derivations in this section. For reference, the kinematic diagram, with spiral abstraction, used to solve the type 1 tendon configuration is illustrated in Figure 5.7(b). In Figure 5.8 we show the capabilities of the derived rear leg model to compute the tendon behavior accurately. Once again, we observe the significant influence of the tendon wrapping on the foot-end position (H), in particular for scenarios where $q_2 = const.$ and $q_1 \neq 0$.

5.1.6. Legged Locomotion

The derived leg models enable the legged locomotion of the rodent robot. Legged locomotion occurs in rhythmic cycles known as gaits. Within the scope of this thesis, we implement three gait types: lateral sequence walk (lat), walking trot (trt1), and ideal trot (trt) (refer to Figure 2.1 for the Hildebrand gait diagrams). One important gait metric is the gait speed s_m, which is computed as

$$s_m = f_g \cdot l_{es}, \tag{5.21}$$

$$l_{es} = g_f \cdot l_s, \tag{5.22}$$

where f_g is the gait frequency, l_{es} is the effective stride length, g_f is a gait specific scalar parameter, and l_s is the total stride length per gait cycle. The g_f parameter depends on the duty factor and phase offset of a gait cycle (refer to Appendix A.1.3). The ideal gait speed s_m assumes no slip of the stance limbs.

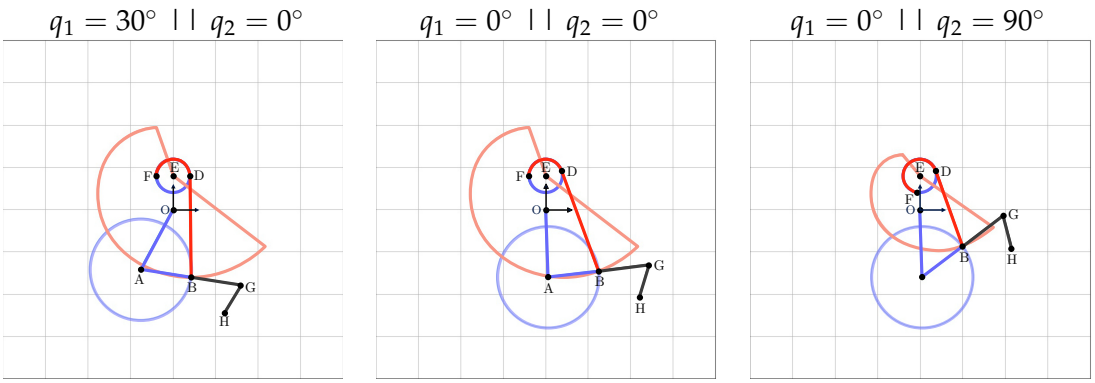


Figure 5.8.: Rear leg type 1 kinematic model solutions for various servo configurations.

5.2. Modeling of the Compliant Spine

This section derives the compliant spine models necessary for the model-based control in the spine controller. We begin by modeling the basic kinematics of the spine. Building on the kinematic model, we explore and model three extended spine functions: 1) spine-based gait extension, 2) spine-based offset compensation, and 3) spine-based balance compensation. Although the spine can be actuated in both vertical and lateral directions simultaneously, we only focus on the modeling and control of the lateral spine flexion within the scope of this thesis. Lastly, it should be noted that the methods outlined in this section can also be adapted to model the tail of Nermo.

5.2.1. Kinematic Model of the Spine

The compliant spine is modeled as a series connection of discretized compliant joints. Figure 5.9 visualizes the CAD model of the spine superimposed by the kinematic model wireframe. We assume the front torso of Nermo is at the base frame {M}. The spine is modeled as an eight-axis manipulator with the rear torso of Nermo - namely the hip, rear legs, and tail - modeled as the tool-end of the manipulator denoted by the frame {10} (see Figure 5.9). It should be noted that the discretized compliant joint model is especially effective for the spine because the angular displacement of each spine joint is below 0.2 rad.

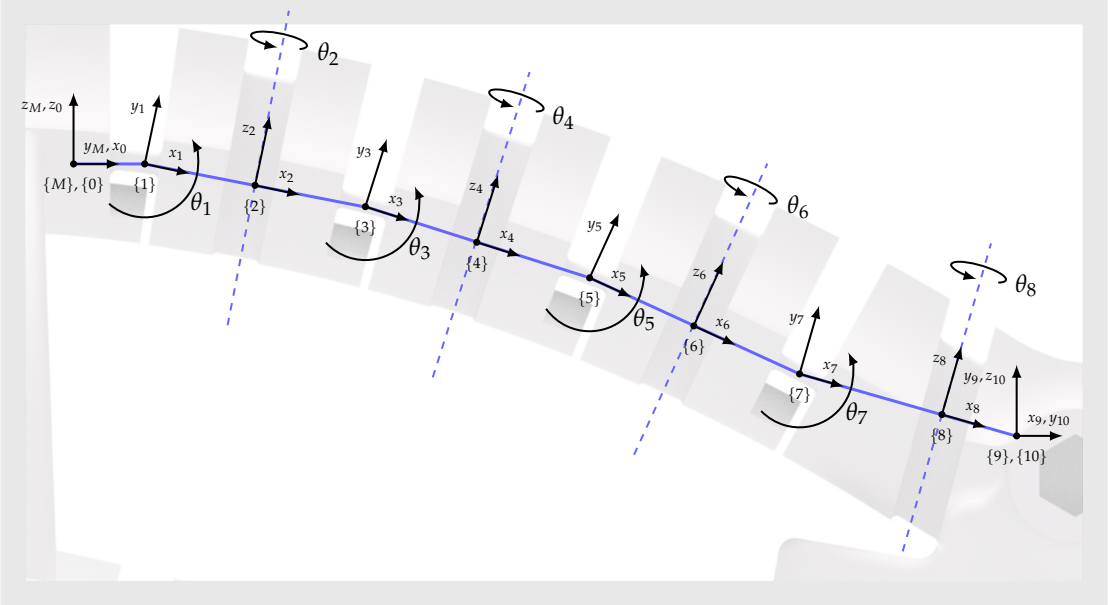


Figure 5.9.: Kinematic diagram of the spine according to the DH convention with eight revolute joints denoted by θ_i. Frame {M} denotes the base frame attached to the main torso of Nermo and frame {10} denotes the end frame attached to the hip of Nermo.

Initially, the spine model depicted in Figure 5.9 includes both the lumbar and lateral joints. The four lateral flexion joints are denoted by θ_2, θ_4, θ_6, and θ_8. The lumbar joints are depicted in Figure 5.9 at frames {1}, {3}, {5}, and {7} with their respective joint angles θ_1, θ_3, θ_5, and θ_7. The lumbar joint values are set constant because we omit lumbar (vertical) flexion in this thesis. However, future iterations of the model can easily be extended to include lumbar flexion by not setting the joint values constant.

Similar to the kinematic models of the legs, we are interested in the forward kinematics of the spine to predict the hip displacement caused by spine flexion. We use the modified Denavit-Hartenberg (DH) convention to compute the forward kinematics of the spine because the DH convention provides a convenient and well-understood modeling framework for multi-joint manipulators. Figure 5.9 illustrates the kinematic wireframe of the spine and the relevant modified DH parameters. The values of the modified DH parameters of the spine model are listed in Table 5.3. Using the modified DH convention, the homogeneous transformation matrix between a previous vertebrate coordinate frame $i-1$ and the current vertebrate frame i is deduced from (5.23).

$$
{}^{i-1}\boldsymbol{T}_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cdot\cos(\alpha_{i-1}) & \cos(\theta_i)\cdot\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -d_i\cdot\sin(\alpha_{i-1}) \\ \sin(\theta_i)\cdot\sin(\alpha_{i-1}) & \cos(\theta_i)\cdot\sin(\alpha_{i-1}) & -\cos(\alpha_{i-1}) & d_i\cdot\cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.23}
$$

Table 5.3.: Modified DH parameters of the spine model shown in Figure 5.9. The actuated joints are indicated with the parameters θ_i.

i (Frame)	$\alpha_{i-1}[^\circ]$	a_{i-1} [m]	θ_i [°]	d_i [m]
0	0	0.0000	90.0	0
1	90	0.0016	θ_1 =-10.7	0
2	-90	0.0070	θ_2	0
3	90	0.0070	θ_3 =-7.0	0
4	-90	0.0070	θ_4	0
5	90	0.0070	θ_5 =-6.9	0
6	-90	0.0070	θ_6	0
7	90	0.0070	θ_7 =9.7	0
8	-90	0.0092	θ_8	0
9	90	0.0014	15.0	0
10	-90	0.0000	-90.0	0

Using the DH convention and DH parameters from Table 5.3, the complete transformation matrix from the base frame {M} of the spine to the end frame {10} located at the hip is

$$^{M}\boldsymbol{T}_{10} =^{M} \boldsymbol{T}_0 \cdot \prod_{\boldsymbol{i=1}}^{\boldsymbol{10}} {}^{i-1}\boldsymbol{T}_i, \tag{5.24}$$

where $^{M}\boldsymbol{T}_{10}$ denotes the homogeneous transformation matrix from the base frame {M} to the end frame {10}. We highlight that $^{M}\boldsymbol{T}_{10} = \boldsymbol{f}(\theta_2, \theta_4, \theta_6, \theta_8)$. With the transformation matrix $^{M}\boldsymbol{T}_{10}$, any point $^{10}\boldsymbol{p_e}$ in the coordinate frame of the end-point (hip) can be described as a point in the base frame $^{M}\boldsymbol{p}_e$ with

$$^{M}\boldsymbol{p}_e =^{M} \boldsymbol{T}_{10} \cdot {}^{10}\boldsymbol{p}_e, \tag{5.25}$$

where $^{M}\boldsymbol{p}_e = \boldsymbol{f}(\theta_2, \theta_4, \theta_6, \theta_8)$.

To simplify the kinematic model, we introduce the assumption that the angular displacement of each spine joint is identical, $\theta_i = \theta_j$ for $i \neq j$, such that the angular displacement of each joint is expressed by a single variable $\theta_i = \theta_v$. The assumption that each joint exhibits equal angular displacement holds as each lateral spine joint in Nermo is mechanically equal and connected in series. Therefore, the introduced assumption reduces the forward kinematics expressed by the homogeneous transformation matrix to a single unknown variable $^{M}\boldsymbol{T}_{10} = \boldsymbol{f}(\theta_v)$.

Thus far, we expressed the kinematics of the spine with respect to the spine vertebrate angle θ_v. However, the spine is actuated by the tendon system with the spine pulley servo q_{sp}. In order to control the spine, we need to express $\theta_v = f(q_s p)$. The rotation of the spine servo exerts a torque that is equally distributed to each of the four joints based on the equal joint assumption. Therefore, the actuation of the spine servo leads to a

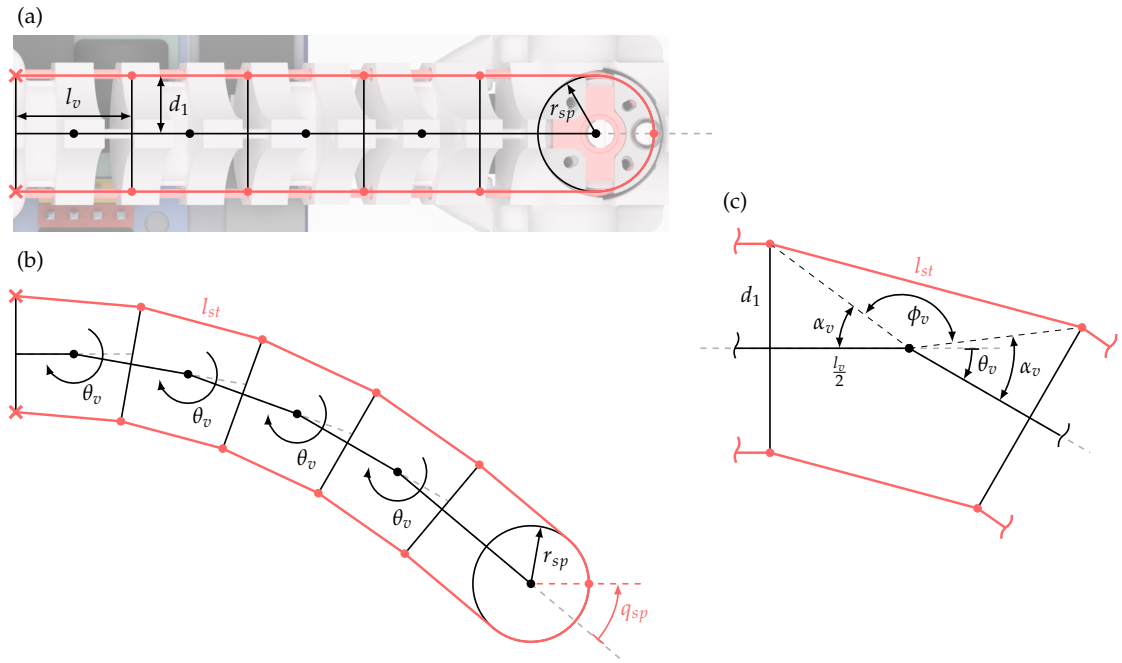


Figure 5.10.: (a) Schematic of the spine in neutral configuration ($q_{sp} = 0$) with the CAD of the spine underneath. (b) Schematic of the spine with the servo actuated ($q_{sp} \neq 0$). (c) Kinematic diagram of a single spine element.

relative change in the spine tendon length l_{st} on either side of the vertebrate dependent on q_{sp}.

The relationship between θ_v and q_{sp} is derived from the free body diagram of a single spine segment (see Figure 5.10(c)) as

$$\theta_v = 2\alpha_v - \pi + \arccos\left(1 - \frac{l_{st}^2}{2((l_v/2)^2 + d_1^2)}\right), \tag{5.26}$$

$$l_{st} = l_v + \Delta l_{st}, \tag{5.27}$$

$$\Delta l_{st} = (r_{sp}(q_{sp} - q_{sp0}))/4, \tag{5.28}$$

$$\alpha_v = \arctan\frac{d_1}{l_v/2}, \tag{5.29}$$

where d_1 is the distance of the tendon to the central axis, l_{st} is the effective tendon length between each spine segment, l_v is the distance between two spine vertebrate elements, Δl_{st} is the change in effective tendon length, θ_{sp} is the actuation angle of the spine servo, θ_{sp0} is the neutral position of the spine servo for which $\Delta l_{st} = 0$ (for $\theta_{sp0} = 0$), and r_{sp} is the radius of the spine servo pulley. The factor 4 in (5.29) is a result of the property that the change in tendon length is distributed equally across all four lateral joints of

the spine. Solving (5.26) for q_{sp} yields the explicit solution

$$q_{sp} = \begin{pmatrix} \frac{\sigma_2 - 4\,l_v + \sigma_1 + r_s\,q_{sp0}}{r_s} \\ -\frac{4\,l_v + \sigma_2 + \sigma_1 - r_s\,q_{sp0}}{r_s} \end{pmatrix}, \tag{5.30}$$

where,

$$\sigma_1 = 2{l_v}^2\sigma_3,$$
$$\sigma_2 = 8{d_1}^2\sigma_3,$$
$$\sigma_3 = \sqrt{-\frac{-2\cos(\theta_v)\cdot\cos(\alpha_v)^2 + 2\sin(\alpha_v)\cdot\sin(\theta_v)\cdot\cos(\alpha_v) + \cos(\theta_v) - 1}{4\,{d_1}^2 + {l_v}^2}}.$$

Lastly, we require the inverse kinematic model to compute a solution for θ_v given a desired hip deflection, which is then used to solve for q_{sp} with (5.30). Similarly to the inverse kinematic models of the legs, the inverse kinematic model of the spine is computed in the velocity space using (5.12), where the Jacobian $\boldsymbol{J} = {}^M\dot{\boldsymbol{T}}_{10}$, $\dot{\boldsymbol{p_H}}$ is the velocity of the hip with respect to the base frame {M}, and $\dot{\boldsymbol{q}} = \dot{\theta_v}$. The solution for θ_v is then computed numerically and used to solve for the spine pulley servo q_{sp} with (5.30).

5.2.2. Spine-Based Functionalities

Rodents rely on their spines to achieve dynamic locomotion and agile turning, as discussed in Section 2. In this section, we aim to derive models for several spine-based functionalities that improve the locomotion and stance behavior of Nermo. Specifically, we focus on three spine-based functionalities: 1) spine offset compensation to improve the roll behavior of Nermo when using the spine, 2) spine-based stride extension to improve the locomotion speed of Nermo, and 3) spine-based balance compensation to improve the static stability of Nermo. We use the forward and inverse kinematic models derived in the previous section as the basis for deriving the spine-based functionalities.

Spine Offset Compensation

The joint axes in the spine that enable lateral flexion are not vertical with respect to the base frame (refer to Figure 5.9), which results in an additional translation of the hip in z-direction and off-axis roll rotation. The asymmetric displacement behavior also shifts the relative neutral stance height of the rear legs. The resulting roll behavior is critical for vision tasks because the head of Nermo can only compensate for yaw and pitch orientation. Therefore, we introduce a spine offset compensation to compute the resulting z-axis hip offset z_{off_i} as

$$z_{off_i} = \left({}^M\boldsymbol{p}_{hip0_i}\right)_z - \left({}^M\boldsymbol{T}_{10}\cdot{}^{10}\boldsymbol{p}_{hip_i}\right)_z, \tag{5.31}$$

where $i \in 0, 1$ refers to the left (0) and right (1) side of the hip, ${}^M\boldsymbol{p}_{hip0_i}$ denotes the left and right neutral hip position for $\theta_v = 0$, and $\boldsymbol{p}_{hip_i}$ the position of the left and right hip

leg point in the end-point frame. The offset is then applied to the neutral stance height of the legs to compensate for the induced roll behavior.

Spine-Based Stride Extension

The lateral flexion of the spine translates the rear torso of Nermo relative to the front torso. As a result, the relative distance between the left shoulder and right hip, or right shoulder and left hip, changes. We make use of the variation in relative shoulder-to-hip distance caused by the lateral spine flexion to introduce the concept of spine-based stride extension. We took inspiration for this concept from the Gecko robot shown in Figure 2.8. Rodents also rely on the scapula for stride length extension (see Figure 5.11). The spine-based stride extension also helps replicate the scapula stride extension (since Nermo has no scapula).

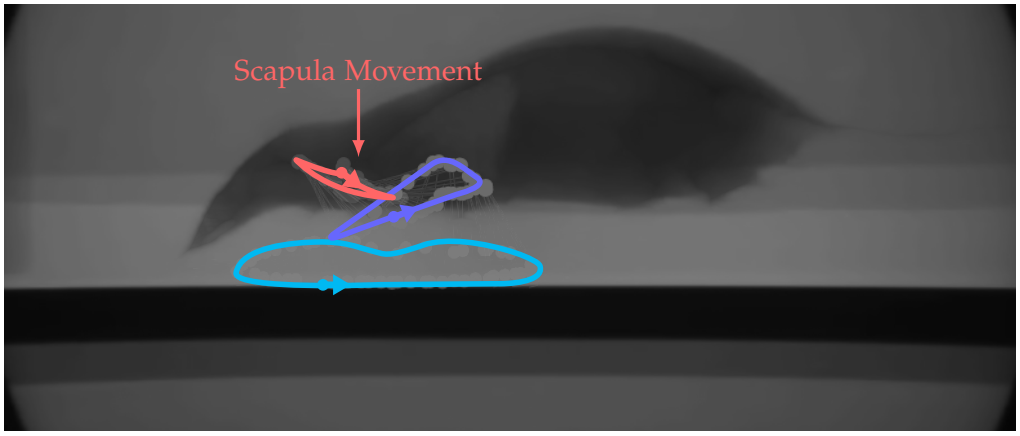


Figure 5.11.: Front leg gait cycle analysis of a mouse. The red curve depicts the displacement induced by the scapula during a full cycle of a symmetric gait.

The distance d_{ext} between a hip-shoulder pair is computed as

$$d_{ext}(\boldsymbol{p}_{sh_j}, \boldsymbol{p}_{hip_i}) = \|\boldsymbol{p}_{sh_j} - \left({}^{M}\boldsymbol{T}_{10} \cdot {}^{10}\boldsymbol{p}_{hip_i}\right)\|, \tag{5.32}$$

where $i, j \in left, right$ and $i \neq j$. The point $\boldsymbol{p}_{sh_j}$ denotes the coordinate of the shoulder in the base frame {M}. It should be noted that $\boldsymbol{p}_{sh_j}$ is constant and therefore $d_{ext}(\boldsymbol{p}_{sh_j}, \boldsymbol{p}_{hip_i}) = f(\theta_v)$. With the distance metric we are able to switch between two different modes: 1) spine-based gait extension (increasing the distance) and 2) spine-based gait compression (decreasing the distance). To enable gait extension we compute a solution for the spine vertebrate angle $\theta_v = \theta^*_{v,ext}$ by maximizing (5.32) such that

$$\theta^*_{v,ext} = \max_{\theta_v} \quad d_{ext}(\boldsymbol{p}_{sh_j}, \boldsymbol{p}_{hip_i}). \tag{5.33}$$

Gait compression is achieved by minimizing the distance defined in (5.32) to obtain a spine vertebrate angle $\theta_v = \theta^*_{v,comp}$.

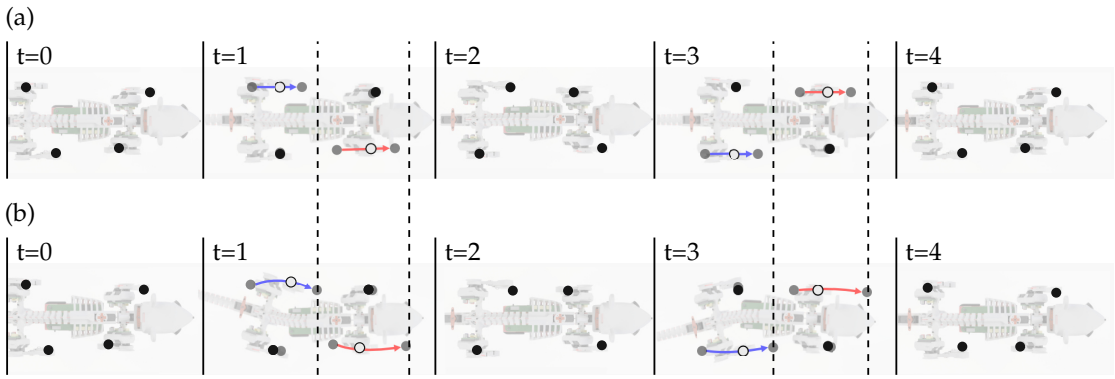


Figure 5.12.: (a) Foot placement for a standard ideal trot across a complete gait cycle. (b) Foot placement for a spine extended ideal trot across a complete gait cycle. Solid circles indicate feet in stance phase, hollow circles indicate feet in swing phase, and the arrows indicate the leg trajectory in swing.

In isolation, spine compression and spine extension have little effect on locomotion performance. However, we can increase the effective stride length by synchronizing spine extension and compression during a gait cycle. The strategy is applied to symmetric gaits (trots) according to two basic rules: 1) for an opposing leg pair in swing state, we apply spine-based gait compression (effectively increasing the length of their swing by forward yaw translation), and 2) for an opposing leg pair in stance state we apply spine-based gait extension (effectively increasing the length of their stance by rearward yaw translation). Figure 5.12 clearly demonstrates the spine-based stride extension effect by comparing a standard ideal trot gait (Figure 5.12(a)) with a spine extended ideal trot (Figure 5.12(b)). From Figure 5.12, we observe that: a) the stride length with spine-based extension is greater than for the standard gait, and b) the swing leg trajectory with the spine-based extension has a characteristic curve shape compared with the linear trajectory of the standard gait. It can be concluded that by increasing the effective stride length, the spine-based stride extension is an alternative strategy for increasing the gait speed.

Modeling of Center of Mass in Nermo

The flexion of the compliant spine constantly influences the position of the center of mass (COM) of Nermo. This behavior occurs because the spine connects and translates two mass bodies, each with its local COM. The mass of the rear torso is 60 g, and the mass of the front torso is 225 g. Therefore, as the spine is actuated, the front torso (with its COM) is translated relative to the rear torso (with its COM), which results in a shift of the Nermo's total COM.

The total COM $\boldsymbol{p}_{com}$ is derived by considering the rear and front torso as two bodies

connected by a massless spine, such that

$$\boldsymbol{p}_{com} = \frac{1}{m_{fr} + m_{rr}} \left[m_{fr} \boldsymbol{p}_{com_{fr}} + m_{rr} ({}^{M}\boldsymbol{T}_{10} \cdot {}^{10}\boldsymbol{p}_{com_{rr}}) \right], \tag{5.34}$$

where m_{fr} and m_{rr} are the masses of the front and rear torso of Nermo, $\boldsymbol{p}_{com_{fr}}$ denotes the position of the COM of the front torso, and ${}^{10}\boldsymbol{p}_{com_{rr}}$ denotes the position of the COM of the rear torso in the reference frame of the hip. With (5.34) the change in COM due to the lateral flexion of the spine can be predicted because $\boldsymbol{p}_{com} = \boldsymbol{f}(\theta_v)$. Understanding the influence of the spine flexion on the COM is important for predicting the stability behavior of Nermo. The spine is considered as massless in the COM model because its mass constitutes less than 2 % of the total mass. The formal model of the COM can now be used to describe the static balance behavior of Nermo.

Spine Balance Compensation

The position of the COM relative to the contact points of the feet determines whether Nermo is in a stable or unstable stance configuration. We have already shown that the COM model from (5.34) describes the COM position with respect to the spine vertebrate angle θ_v and, by extension, spine pulley actuation q_{sp}. Therefore, by extension, the COM position model can be used to derive a model that controls the stability of Nermo in the three relevant contact configurations: 1) four-foot contact, 2) three-foot contact, and 3) two-foot contact. Each contact configuration has unique stability conditions (illustrated in Figure 5.13). Therefore, the spine can be used as a mechanism for balance compensation.

The different contact scenarios outlined in Figure 5.13 highlight the importance of the COM position relative to the stability diagonal for achieving a stable stance position. The stability diagonal is the line vector that joins the opposing front and rear foot positions. It is conceptually similar to the opposing foot distance computed for spine-based stride extension. In four-foot contact, Nermo is always statically stable (unless at extreme roll or pitch angles) because the COM is always within a stability triangle enclosed by the stability diagonal. Therefore, we omit the four-foot stability case. We observe from Figure 5.13 that in the three-foot contact scenarios, Nermo can transition between static stability and instability depending on the COM position. The COM lies within the stability triangle enclosed by the stability diagonal when both rear feet and one front foot is in contact (stable configuration). Conversely, Nermo is statically unstable for cases with two front feet and one rear foot in contact because the COM is outside the stability triangle enclosed by the stability diagonal. Therefore, actuating the spine to translate the COM relative to the stability diagonal influences the static stability of Nermo. For the two-foot contact scenario, Nermo can be in dynamic stability/instability. Dynamic stability is achieved when the COM is actuated such that the COM coincides with the stability diagonal. Similarly, the COM can be actuated to not coincide with the stability line to induce a pivot behavior tangential to the stability diagonal (unstable configuration).

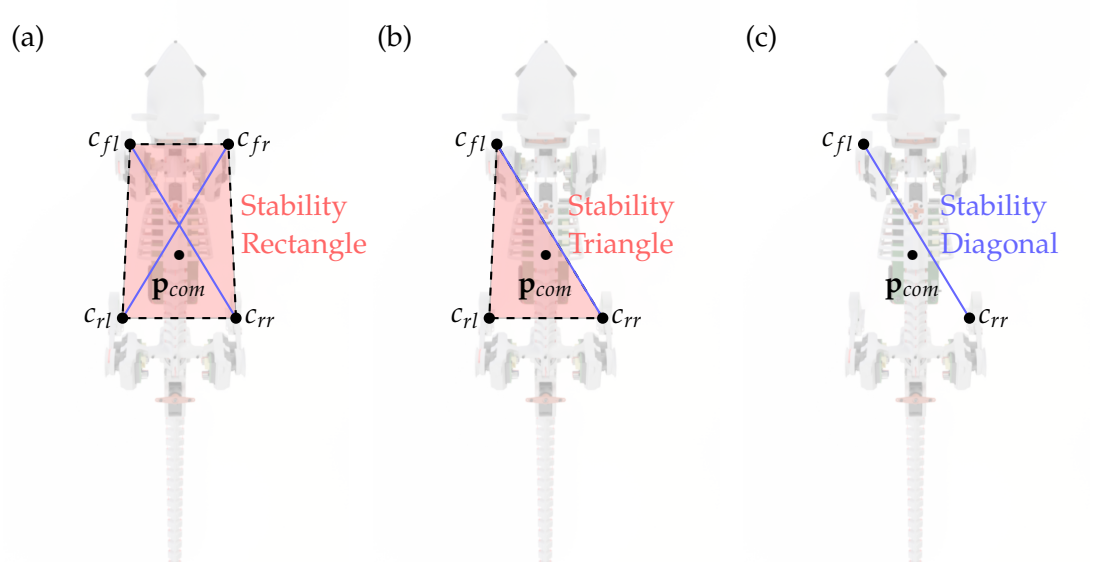


Figure 5.13.: Illustration of the different contact configurations: (a) four-foot contact, (b) three-foot contact with triangle stability, and (c) two-foot contact with diagonal stability line.

The observations from Figure 5.13 clearly outline the dependence of the static stance stability on the position of the COM relative to the stability diagonal. We can describe the COM position relative to the stability diagonal using a distance metric d_{stab} defined as

$$d_{stab} = \frac{\left| a\boldsymbol{p}_{com,x} + b\boldsymbol{p}_{com,y} + c \right|}{\sqrt{a^2 + b^2}}, \tag{5.35}$$

where

$$\begin{aligned} a &= \boldsymbol{p}_{fe,y} - \boldsymbol{p}_{re,y}, \\ b &= \boldsymbol{p}_{re,x} - \boldsymbol{p}_{fe,x}, \\ c &= \boldsymbol{p}_{fe,x}\boldsymbol{p}_{re,y} - \boldsymbol{p}_{fe,y}\boldsymbol{p}_{re,x}. \end{aligned}$$

The parameters $\boldsymbol{p}_{com,x}$ and $\boldsymbol{p}_{com,y}$ denote the x- and y-projection of the COM onto the ground plane. The parameters $\boldsymbol{p}_{fe}$ and $\boldsymbol{p}_{re}$ denote the known front and rear foot-end positions of the opposing feet contact pair defining the stability line. Therefore, the distance $d_{stab} = f(\theta_v)$ can be solved for θ_v and then for q_{sp} with (5.30).

Additionally, the stability distance d_{stab} is a signed function that provides information on the relative orientation between the COM position and stability diagonal. For example, if the COM is located to the right of the stability diagonal, then $d_{stab} > 0$, while on the left of the stability diagonal $d_{stab} < 0$. The signed function property provides the necessary information to translate the COM into the desired stability configuration. The

desired stability configurations are reached by finding a solution $\theta_v = \theta_v^*$ for (5.35) that satisfies one of the following stability conditions:

- **Two-Foot Contact Dynamic Stability:** Find a solution $\theta_v = \theta_v^*$ for which the COM position coincides with stability diagonal such that $d_{stab}(\theta_v) = 0$.
- **Three-Foot Contact Dynamic Stability:** Find a solution $\theta_v = \theta_v^*$ for which the COM position coincides with stability diagonal such that $d_{stab}(\theta_v) = 0$.
- **Three-Foot Contact Static Stability:** Find a solution $\theta_v = \theta_v^*$ for which the COM position is inside the stability triangle such that $d_{stab}(\theta_v) = d_{lift}$. We pick d_{lift} as a scalar that defines the distance of the COM inside the stability triangle, and the signed identifier for the stability triangle (i.e. is the COM located to the left/right of the stability diagonal). We can also formulate the condition such that $\theta_v^* = \max_{\theta_v} d_{stab}$ which finds a solution that pushes the COM maximally inside the stability triangle.

5.3. Turning Strategies

Thus far, we have covered the leg and spine models beneficial for straight-line locomotion. Equally crucial for navigation is the ability of a quadruped robot to turn. Current state-of-the-art quadruped robots have 3 DOF per leg, allowing lateral foot positioning to control yaw orientation directly. Such direct yaw orientation control is not possible as the 2 DOF legs do not allow lateral displacement. Therefore, we introduce the two control strategies for underactuated quadrupeds to control the yaw orientation: leg-based and spine-based lateral flexion turning.

5.3.1. Leg-Based Turning

The inspiration for leg-based turning is the concept of torque vectoring. Torque vectoring is used in automobiles to induce a yaw moment around the COM of a vehicle by varying the wheel torque. The induced yaw moment improves the turning performance. We apply this concept to quadruped robots by varying the stride length instead of the torque. Varying the stride length introduces an asymmetric contact force distribution, which induces a yaw moment around the COM (illustrated in Figure 5.14).

The control of the leg-based turning is trivial. We apply a linear ramp response to the stride length modulation based on the turn-rate control signal α_{tr} from the high-level controller. We apply the leg modulation parameters

$$\alpha_{left} = \min(1 - \alpha_{tr}, 1), \tag{5.36}$$

$$\alpha_{right} = \min(1 + \alpha_{tr}, 1), \tag{5.37}$$

where α_{left} modulates the left legs and α_{right} modulates the right legs. It should be noted that α_{tr} is normalized such that $-1 \leq \alpha_{tr} \leq 1$. Therefore, the stride length goes to zero for a maximum turn input from the high-level controller (for either right or left).

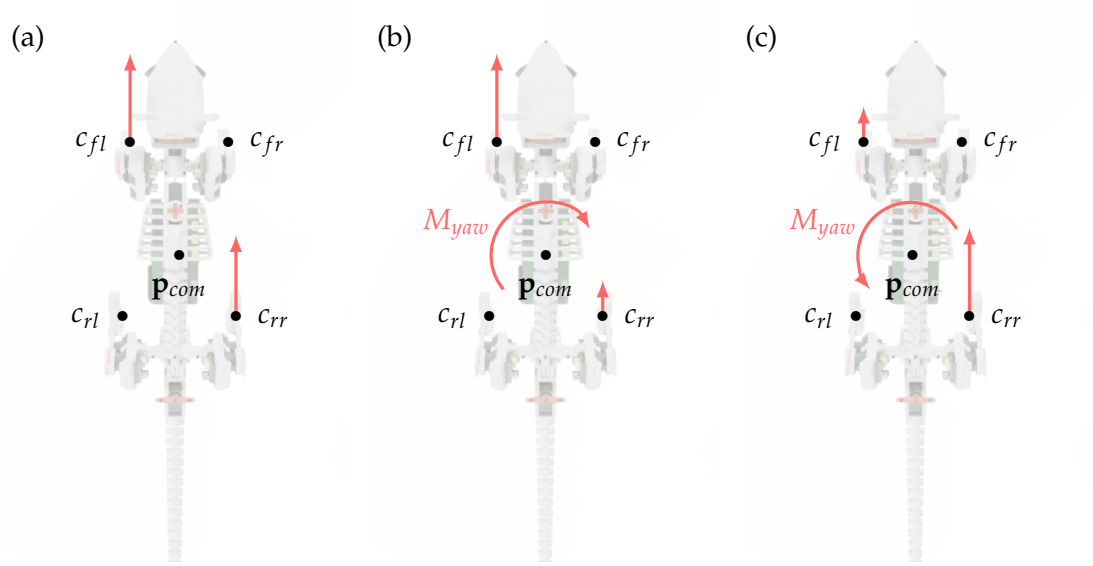


Figure 5.14.: Illustration of leg-based turning for trot gait with two-foot contact. (a) Case with no stride length modulation resulting in no yaw moment. (b) Right turning yaw moment caused by decreased rear leg stride length. (c) Left yaw moment turning case.

The leg modulation factors can be generalized to

$$\alpha_{left,t} = \min(1 - \alpha_{tr}^{t}, 1), \tag{5.38}$$
$$\alpha_{right,t} = \min(1 + \alpha_{tr}^{t}, 1), \tag{5.39}$$

where $t = \{t \in \mathbb{R} | t \geq 0\}$ is the response curve adjustment factor. Using values $t > 1$ results in a delayed response curve, whilst $t < 1$ results in a low turn-rate sensitive response. Leg modulation is turned off for $t = 0$. The factor t can be tuned on-the-fly to adjust the turn-rate response. We model the turning radius $r_{turn,lb}$ similar to that of the gecko robot from (2.1) as

$$r_{turn,lb} = \frac{l_s \cdot \sin(\frac{\pi - \theta_{lb}}{2})}{\sin(\theta_{lb})}, \tag{5.40}$$
$$\theta_{lb} = \frac{l_s \Delta m}{d_{diag}}, \tag{5.41}$$
$$\Delta m = |\alpha_{left,t} - \alpha_{right,t}|, \tag{5.42}$$

where l_s is the stride length, θ_{lb} the stride rotation, d_{diag} the distance between alternate leg pairs, and Δm the modulation difference between the left and right legs. The above equations do not hold for the case where $\alpha_{left,t} = \alpha_{right,t}$ (no leg modulation).

5.3.2. Spine-Based Lateral Flexion Turning

The second turning strategy takes advantage of the lateral spine flexion by mimicking the turning behavior of rodents. The induced spine lateral flexion displaces the rear legs and changes the yaw orientation by angle θ_{turn} as illustrated in Figure 5.15. We call this spine-based turning. Turning is enabled by the changed shoulder to hip geometry which induces an asymmetric yaw moment around the COM. Trivially, $\theta_{turn} = 4\theta_v$, where θ_v is the joint angle of the spine vertebrate introduced in Section 5.2.1.

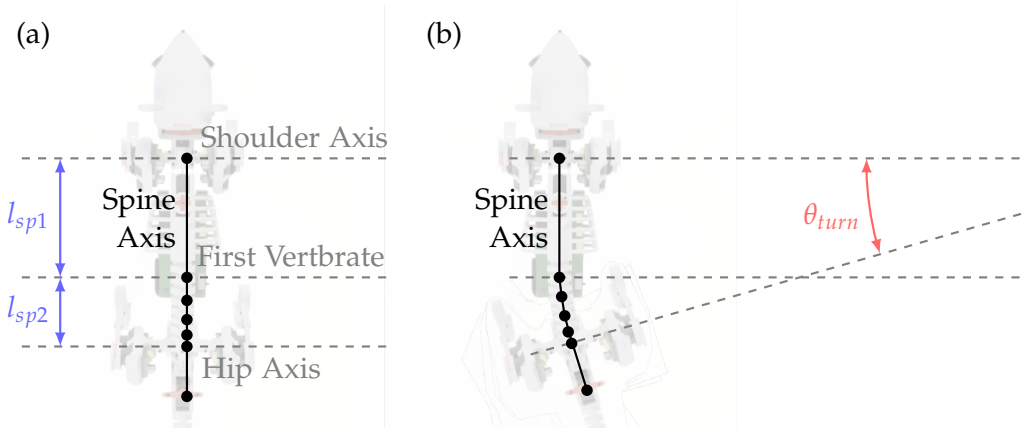


Figure 5.15.: Illustration of the spine-based turning. The shoulder and hip axes are parallel in (a) with no turning radius induced by the spine. (b) Spine-based lateral flexion turning. The shoulder and hip axis are no longer parallel intersecting at the center of rotation with turning angle $\theta_{turn,sb}$.

From Figure 5.15 we derive the turning radius from the shoulder axis $r_{turn,f}$ and turning radius from the hip axis $r_{turn,r}$ as

$$r_{turn,f}(\theta_{turn}) = \frac{l_{sp2}}{\theta_{turn}} + \frac{l_{sp1}}{\tan(\theta_{turn})}, \tag{5.43}$$

$$r_{turn,r}(\theta_{turn}) = \frac{l_{sp2}}{\theta_{turn}} + \frac{l_{sp1}}{\sin(\theta_{turn})}, \tag{5.44}$$

where l_{sp1} is the length of the spine from shoulder axis to first lateral vertebrate and l_{sp2} the length from first vertebrate to hip axis (which remains constant, even under flexion). It should be noted that

$$r_{turn,f}(\theta_{turn}) \xrightarrow[\theta_{turn}\to 0]{} \infty,$$

$$r_{turn,r}(\theta_{turn}) \xrightarrow[\theta_{turn}\to 0]{} \infty,$$

because the shoulder and hip axes are parallel for no spine flexion. In this thesis we evaluate the turning radius of the shoulder axis for spine-based turning such that $r_{turn,sb} = r_{turn,f}(\theta_{turn})$.

5.3.3. Mix-Based Turning

The two turning strategies we have introduced have several benefits and drawbacks as outlined in Table 5.4. We introduce a mix-based turning strategy that combines the benefits of both turning strategies.

Table 5.4.: Qualitative comparison of the leg-based and spine-based turning.

Turning Strategy	Benefits	Drawbacks
Leg-based turning	Large range of turning radii. Independent leg control.	Loss of speed from leg modulation. Friction sensitive around pivot leg.
Spine-based turning	High speed by not modulating stride length.	Limited range of turning radii. Friction sensitive around inner legs.

The mix-based turning applies the following logic for combining both turning strategies. For low turn-rates from the high-level controller, the mix-based turning prioritizes the faster spine-based turning. For large turn-rates, i.e., smaller turning radii, the mix-based strategy applies additional leg-based turning. The behavior of the mix-based strategy can be tuned with the curve adjustment factor t introduced for leg-based turning. With a factor $t > 1$, the leg-based turning responds for large turn-rate values. The turn radius is computed in the same way as for leg-based turning, except that $d_{diag} \neq constant$. Instead, d_{diag} is dependent on the spine vertebrate angle θ_v.

5.4. Control Architecture

In this section, we cover the control architecture used for the digital twin and robot. The control architecture integrates the leg and spine models derived in the previous section into a cohesive system to achieve quadruped locomotion. The control architecture depicted in Figure 5.16 consists of four main components: 1) the high-level controller (see Appendix A.1.4), 2) the motion module, 3) the trajectory generator, and 4) the gait parameter knowledge-base (see Appendix A.1.4). We focus on the motion module and trajectory generator within the scope of this thesis.

The control architecture relies on highly synchronized communication between all modules to achieve successful control. In Figure 5.16 we show the information flow between the main modules where: $\boldsymbol{k}_g$ is the gait parameter vector, $\boldsymbol{s}_{legs}$ is the leg states vector, $\boldsymbol{v}_{legs}$ is the leg velocity vector, $\boldsymbol{tr}_{foot}$ is the foot-end trajectory, v_m is the control velocity, α_{tr} is the turn-rate, and g_d is the gait mode. The controller has two external interfaces: 1) to an upstream input device that sends control messages $\boldsymbol{c}_m$ and 2) to the simulation and Nermo with control $\boldsymbol{q}_{m,d}$ and sensor $\boldsymbol{q}_{m,s}$ messages.

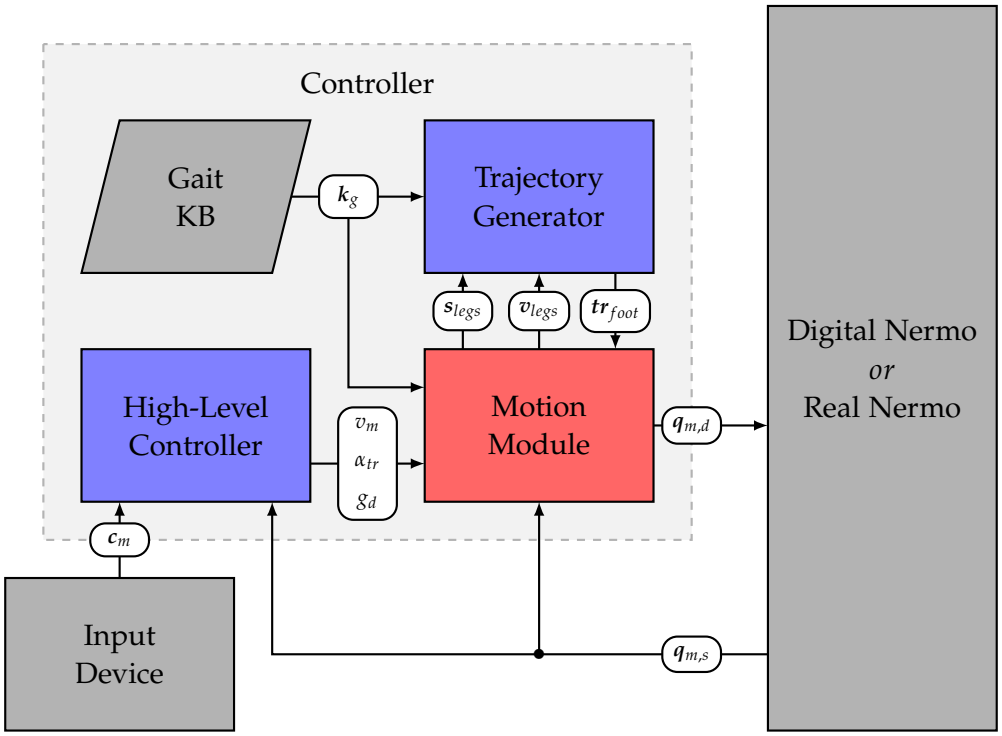


Figure 5.16.: Control architecture of Nermo. The controller consists of four core modules: 1) the high-level controller, 2) the motion module, 3) the trajectory generator, and 4) the gait knowledge-base. The controller interfaces with two external components: 1) any upstream input device that sends a control message $\boldsymbol{c}_m$, and 2) sending servo controls $\boldsymbol{q}_{m,d}$ and receiving sensor values $\boldsymbol{q}_{m,s}$ from digital and/or real Nermo.

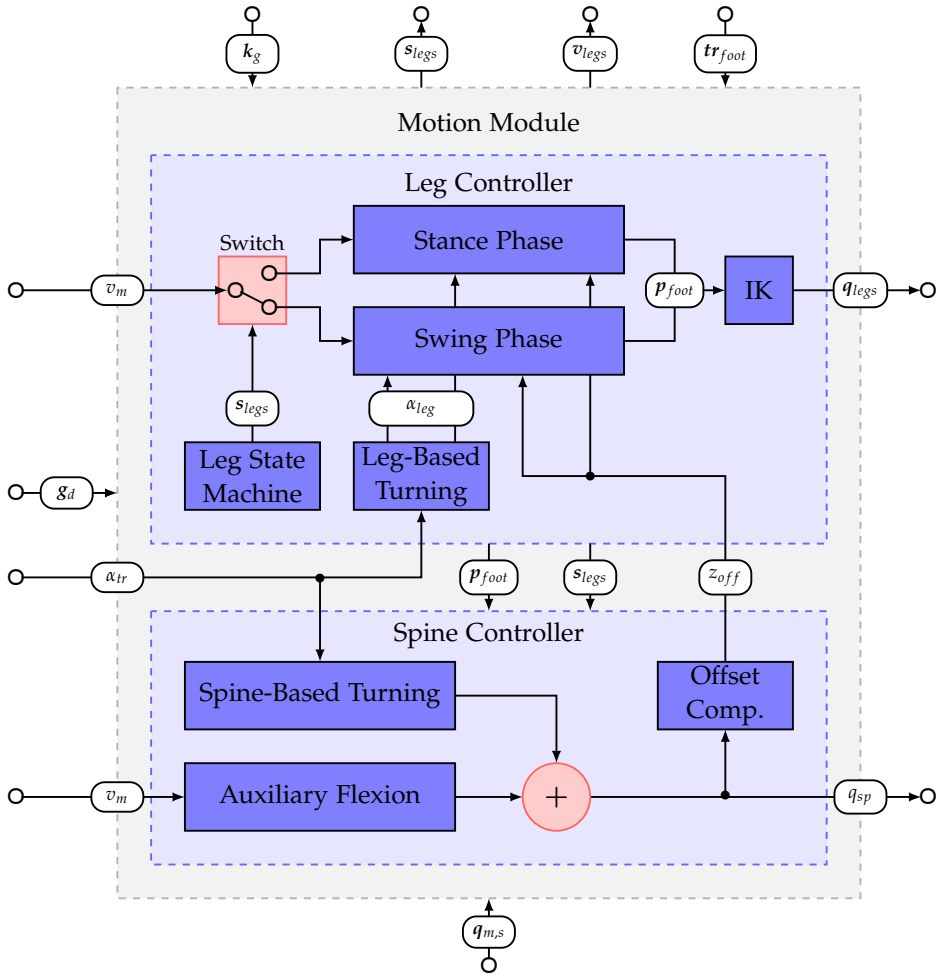


Figure 5.17.: The architecture of the motion module. The motion module computes the low-level control commands for the servo motors.

5.4.1. Motion Module

The motion module implements the leg and spine models covered in Section 5.1.2 and Section 5.2.1. In particular, the motion module must ensure that the leg and spine are synchronized for all behaviors. Figure 5.17 depicts the motion module and the interdependencies between the low-level leg and spine controllers.

In addition to the internal communication between the leg and spine controller, the motion module's main interface is sending the servo control values $\boldsymbol{q}_{m,d}$ to the downstream device. The motion module also interfaces with the gait knowledge-base module to retrieve relevant gait parameters $\boldsymbol{k}_g$, and the trajectory generator to obtain the foot-end trajectories tr_{foot}. The external and internal control signals of the motion module are outlined in Table 5.5. It should be noted that in Figure 5.16 the parameter $\boldsymbol{q}_{m,d} = (\boldsymbol{q}_{legs}, q_{sp})$.

Table 5.5.: Description of the input, internal, and output signal parameters from the motion module controller.

Signal Parameter	Description	Unit
$\boldsymbol{c}_m$	Target velocity and position	m/s and m
v_m	Adjusted velocity	m/s
α_{tr}	Normalized target turn-rate	-
$\boldsymbol{g}_m$	Vector of gait mode parameters: gait type g_t, gait frequency g_f, spine modes sp_m,and leg-based turning adjustment parameters t	-
$\boldsymbol{q}_{m,s}$	Sensor values: servo positions, IMU	-
$\boldsymbol{k}_g$	Gait and model parameters	-
$\boldsymbol{s}_{legs}$	Leg state vector (swing or stance)	-
$\boldsymbol{v}_{legs}$	Leg velocity vector	m/s
$\boldsymbol{tr}_{foot}$	Foot-end trajectory object	m
$\boldsymbol{x}_{foot}$	Target foot-end position	m
z_{off}	Z-axis rear leg offset	m
α_{leg}	Per leg modulation value	-
$\boldsymbol{q}_{legs}$	Leg servo control values	-
q_{sp}	Spine servo control value	-

Spine Controller

The spine controller is responsible for computing and combining the control values for the spine-based functionalities. The spine-based functionalities are divided into two modes: 1) the spine-based turning (primary mode) controlled by α_{tr} and 2) the spine-based auxiliary flexion modes (secondary mode). The auxiliary flexion modes include the spine-based gait extension and balance compensation. The primary and secondary modes are enabled or disabled by the behavior control value $\boldsymbol{g}_m$ from the high-level controller.

One problem that arises from having multiple spine modes enabled is cross actuation interference. For example, the spine-based stride extension may require an opposite spine flexion than the spine-based turning resulting in a signal block scenario. To avoid this problem, the spine controller handles the desired spine flexion from the primary and secondary modes via a scaled fusion controller defined as

$$q_{sp} = q_{max} \cdot [\alpha_{tr} + \alpha_2^s \cdot f_{sp}], \tag{5.45}$$

$$\alpha_2 = 1 - \alpha_{tr}, \tag{5.46}$$

where q_{sp} is the desired spine servo actuation angle, q_{max} the maximum spine servo actuation, α_{tr} the turn-rate value from the high-level controller, $s = \{s \in \mathbb{R} | s \geq 0\}$ as the response curve factor, and f_{sp} the spine actuation value from the auxiliary modes. By adjusting the response curve factor s, we tune the controller to prioritize between primary and secondary modes. For $s > 1$, the fusion controller prioritizes the turning commands. Within the scope of this thesis $s = 1$. The computed q_{sp} value is passed onto the downstream devices as the desired servo angle command.

The output value q_{sp} is also passed to the spine offset compensation block. As touched on in Section 5.2.1, lateral spine flexion leads to a vertical displacement and roll of the hip, which changes the neutral stance height of the rear legs. The offset compensation block computes the z_{off} value to compensate for the changed stance height and roll. The offset compensation is enabled or disabled by the $\boldsymbol{g}_d$ parameter. The computed spine offset value z_{off} for each leg is passed to the leg controller as an offset parameter for the leg trajectories.

Leg Controller

The leg controller handles the low-level control of each leg to reach the target velocity v_m and turn-rate α_{tr} sent from the high-level controller. The control flow for each leg occurs in three steps. First, the ideal foot-end position is computed based on the current leg phase and foot-end trajectory. Second, the target foot-end position is evaluated by applying offset and adjustment trajectories (such as the z_{off} from the spine offset compensation). Last, the servo control values $\boldsymbol{q}_{legs}$ are computed from the target foot-end position in the inverse kinematic (IK) block.

Two of the three steps in leg controller involve the computation of the target foot-end position from the ideal foot-end position. The ideal foot-end position is computed

from an ideal foot-end trajectory generated by the trajectory generator module, which depends on the leg state (stance or swing). The leg state is determined by the leg state machine. The leg state machine is a timed automaton that tracks the state of each leg and initiates the transition between the stance and swing phase. The state of each leg is derived from the gait parameters passed to the motion module from the gait knowledge-base. In particular, the leg state machine translates the gait frequency, duty cycle, and leg phase offsets (i.e., the offsets between each leg to achieve a desired gait pattern) into a state control signal $\boldsymbol{s}_{legs}$. The parameter $\boldsymbol{s}_{legs}$ also contains a transition flag to signal a leg state switch. The transition flag flips the state switch between stance or swing and sends a control signal to the trajectory generator to generate a new leg trajectory for the next gait cycle. Therefore, the state machine is responsible for synchronization between the leg and spine controller.

5.4.2. Trajectory Generator

The trajectory generator receives gait, leg state, and leg velocity parameters to generate an ideal foot-end trajectory. A trajectory for each foot is generated when a leg switches from stance to swing state or vice-versa. A memory module handles the state switch internally by comparing the current leg state with the previous leg state. Figure 5.18 shows that the trajectory generation happens in two functional blocks.

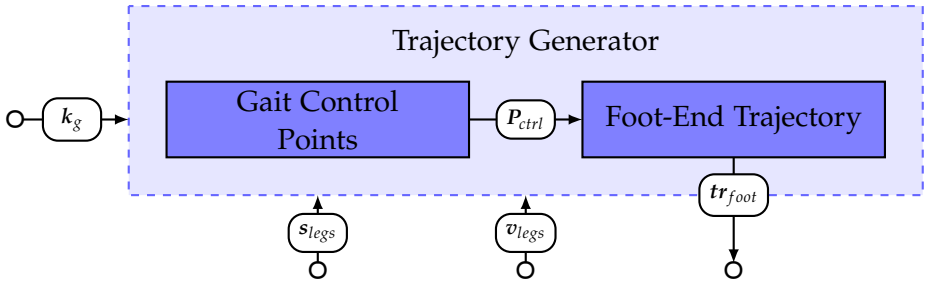


Figure 5.18.: The architecture of the trajectory generator module. The trajectory generator interfaces with the gait knowledge-base and motion module.

The first functional block, the gait control point block, computes a set of control points $\boldsymbol{p}_i = (p_{i,y}, p_{i,z})^T$ for the next gait cycle and leg phase (see Figure 5.19). The control points are computed based on the foot-end parameters: neutral stance height l_{nsh}, neutral stance position l_{nsp}, forward l_{fs} and rear l_{rs} stride length, and swing a_{sw} and stance phase a_{st} amplitude. The control points are constrained in the horizontal direction by a maximum stride length parameter. The numerical values used to compute the control points can be found in Appendix A.4.1. It should be noted that the same parameters apply to the rear leg, with the only difference being that the parameters are centered around the hip axis and not the shoulder axis. The control points must satisfy positional continuity of the foot-end between the transition from swing to stance phase. Therefore, the generated control points establish a partial gait symmetry, where the end half of

the previous half-gait cycle is the first half of the next half-gait cycle. Each swing and stance phase cycle has three control points: 1) starting point $\boldsymbol{p}_i$, 2) mid-phase point $\boldsymbol{p}_{i,m}$ (centered around the neutral axis), and 3) the endpoint $\boldsymbol{p}_{i+1}$. The point $\boldsymbol{p}_n$ is the neutral foot-end position. In Figure 5.19 we only depict the control points for the swing phase for clarity.

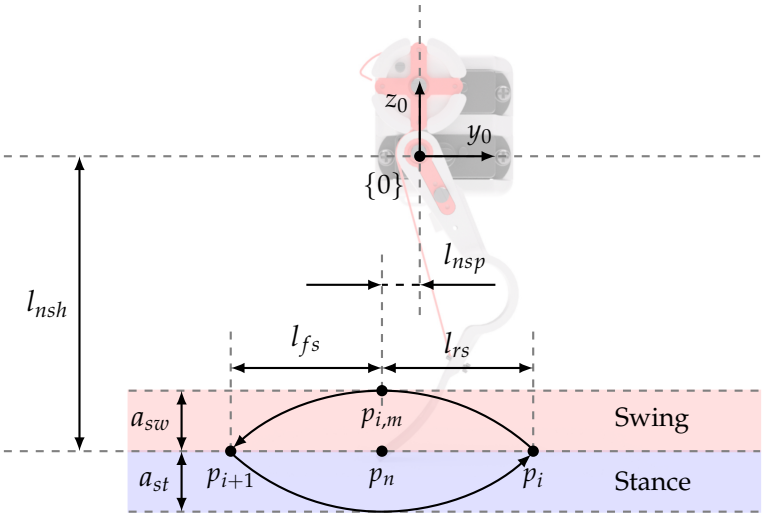


Figure 5.19.: Illustration of the leg parameters used to compute the foot-end control points. In stance phase the

Second functional control block computes the trajectory from the control points. Specifically, the foot-end trajectory block generates an ideal foot-end trajectory $\boldsymbol{tr}_{foot}(t)$ by fitting a time-dependent motion spline to the control points. Initially, we explored three different trajectory curve types: linear, parabolic, and cubic. The linear and parabolic trajectory curves are easier to define and offer positional continuity with the downside of not controlling velocity and acceleration at foot touchdown and lift-off. Therefore, we used a cubic trajectory for fitting the control points. The cubic trajectory is defined as

$$\boldsymbol{tr}_{foot}(t) = \begin{cases} \boldsymbol{tr}_{foot,1}(t), & 0 \leq t < \frac{T}{2} \\ \boldsymbol{tr}_{foot,2}(t), & \frac{T}{2} \leq t < T \end{cases}, \tag{5.47}$$

where T is the period of the half-gait cycle and $\boldsymbol{tr}_{foot,i}(t) = (y_i(t), z_i(t))^T$ for $i \in [1,2]$ defines the 2D foot-end coordinates relative to the hip/shoulder axis. The general form of the sub-trajectories is given as

$$\boldsymbol{tr}_{foot,i}(t) = \boldsymbol{a}_i t^3 + \boldsymbol{b}_i t^2 + \boldsymbol{c}_i t + \boldsymbol{d}_i, \tag{5.48}$$

where $\boldsymbol{a}_i$, $\boldsymbol{b}_i$, $\boldsymbol{c}_i$, and $\boldsymbol{d}_i$ are the cubic spline parameters.

We introduce the boundary conditions listed in Table 5.6 to construct the cubic spline trajectories from the control points $\boldsymbol{p}_i$, $\boldsymbol{p}_{i,m}$, and $\boldsymbol{p}_{i+1}$. The first four conditions ensure the spline satisfies positional continuity. For the y-position, the boundary conditions focus

on velocity continuity at the mid-point and introduce the constraint of no horizontal velocity at start and endpoints. Contrary, the z-position boundary conditions focus on constraining vertical acceleration at the start and endpoints. The two cubic sub-trajectories are solved using the boundary conditions from Table 5.6 and (5.48). Figure 5.20 shows a sample solution of the cubic spline trajectory compared with the parabolic and linear splines. The solved foot-end trajectory $\boldsymbol{tr}_{foot}(t)$ is sent to the motion module for further processing. As outlined previously, any further adjustments to the foot-end trajectory are applied in the motion module.

Table 5.6.: The boundary conditions for the cubic spline fitting of the control points.

Y-Conditions	Z-Conditions
$y_1(0) = p_{i,y}$	$z_1(0) = p_{i,z}$
$y_1(\frac{T}{2}) = p_{i,m,y}$	$z_1(\frac{T}{2}) = p_{i,m,z}$
$y_2(\frac{T}{2}) = p_{i,m,y}$	$z_2(\frac{T}{2}) = p_{i,m,z}$
$y_2(T) = p_{i+1,y}$	$z_2(T) = p_{i+1,z}$
$\dot{y}_1(0) = 0$	$\ddot{z}_1(0) = 0$
$\dot{y}_2(T) = 0$	$\ddot{z}_2(T) = 0$
$\dot{y}_1(\frac{T}{2}) = \dot{y}_2(\frac{T}{2})$	$\dot{z}_1(\frac{T}{2}) = \dot{z}_2(\frac{T}{2})$
$\ddot{y}_1(\frac{T}{2}) = \ddot{y}_2(\frac{T}{2})$	$\ddot{z}_1(\frac{T}{2}) = \ddot{z}_2(\frac{T}{2})$

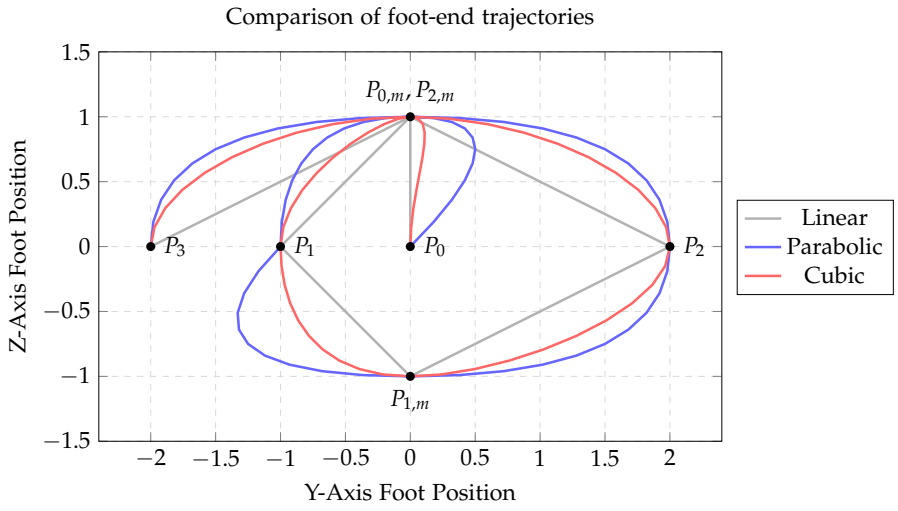


Figure 5.20.: Sample control points and foot-end trajectory. The amplitude for swing and stance phase is normalized to 1 and -1 respectively. The neutral stance position P_0 is centered at the origin.

6. Experiments

In this chapter, we first present the choice of simulation environment and outline shortly the steps taken to reduce the gap between the simulation and reality. Second, we outline the four test case scenarios designed to evaluate the gait and spine performance: straight-line tests, turn tests, balance tests, and a combined maze navigation test. Lastly, we cover the test results for both the simulation and the real-world tests.

6.1. Test Scenario Setup

6.1.1. Simulation Engine

We implement the digital twin in the physics engine MuJoCo. MuJoCo has several advantages compared with other simulation engines, as touched on in section 2.3: 1) handling multi-joint dynamics and contact physics engine, 2) providing a framework and modeling paradigm for tendon driven components, and 3) the widespread use of MuJoCo in reinforcement learning tasks as an open-source physics engine. The last point is crucial as it guarantees the longevity of the digital twin beyond the research conducted in this thesis. We use the high-level controller to maintain the desired path in the test scenarios that require a predefined path to be followed.

We applied two strategies to develop a realistic and modular digital twin of Nermo. First, we implemented each core component of Nermo as a modular functional block in MuJoCo. We combined the modular functional blocks in a central model using the MuJoCo XML framework. Second, we run two concurrent models of Nermo in MuJoCo: a model using geometric primitives to handle collisions and a second model for high fidelity visual representation of Nermo. We explain the two modeling strategies in detail in Appendix A.3.

6.1.2. Setup for Improving the Simulation to Reality Gap

The reality gap is an often occurring phenomenon that describes the disparity between behavior and results found in simulation and the real world (see Section 2.3). We have applied two strategies to reduce the reality gap between the digital twin and Nermo.

First, we developed a modular software architecture built with ROS to enable flexible transition between running the controller in simulation and the real world. The implemented architecture is shown in Figure 6.1. The controller computations are carried out on a central computer, which is also used to execute the MuJoCo simulation. The control commands are sent wirelessly to Nermo using the ROS TCP/IP communication

backend. A custom wireless network enables communication between Nermo and the central computer. Therefore, the modular software architecture enables us to control the digital twin and Nermo from the identical controller.

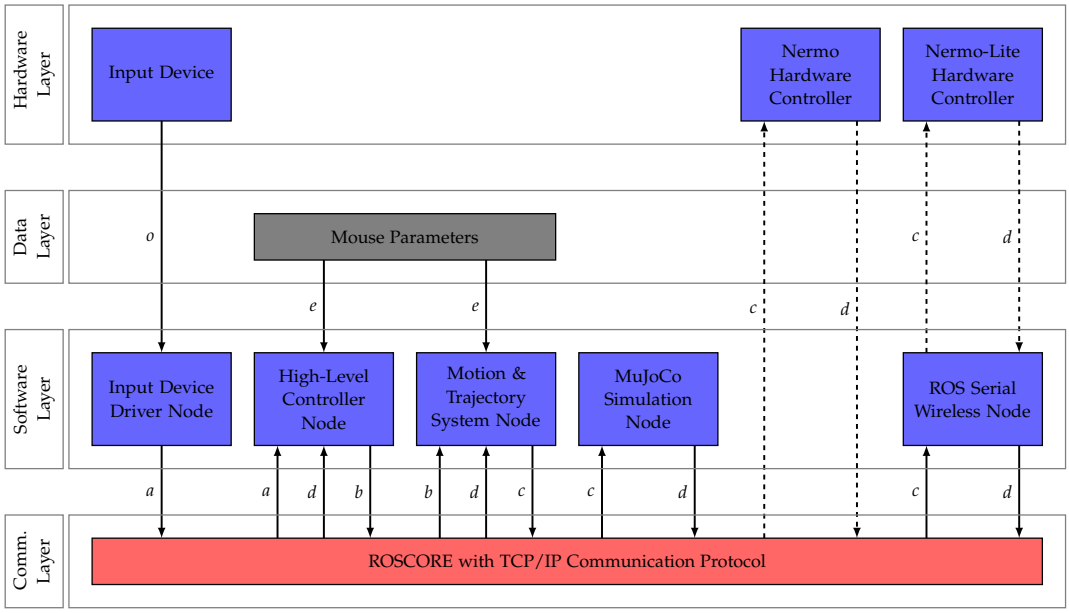


Figure 6.1.: Overview of the implemented ROS architecture for controlling the MuJoCo simulation, Nermo and Nermo lite. Dashed arrows indicate wireless communication. The letters represent different topics communicated across the ROS backend corresponding to: *o* for raw input data, *a* for processed control topic, *b* for target velocity command topic, *c* for servo command topic, *d* for sensor data topic, and *e* for shared model parameters. The communication with the ROS backend occurs via topics.

Second, we developed a modified version of Nermo, Nermo-Lite (see Figure 6.2). Nermo-Lite has four main mechanical and electrical design modifications compared with Nermo. First, the compliant spine in Nermo-Lite no longer allows vertical flexion, just like the simulation model. Second, the leg mounting points are rotated to improve tendon routing around the shoulder and hip joints. Third, we designed a new rear leg that follows the three-bar linkage model used in the kinematic model. Although not within the scope of this thesis, the new rear leg model has shown early signs of significant performance improvement over the previous leg. Lastly, we simplified the electrical architecture of Nermo-Lite to use unmodified servo motors and a real-time microcontroller. We made these choices for two reasons: 1) to reduce the manufacturing error resulting from modifying the servos and 2) to offload the main computations to a central computer and instead achieve real-time servo control using a microcontroller. We explain the changes in further detail in Appendix A.2.1.

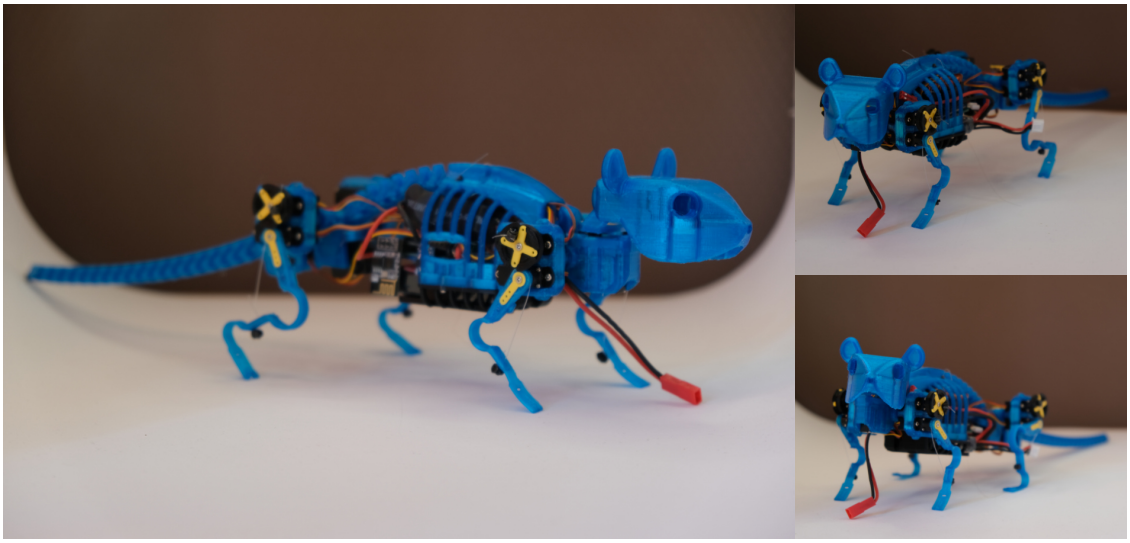


Figure 6.2.: Overview of Nermo-Lite in different views. Nermo-Lite is a modified version of Nermo that: 1) has fewer servo motors (unmodified) to simplify the electrical design, 2) has an improved mechanical design, and 3) is 80 % cheaper than Nermo. Nermo-Lite is fully self-contained, untethered, and autonomous capable.

6.1.3. Straight-Line Test Setup

We designed the straight-line tests to evaluate the gait speed and gait orientation stability with and without the spine. The overall test setup is shown in Figure 6.3. The distance between start and end is 2.0 m in the simulation and 1.5 m in the real-world tests (due to space constraints). In the real-world tests, we used a rubberized surface mat to replicate the high contact friction used in the simulation.

The methodology for the straight-line tests is as follows:

1. Nermo is placed at the starting point as shown in Figure 6.3.
2. A control signal is sent to begin the test, and the high-level controller sends a constant forward velocity command to the motion module. Concurrently, the high-level controller sends turn-rate commands to ensure Nermo moves along a straight line during the test.
3. The test is completed when Nermo crosses the designated end line as shown in Figure 6.3.
4. Steps 2-3 are repeated three times for each parameter set.
5. Steps 2-4 are repeated for each new parameter set.

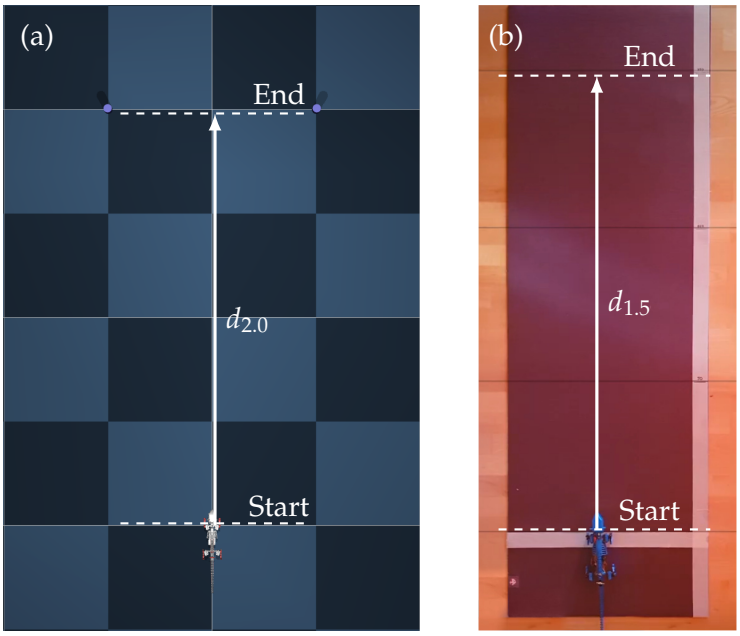


Figure 6.3.: Testing setup in MuJoCo (a) and in the real-world (b) for the straight-line test. We use a rubberized surface mat for the real-world tests.

The independent variables and values used in the simulation and real-world tests are listed in Table 6.1. The normalized stride length is defined as $l_n = l_{st}/l_{st,max}$, where l_{st} is the stride length of the current gait pattern and $l_{st,max}$ is the maximum stride length. Note that the normalized stride length variable is used in a subset of the straight-line tests to isolate the performance of the spine actuation on the gait. We control all other independent variables in the test cases with varying normalized stride length. A complete list of the set of parameters for each test run can be found in Appendix A.4. Additionally, the test cases with gait type lateral sequence walk (lat) do not use the spine.

We use the integrated MuJoCo sensors to collect data during the simulation tests for later evaluation. In contrast, the Nermo-Lite version has no sensors for data collection. Instead, we use three cameras to capture the tests from the front, top, and side view for later evaluation. We use the collected data to calculate the dependent variables listed in Table 6.2. We are limited to measuring the gait speed in the real world tests as the video data is too inaccurate for extracting numerical orientation data. Instead, the video data is used for qualitative orientation evaluation.

Table 6.1.: List of independent variables for the straight-line tests.

Independent Variable	Simulation	Real-World
Gait Frequency [Hz]	0.5, 0.67, 0.8	0.5, 0.67, 0.8
Gait Type	ideal trot (trt), walking trot (trt1), lateral sequence walk (lat)	ideal trot (trt), lateral sequence walk (lat)
Spine Actuation	no spine actuation (ns), spine actuation (s)	no spine actuation (ns), spine actuation (s)
Normalized Stride Length	0.0, 0.5, 0.8, 1.0	0.8

Table 6.2.: List of dependent variables for the straight-line tests.

Dependent Variable	Reason
Speed	Crucial for gait performance and agility
Yaw, Roll And Pitch Noise	Crucial for vision and gait stability

6.1.4. Turning Test Setup

The turning test is designed to test the turning speed and minimum turning radius with and without the spine. We consider the minimum turning radius a proxy for Nermo's agility. Figure 6.4 illustrates the turning test setup with the different path radii. The real-world tests only have a turning radius of 0.5 m due to space constraints. The methodology for the turning test is similar to the straight-line test:

- Nermo is placed at the starting point as shown in Figure 6.4.
- At test begin, the high-level controller sends a constant forward velocity command to the motion module while also sending a turn-rate control signal to follow the desired curved path.
- The test is completed when Nermo reaches the designated endpoint as shown in Figure 6.4.
- Steps 2-3 are repeated three times for each parameter set.
- Steps 2-4 are repeated for each new parameter set.

The independent variables and chosen values for the turn tests are listed in Table 6.3. We also use the sensors in MuJoCo and a camera set up to record data during the experiments. The dependent variables are identical to the straight-line test variables listed in Table 6.1. However, for the turn test, we evaluate the dependent variable speed as the time taken for Nermo to follow a predefined turning radius trajectory.

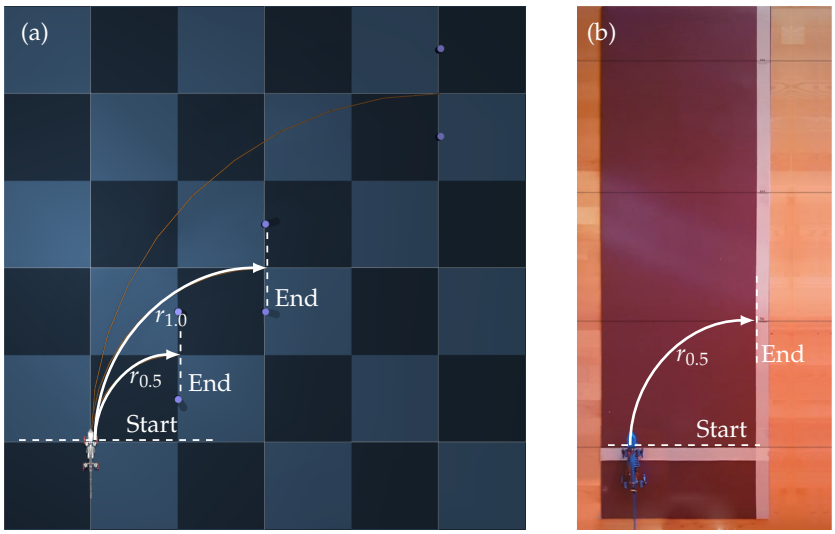


Figure 6.4.: Testing setup in MuJoCo (a) and in the real-world (b) for the straight-line test. The turning radii in simulation are 1.0 m, 0.5 m, and full turning lock. The turning radius in the real-world tests is set to 0.5 m (space constraints).

Additionally, we compute the gait turning radius in each test. This value is trivial if the desired trajectory is followed, i.e., the minimum turning radius is lower than the trajectory radius. Therefore, we also conduct a turning test with a 0.0 m trajectory to evaluate the minimum achievable turning radius. The minimum turning radius is an indicator for the maximum agility for a given turning strategy, gait, and gait frequency type. It should be noted that in the 0.0 m trajectory tests, the high-level controller continuously sends the maximum turn-rate signal.

Table 6.3.: List of independent variables for the turn tests. The 0.0 m turn tests evaluate the minimum turning radius.

Independent Variable	Simulation	Real-World
Gait Frequency [Hz]	0.5, 0.67, 0.8	0.67
Gait Type	ideal trot (trt), walking trot (trt1), lateral sequence walk (lat)	ideal trot (trt)
Turn Type	leg-based , spine-based , mix-based	leg-based , spine-based, mix-based
Trajectory Radius [m]	0.0, 0.5, 1.0	0.5

6.1.5. Balance Test Setup

We designed the balance test to test the capabilities of the spine balance compensation controller. In each test instance, one of the four legs is lifted to evaluate the static stability (with and without balance compensation). The test methodology is as follows:

1. Nermo is brought into a neutral four-foot static contact configuration.
2. Then, each leg will be lifted individually – transitioning Nermo into a three-foot static contact configuration.
3. Next, the lifted leg is compressed fully (via the knee servos) and swung through the full range of motion of the hip/shoulder servo.
4. Steps 2-3 are repeated for each leg (front left, front right, rear left, and rear right) with the spine compensation enabled and disabled.

We record Nermo's ability to successfully remain upright for each test, i.e., not tilt over its dynamic stability axis. In the simulation tests, we also record the contact force on each foot throughout the trials. We repeat the same balance test scenarios with Nermo-Lite. A complete list of the set of parameters for each test run can be found in Appendix A.4.

6.1.6. Maze Scenario Test Setup

The maze test evaluates the gait performance in a scenario that combines straight-line and turning movements. The maze-like environment and test trajectory are shown in Figure 6.5. The test setup is once again similar to the straight-line and turning tests.

1. Nermo is placed at the starting point shown in Figure 6.5 with the spine functions disabled.
2. At test begin, the high-level controller sends a constant forward velocity command to the motion module. Turning motion is handled by the high-level controller designed to follow a predefined path through the maze.
3. The test is completed when Nermo reaches the designated endpoint shown in Figure 6.5.
4. Steps 2-3 are repeated three times for the current configuration.
5. Steps 2-4 are repeated for the configuration where all spine functions are enabled.

During the maze test, Nermo is in two configurations: the spine controller enabled and the spine controller disabled. With the spine controller enabled, Nermo can perform mix-based turning (leg-based and spine-based turning) and spine-based stride extension. Without the spine controller, Nermo performs regular locomotion and only leg-based turning. A complete list of the set of parameters for each test run can be found in Appendix A.4.

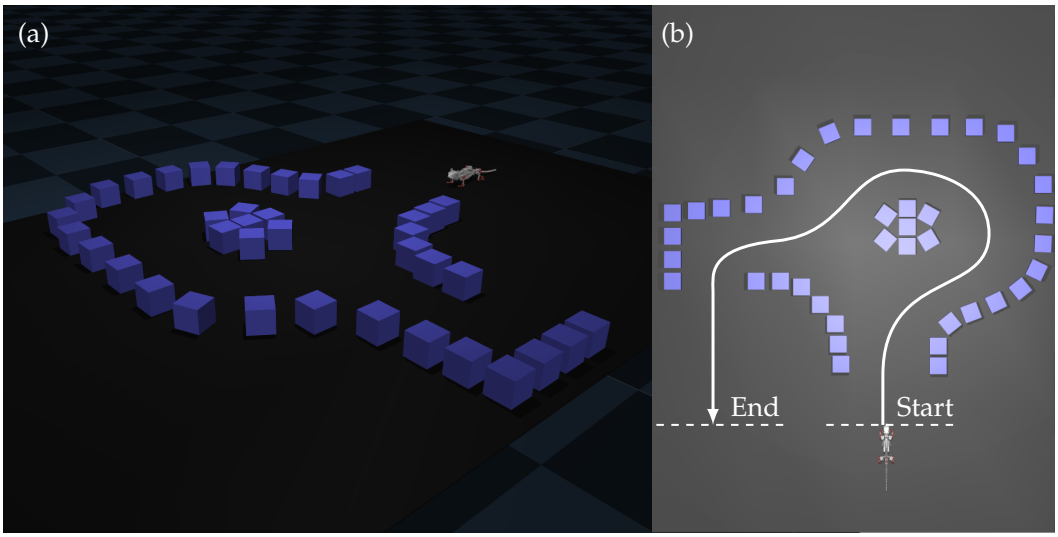


Figure 6.5.: Testing setup for the maze test in MuJoCo. (a) 3D view of maze test. (b) Top view of maze test with start and end points.

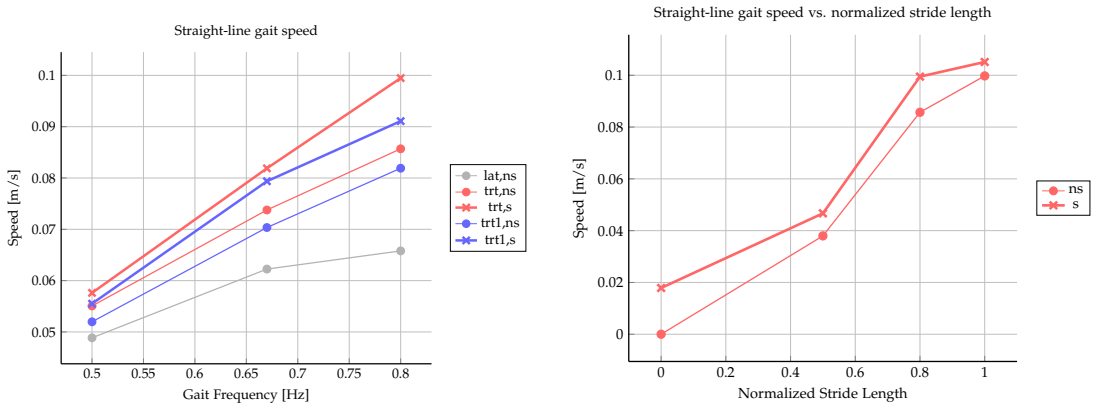
6.2. Test Results

In the previous section, we defined the different test scenarios and independent and dependent test variables. In addition, we covered the general testing methodology for each test scenario. We now cover the test results for each experiment scenario. A complete list of the main gait parameters used for the simulation and real-world tests can be found in the Appendix A.4.1. For information regarding the used gait types, reference Appendix A.1.4.

6.2.1. Straight-Line Test Results

The straight-line test shows a significant relationship between the use of the spine-based stride extension and the gait speed (see Figure 6.6. Depending on the gait frequency, the speed increase ranges between 5 % (for 0.5 Hz ideal trot) to 22 % (for 0.8 Hz ideal trot). For a gait frequency of 0.8 Hz and no contact slip, the theoretical maximum trot gait speed with no spine actuation is 0.096 m/s, and with spine-based stride extension is 0.12 m/s (due to the additional 0.015 m gained per stride; see (5.21)). The theoretical difference in speed is 25 %. The experimental results for the ideal trot at 0.8 Hz closely match the theoretical maximum performance gain. This outcome is expected as the spine-flexion increases the effective stride length per gait cycle. However, we also observe that the experimental results of the speed are approximately 10 % lower than the theoretical maximums.

Furthermore, an increase of the stride length for a fixed gait frequency leads to an increase of the straight-line speed (Figure 6.6(b)). We expect this result given that the gait speed linearly increases with increasing stride length (see (5.21)). We also observe



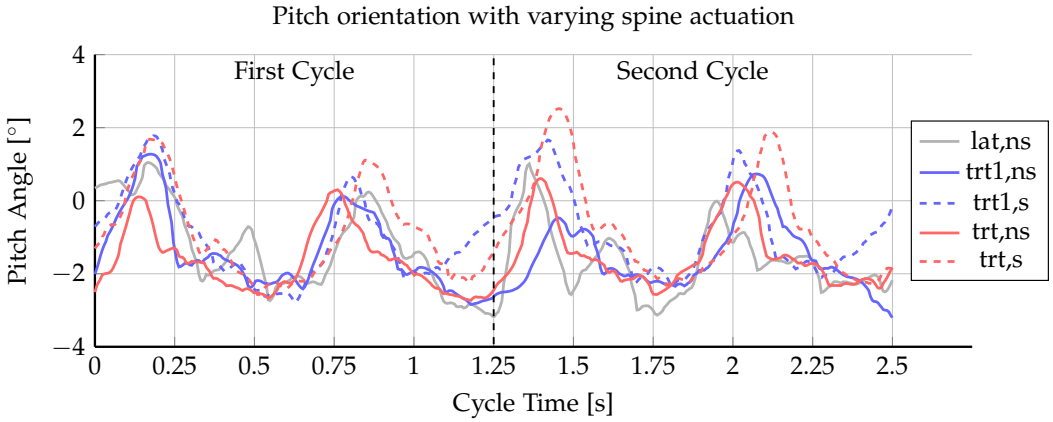
(a) (b)

Figure 6.6.: (a) Straight-line speed dependent on gait frequency, gait type, and spine actuation. The gait types are lateral sequence walk (lat), walking trot (trt1), and ideal trot (trt). (b) Straight-line speed dependent on the normalized stride length for ideal trot at 0.8 Hz. The label (ns) denotes without spine-based stride extension, and (s) denotes with spine-based stride extension.

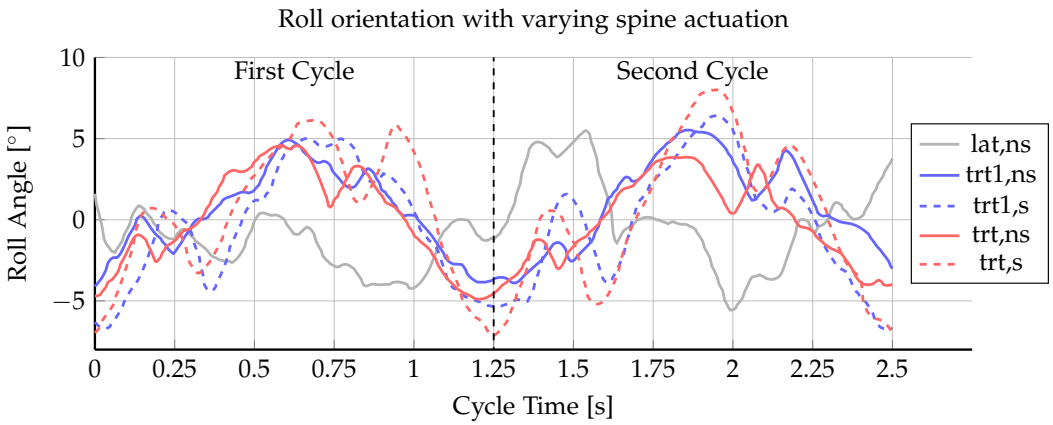
that the ideal trot (duty factor 0.5) is the fastest gait type across all frequencies, followed by the walking trot (duty factor 0.6) and last, the lateral sequence walk (duty factor 0.68; see Figure 6.6(a)). At the highest testing frequency of 0.8 Hz, the ideal trot without gait extension is 27 % faster and with gait extension 48 % faster than the lateral sequence walk. These results also match the gait characteristics of rodents in nature, as the gait speed increases for gait types with lower duty factors. This result is also expected, given that a decreasing duty factor increases the total stride length across a complete gait cycle (also refer to (5.21)).

The results in Figure 6.6(a) also indicate that the gait speed increases with increasing gait frequency, similar to the characteristics seen in biological rodent gaits. For the ideal trot (trt) and walking trot (trt1), the increase is linear within the 0.5 Hz to 0.8 Hz region. For the lateral sequence walk (lat), we observe a gait speed saturation from 0.5 Hz to 0.8 Hz (see Figure 6.6(a)). These results are also expected, as the forward movement speed s_m for the no-slip contact case increases linearly with increasing gait frequency (refer to (5.21)). We attribute the speed saturation for the lateral sequence walk to contact interference in the three-foot contact states at higher frequencies.

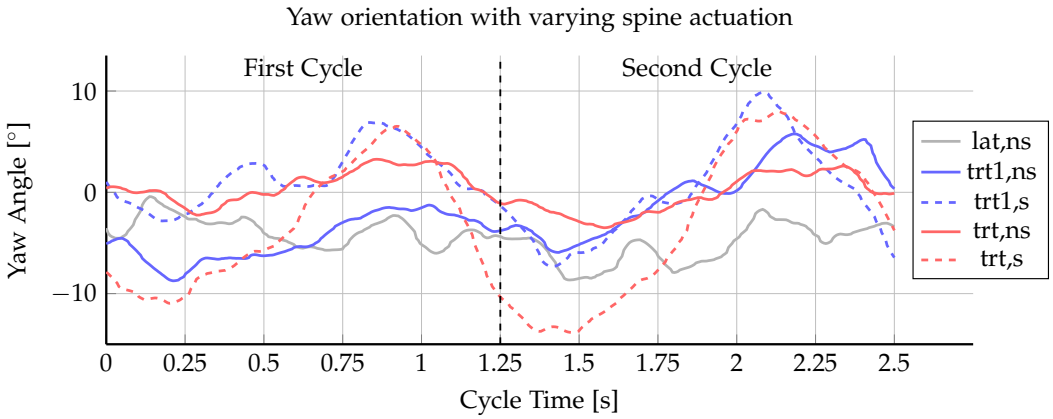
The straight-line tests show the influence of the normalized stride length on the gait performance, as was previously noted. The gait speed increases linearly with increasing stride length for ideal trot (trt) at 0.8 Hz (see Figure 6.6(b)). Furthermore, for a normalized stride length of 0 and actuating spine (s), the locomotion speed is 0.019 m/s. This means that for the ideal trot at 0.8 Hz, the spine-based stride extension provides a 0.019 m/s gait speed increase. This isolated test case highlights the effectiveness of the spine-based stride extension. In comparison, the experimental result is 20 % less than



(a) Pitch orientation behavior in simulation during the two gait cycles.



(b) Roll orientation behavior in simulation during the two gait cycles.



(c) Yaw orientation behavior in simulation during the two gait cycles.

Figure 6.7.: Time-series results in simulation of the orientation during two gait cycles at 0.8 Hz for the straight-line tests.

the theoretical maximum speed of 0.024 m/s for the assumption that the spine-based stride extension increases the stride length by 0.015 m.

Figure 6.7 shows the effect of gait extension, gait frequency, stride length, and gait type on the orientation behavior of the digital twin during two gait cycles. We observe that the spine-based stride extension increases the pitch angle during the gait cycle (Figure 6.7(a)). We also observe that the average pitch is -0.5°. The negative pitch indicates that the digital twin naturally pitches forward with the current gait parameters. However, we also observe that during the gait cycle, particularly with the spine active, Nermo experiences excessive upward pitch. In absolute values, the standard deviation in the pitch behavior across a gait cycle is minimally higher in the gaits with active spine flexion and is always less than 2°. The roll and yaw orientation amplitudes and standard deviation increase with active spine flexion. In particular, we find that the gaits with active spine flexion (s) exhibit 50 % greater yaw orientation amplitudes and 186 % greater yaw orientation standard deviation (refer to Figure 6.7(c)). Therefore, the yaw orientation is most sensitive to the spine-based functionalities.

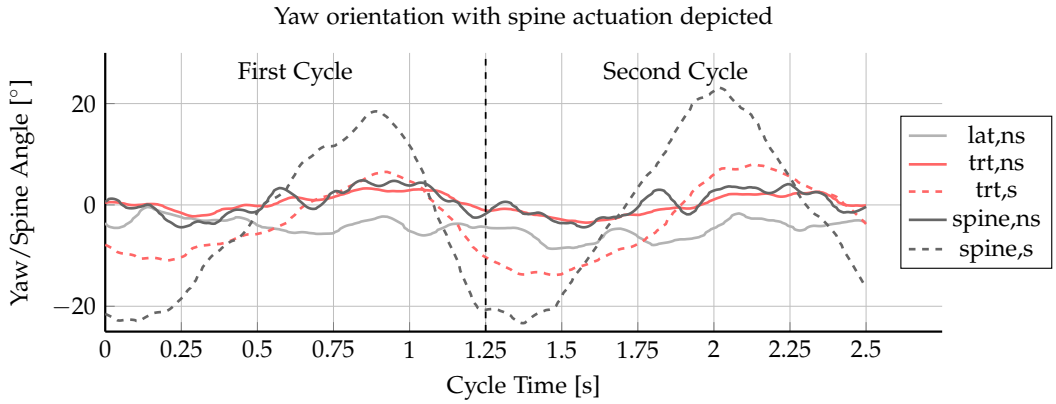
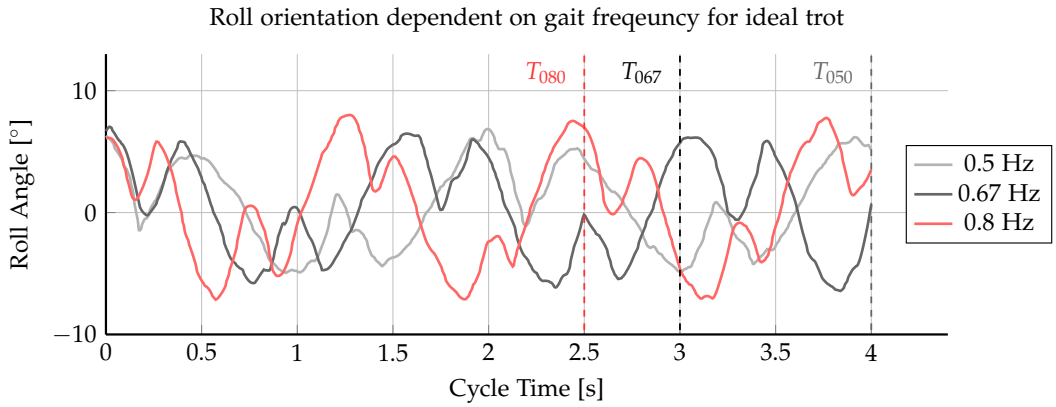


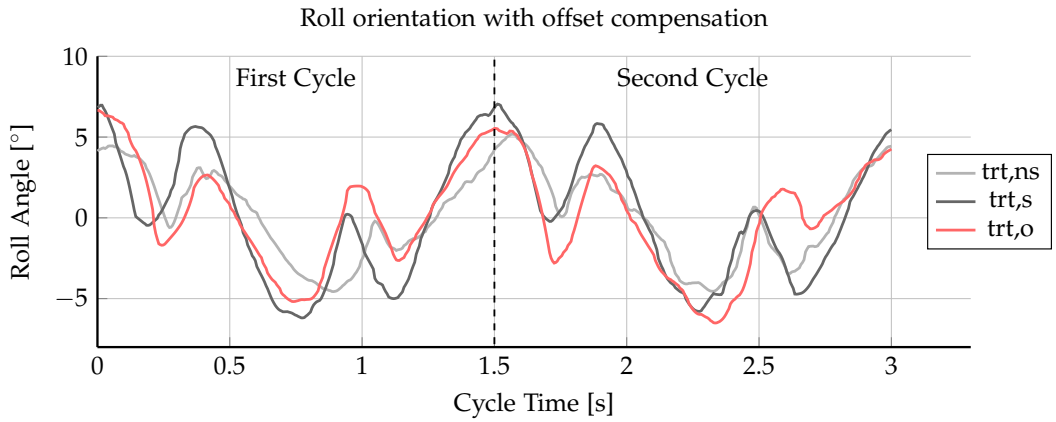
Figure 6.8.: Time-series results from simulation of the yaw orientation compared with the spine actuation angle during two gait cycles at 0.8 Hz. The vertical dashed line indicates the start of the second gait cycle.

Furthermore, the yaw deflection is periodic and most significantly affected in gait types with lower duty factors (see Figure 6.7(c)). We believe this result has one significant implication; multi-foot contact phases influence the spine yaw flexion during the gait cycle. For example, initiating spine flexion in a three-foot contact scenario means that the additional foot contact that is not part of the contact diagonal induces a contact friction force that opposes the spine flexion. This phenomenon is even worse for four-foot contact scenarios. The yaw orientation behavior is also periodic for gaits with active spine flexion during the gait cycle. We see that the yaw behavior does indeed correlate with the spine flexion (observe Figure 6.8). This correlation is expected as the spine-flexion actuates along with the yaw orientation. As an aside, the periodic orientation behavior can also be observed for the pitch and roll orientation.

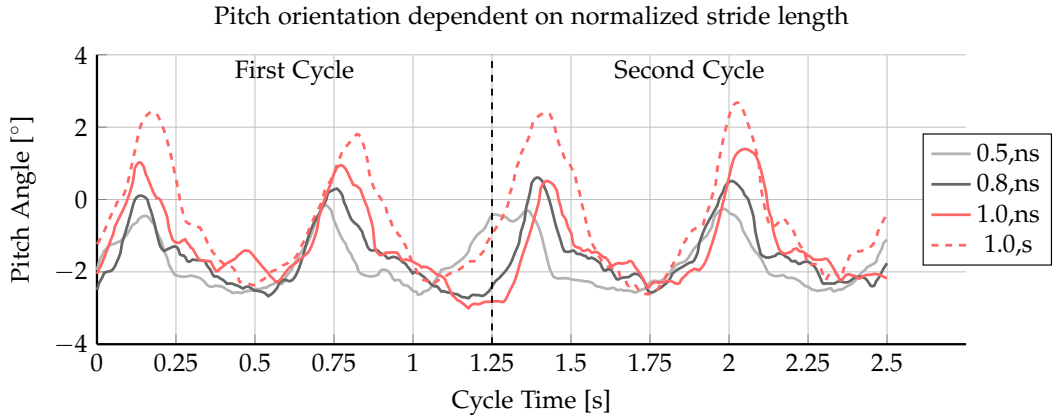
Critically, the standard deviation of the roll orientation increases by 25 % to 40 % with



(a) Roll behavior with varying gait frequency for ideal trot. The vertical lines indicate the end of the second gait cycle for each frequency.



(b) Roll behavior with offset compensation enabled for ideal trot at 0.67 Hz.



(c) Pitch behavior with varying normalized stride length for ideal trot at 0.8 Hz.

Figure 6.9.: Time-series results from simulation of the roll and pitch orientation behavior.

the spine-based stride extension enabled (see Figure 6.7(b)), and the maximum amplitude range is between -8° to 8° for the ideal trot. Furthermore, gait types with lower duty factors lead to higher roll behavior (Figure 6.7(b)). We believe the increased roll behavior occurs because in gaits with lower duty factors, more time is spent in two-foot contact leading to roll across the dynamic stability line. The roll orientation behavior is especially critical as it cannot be compensated by the head and neck actuators. Furthermore, the roll noise successively increases by 20 % with increasing gait frequency (Figure 6.9(a)). We also observe that the offset compensation reduces the roll amplitude and standard deviation by 12.5 % and 15 %, respectively (Figure 6.9(b)). As the offset compensation only compensates against roll rotation of the hip, we observed no meaningful impact on the yaw and pitch orientation (refer to Figure A.9 in the Appendix). The standard deviation in the pitch, roll, and yaw orientation is smallest in gaits without spine actuation (ns).

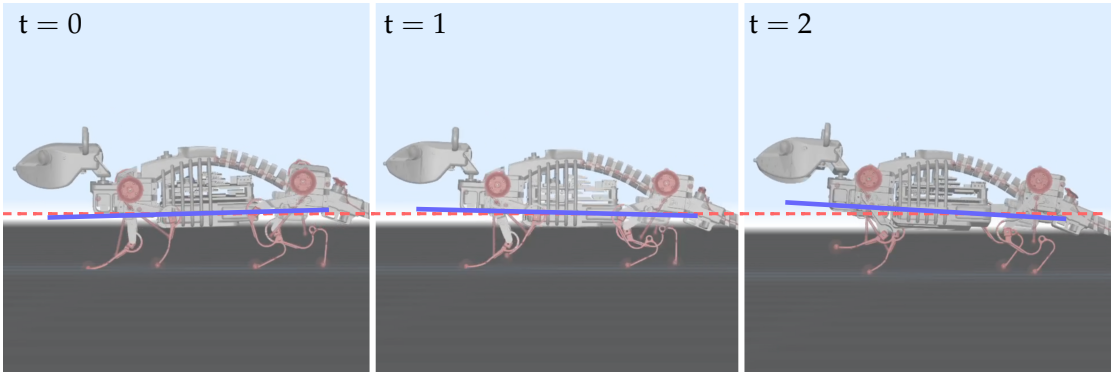


Figure 6.10.: Depiction of upward pitch during rear leg lift-off phase during a gait cycle from simulation (time step from left to right).

From Figure 6.9(c), we find that the normalized stride length influences the pitch orientation noise. We observe that with increasing normalized stride length from 0.8 to 1.0, the standard deviation and the maximum amplitude of the pitch orientation behavior increases by up to 50 %. Figure 6.10 shows the pivot action around the rear legs during lift-off, which induces the upward pitch orientation during the gait cycle. As each gait cycle has two lift phases, we also observe two peaks. By extending the normalized stride length, the opposite rear leg (not in lift-phase) is further forward, increasing the rearward pivot point of the rear stance leg. The increased pivot point results in a more excellent upward pitch during the beginning of the lift-off phase of each half leg cycle. The upward pitch behavior is further increased by 100 % with active spine flexion for a normalized stride length of 1.0 (see Figure 6.9(c)).

In Figure 6.11 we observe that the use of the spine-based stride extension increases the gait speed of Nermo-Lite in the real-world tests. The real-world test results validate the outcomes and trends of the simulation tests. Specifically, the spine-based stride extension increases the real-world speed by 15 % to 22 % depending on the gait frequency. For

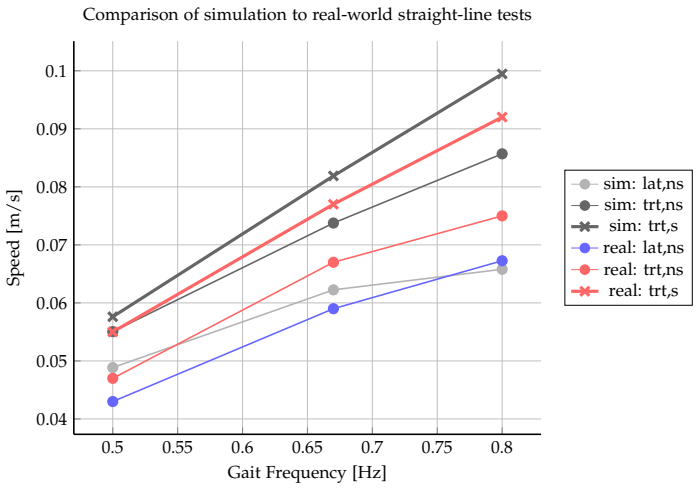


Figure 6.11.: Comparison of straight-line speeds between simulation and real-world experiments.

example, for the ideal trot at 0.6 Hz the speed increases from 0.067 m/s to 0.077 m/s (increase of 15 %). Similar to the simulation results, we also observe that the ideal trot (trt,ns) is on average 11 % faster than the lateral sequence walk (lat). However, we also observe a speed discrepancy between the real-world and simulation tests. The speed of Nermo-Lite is on average 5 % slower in lateral sequence walk (lat), 12 % slower in ideal trot without gait extension (trt,ns), and 6 % slower in ideal trot with gait extension (trt,s).

6.2.2. Turn Test Results

The turning test results clearly demonstrate that spine-based lateral flexion turning is the fastest across all gait frequencies and gait types irrespective of the turning trajectory (see Figure 6.12). The spine-based turning strategy achieves 15 % to 30 % faster traversal speed than leg-based turning. Similar to the straight-line test results, increasing the gait frequency leads to increased turning traversal speed. The mix-based (mb) turning strategy achieves faster traversal speeds than leg-based turning (lb) (but slower than spine-based (sb) turning. However, for the ideal trot (trt) and walking trot (trt1) cases at 1.0 m turning radius and 0.5 Hz gait frequency, the difference between the spine and mix-based turning is negligible at 1 % and 3 %, respectively. Therefore, the mix-based turning is predominately spine-driven at larger turning radii.

Although spine-based (sb) turning increases the turning speed compared with the other turning strategies, its most significant limitation is the minimum achievable turning radius compared with the other turning strategies. The digital twin achieves a turning radius of 0.96 m (see Figure 6.13) for spine-based turning with lateral sequence walk (lat) at 0.8 Hz, which is greater than the 0.5 m trajectory and results in a failed test (denoted by *TestFailed* in Figure 6.12)). Spine-based turning with the lateral sequence

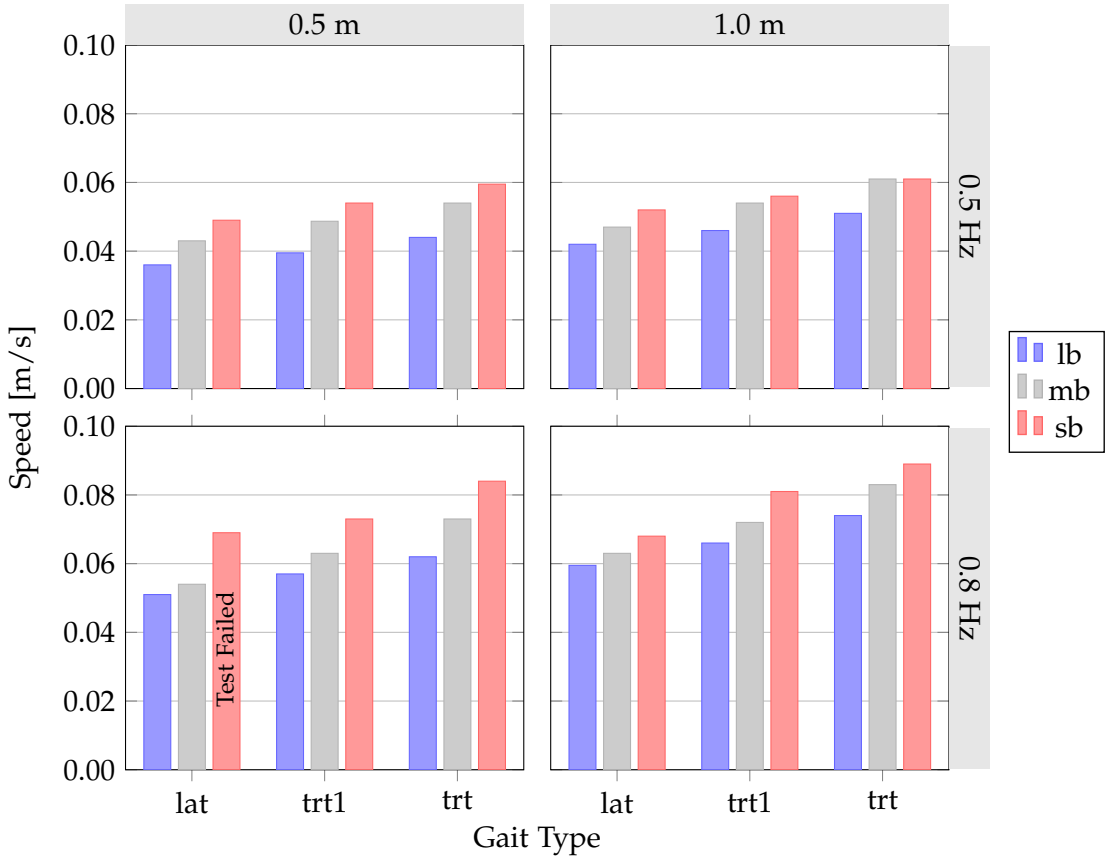


Figure 6.12.: Turning speed performance with different gait and turning strategies across 0.5 m and 1.0 m radius trajectory. Note that for the test with 0.5 m radius and 0.8 Hz gait frequency, the digital twin failed to follow the specified radius for the gait type lateral sequence walk (lat) with spine-based (sb) turning - denoted by the label "Test Failed". Results from simulation.

walk gait is especially sensitive to changes in the gait frequency; at a gait frequency of 0.5 Hz the minimum turning radius is 0.49 m, while at a gait frequency of 0.8 Hz the turning radius is 0.96 m (increase of 97 %). This result is significant because spine-based turning is the only strategy to fail a turning test.

The results show that the mix-based turning strategy compensates for the limited turning radius of spine-based turning while still achieving a faster turning performance than leg-based turning. With mix-based turning, the minimum turning radius achieved by Nermo in the simulation is 0.12 m (0.6 body-length). The mix-based (mb) turning mode achieves the best minimum turning radius; 25 % to 30% lower than the minimum turning radius achieved by leg-based turning (lb). Furthermore, the minimum turning radius of the mix-based (mb) turning is 62.5 % (ideal trot at 0.5 Hz) to 90 % (lateral sequence walk at 0.8 Hz) smaller than for spine-based (sb) turning. Therefore, mix-based turning combines the speed benefit of spine-based turning with the turning radius benefit of leg-based turning.

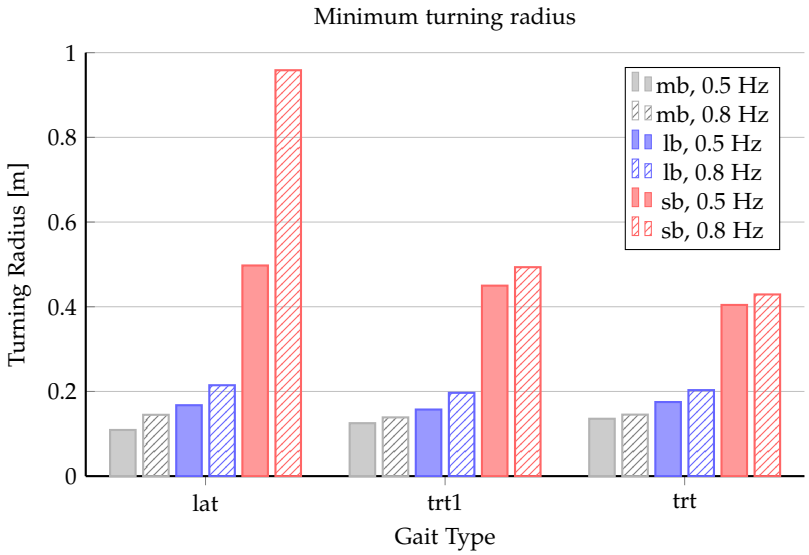


Figure 6.13.: Minimum turning radius achieved by different gait types, gait frequency, and turning strategy modes. The turning strategies tested are mix-based turning (mb), leg-based turning (lb), and spine-based (sb) turning. Results from simulation

For the ideal trot under no-slip contact conditions, we expected a minimum turning radius of 0.151 m for leg-based turning (lb) and 0.134 m for mix-based turning (mb), constituting an improved turning radius of 10 %. A smaller turning radius for mix-based turning is expected, as the additional spine flexion increases the yaw moment induced by each leg by shortening the inner leg diagonal. The minimum achieved turning radius for leg-based (trt,lb) was 0.174 m, and for mix-based turning (trt,mb) 0.135 m, an improvement of 22 %. Therefore, the mix-based (mb) turning values matched the theoretical predictions, while the leg-based turning (lb) performance diverged by 15 %.

Therefore, the results follow the expected theoretical performance trends, albeit with a deviation in the absolute performance values.

The real-world test results confirm that spine-based and mix-based turning improve the turning speed and radius compared with leg-based turning, as seen in simulation. The spine-based turning strategy enables the fastest turning speed while leg-based turning performs the slowest (30 % reduction in time, see Table 6.4). However, contrary to the simulation results, the spine-based (sb) turning fails to follow the 0.5 m trajectory. There is a 12 % to 19 % difference in results between simulation and real-world tests (see Figure 6.4). It can be concluded that the trends from the simulation for the turning behavior are transferable to the real world but at a loss of expected performance.

Table 6.4.: Numerical results of the real world turning tests for a 0.5 m radius trajectory. The gait type used was ideal trot (trt) at 0.67 Hz gait frequency with all three turning strategies tested. The table also notes whether a trajectory was successfully followed (pass or fail).

Turn Type	Simulation		Real-World		Difference
	Time [s]	Status	Time [s]	Status	
Spine-based turning (sb)	13.0	Pass	15.5	Fail	19 %
Leg-based turning (lb)	18.4	Pass	22.0	Pass	19 %
Mix-based turning (mb)	14.5	Pass	16.3	Pass	12 %

6.2.3. Balance Test Results

The balance test results demonstrate the ability of the spine balance compensation (SBC) to ensure static stability even in initially unstable configurations (see Table 6.5). As expected, the digital twin passed the front leg lift tests without and with SBC enabled as the COM position is within the stability triangles (see Figure 6.14(a)). In contrast, the COM lies outside the stability triangle for the rear leg lift tests. For example, lifting the rear left leg without spine balance compensation leads to the digital twin pivoting over the balance diagonal and reaching the resting state depicted in Figure 6.14(b). The rear left foot touches the ground in this resting state, and the right front foot is in the air, resulting in a failed balance test. Therefore, the digital twin is initially unstable for balance configurations of the rear legs.

The initially unstable position is resolved by shifting the COM into the stability triangle and changing the foot contact load distribution. The change in contact load distribution is confirmed by the test results when the spine balance compensation is pre-enabled in the four-foot stance, leading to increased contact forces at the balance diagonal (front left and rear right foot) and a decrease in contact force at the rear left foot (see Figure 6.14(c)). The change in contact force is confirmed by the numerical results from simulation depicted in Figure 6.16. Specifically, the contact load on the rear

Table 6.5.: Results of the four spine balance test cases.

Foot Lifted	Simulation		Real-World	
	No SBC	SBC	No SBC	SBC
Front Left	Pass	Pass	Fail	Pass
Front Right	Pass	Pass	Fail	Pass
Rear Left	Fail	Pass	Fail	Pass
Rear Right	Fail	Pass	Fail	Pass

left leg reduces from 0.73 N to 0.16 N while the contact load on the front right support foot is at 0.35 N (see Figure 6.16). With the spine balance compensation enabled, the rear left foot is lifted without Nermo falling over (see transition from Figure 6.14(c) to Figure 6.14(d)). In this configuration, the majority of the contact load is on the diagonal contact pair (front left and rear right; see Figure 6.16). Therefore, we confirm that the COM translation by the spine compensation changes the contact force distribution at the feet.

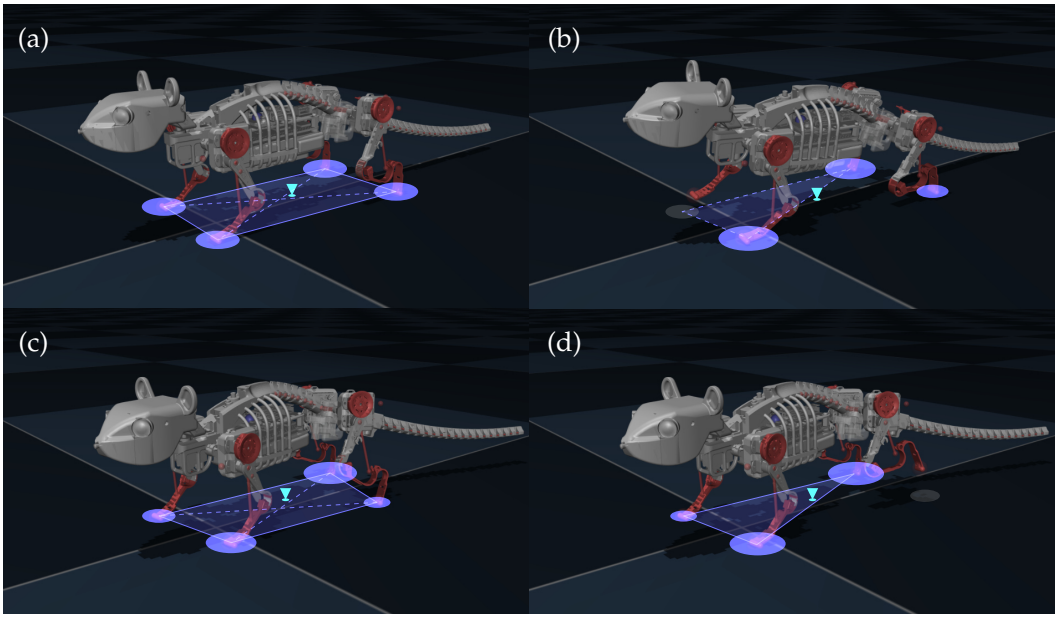
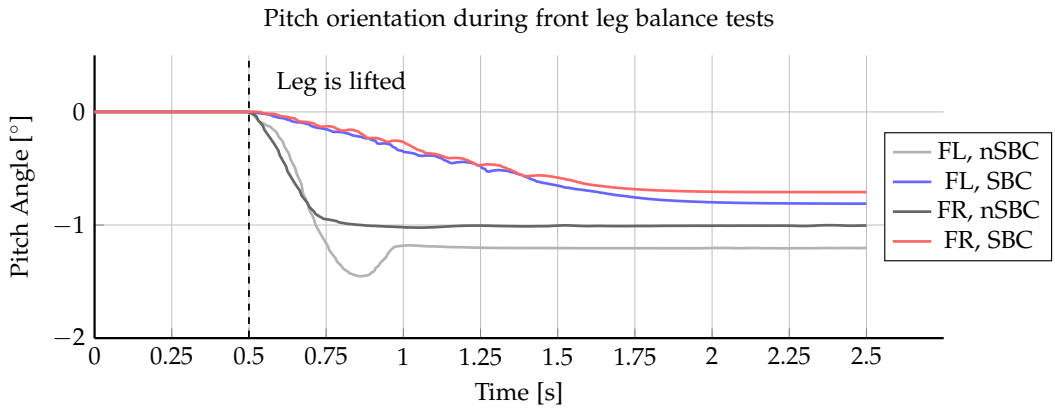
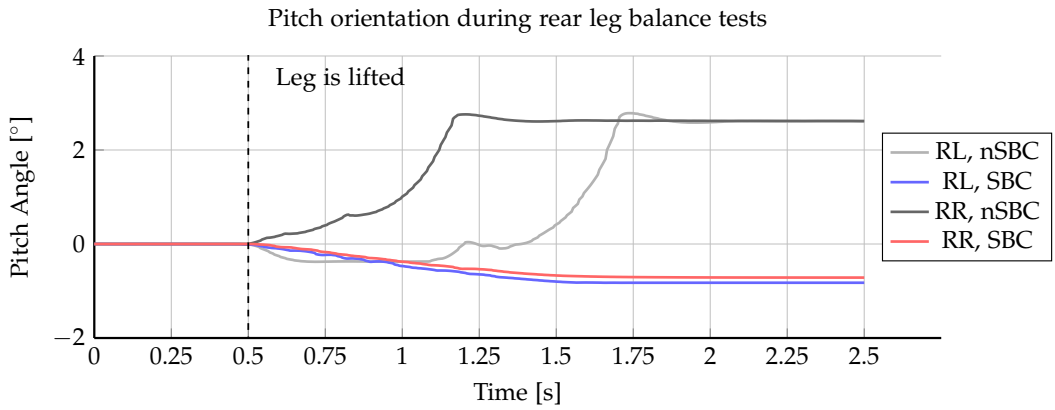


Figure 6.14.: Rear left leg balance test. The blue ellipses qualitatively depict the contact feet and forces (larger ellipse for larger load). The triangle icon depicts the projection of the COM onto the ground. (a) Four-foot contact configuration without spine balance compensation (SBC). (b) Static three-foot contact configuration without SBC. The digital twin attempts to lift its rear left foot, but fails to maintain balance and falls onto the rear left leg (test failed). (c) Static four-foot contact with SBC and (d) static three-foot contact with SBC (test passed).



(a) Pitch behavior of front leg balance tests



(b) Pitch behavior of rear leg balance tests

Figure 6.15.: Time-series results from simulation of the pitch orientation during the balance tests. The legend indicates which leg is lifted - front left (FR), front right (FR), rear left (RL), rear right (RR) - and whether spine balance compensation is enabled (SBC) or disabled (nSBC).

Additionally, the simulated balance tests show that the spine balance compensation improves the pitch orientation stability during three-foot static contact. In Figure 6.15, we can observe the pitch orientation behavior during the balance tests. Initially, the pitch orientation is stable until the leg is lifted. In the front leg balance test results (see Figure 6.15(a)), we observe that SBC reduces the forward pitch resting orientation by 30 %. In the rear leg balance test results (see Figure 6.15(b)), the SBC reduces the absolute pitch deviation by 73 %. Figure 6.15(b) also shows the ground impact in the failed rear leg tests (without SBC (nSBC)) at times 1.2 s (RR, nSBC) and 1.75 (RL, nSBC). Therefore, SBC enables static stability for statically unstable scenarios and improves the resting pitch orientation.

Figure 6.16.: Foot contact forces for the rear left balance test. From left to right the test case categories of the x-axis correspond to (a), (c), (b), (d) in Figure 6.14. The number 4 corresponds to four-foot contact, 3 to three-foot contact, and SBC if the spine balance compensation is enabled.

Contrary to the simulation results, Nermo-Lite did not pass the front leg balance tests (with SBC disabled) due to additional spine and front leg compliance. The additional compliance leads to forward leg compression that shifts the COM outside the stability triangle, which leads to Nermo falling over. This means that Nermo-Lite experiences much more significant front leg compression than the digital twin, even with SBC enabled (refer to Figure 6.17). The real-world results for the rear leg tests match the results from simulation. In Figure 6.17 we also observe that the stiffer rear leg joint results in decreased leg compression in the three-foot contact case compared with the front leg tests (with SBC enabled). Overall, Nermo-Lite passed all balance tests with SBC enabled. It can be concluded that the spine balance compensation significantly improves the three-foot static contact stability of both the digital twin and Nermo-Lite.

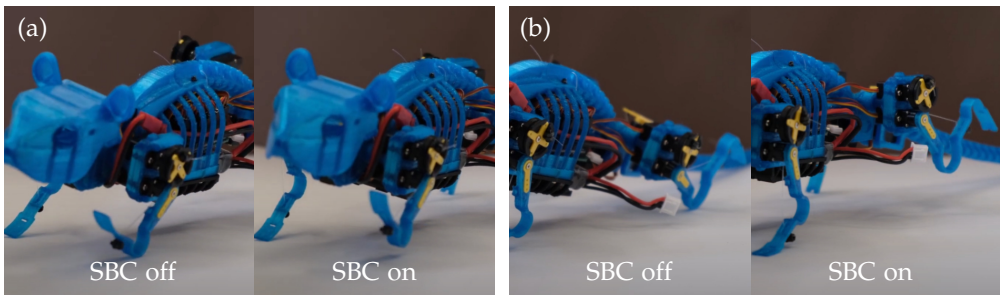


Figure 6.17.: (a) Front left real world balance test. (b) Rear left real world balance test. Without SBC, Nermo-Lite falls over.

6.2.4. Maze Scenario Test Results

The most profound influence of the spine functionalities on the locomotion performance is observed in the maze scenario experiments (see Table 6.6 for simulation results). With spine-based stride extension and mix-based turning, the digital twin completed the maze in 28 % less time and achieved a 37 % faster average speed through the maze. In the real-world tests shown in Figure 6.18, Nermo-Lite took 32 % less time to complete the maze. It should be noted that the real-world maze tests were conducted in a smaller test environment and are therefore not directly comparable with the simulation results. The simulation and real-world results confirm the performance increase of the spine-based functionalities demonstrated in the straight-line and turning tests.

Table 6.6.: Numerical results of the maze test from simulation. The maze test is conducted with all spine functions disabled and all spine functions enabled (gait extension and mix-based turning).

Metric	No Spine	With Spine	Change
Time Taken [s]	85.7	61.1	-28 %
Avg Speed [m/s]	0.060	0.082	37 %
Pitch Standard Deviation [°]	0.75	1.25	67 %
Roll Standard Deviation [°]	3.09	4.13	34 %

The sample trajectories shown in Figure 6.19 help highlight performance differences between the spine enabled and spine disabled gait performance. The depicted trajectories contain oscillations that correspond with the completed gait cycles. In Figure 6.19(b) we observe a high concentration of gait oscillations in the turning sections compared with Figure 6.19(a) - an indicator that the digital twin spends more gait cycles in the turning section without spine flexion than with spine flexion. Therefore, given the fixed gait frequency across both tests, the digital twin requires more time to complete the turning sections when the spine-based functionalities are disabled.

Furthermore, the maze results confirm that the spine-enabled gait exhibits increased orientation and sudden positional deviations compared with the non-spine enabled gait. The enabled spine-based gait functions increase the standard deviation of the roll orientation by 34 % and pitch orientation by 67 % (see numerical results in Table 6.6). To reiterate, the roll noise is most critical because the head actuators cannot compensate for the roll behavior, which negatively impacts the visual perception sensors (i.e., cameras in the head) and tasks. We also observe that the trajectory of the digital twin with the spine enabled locomotion exhibits micro-peaks in the oscillations along the straight-line segments (refer to Figure 6.19(a)). The micro-peaks represent positional noise induced by the abrupt change in COM position during spine-based stride extension.

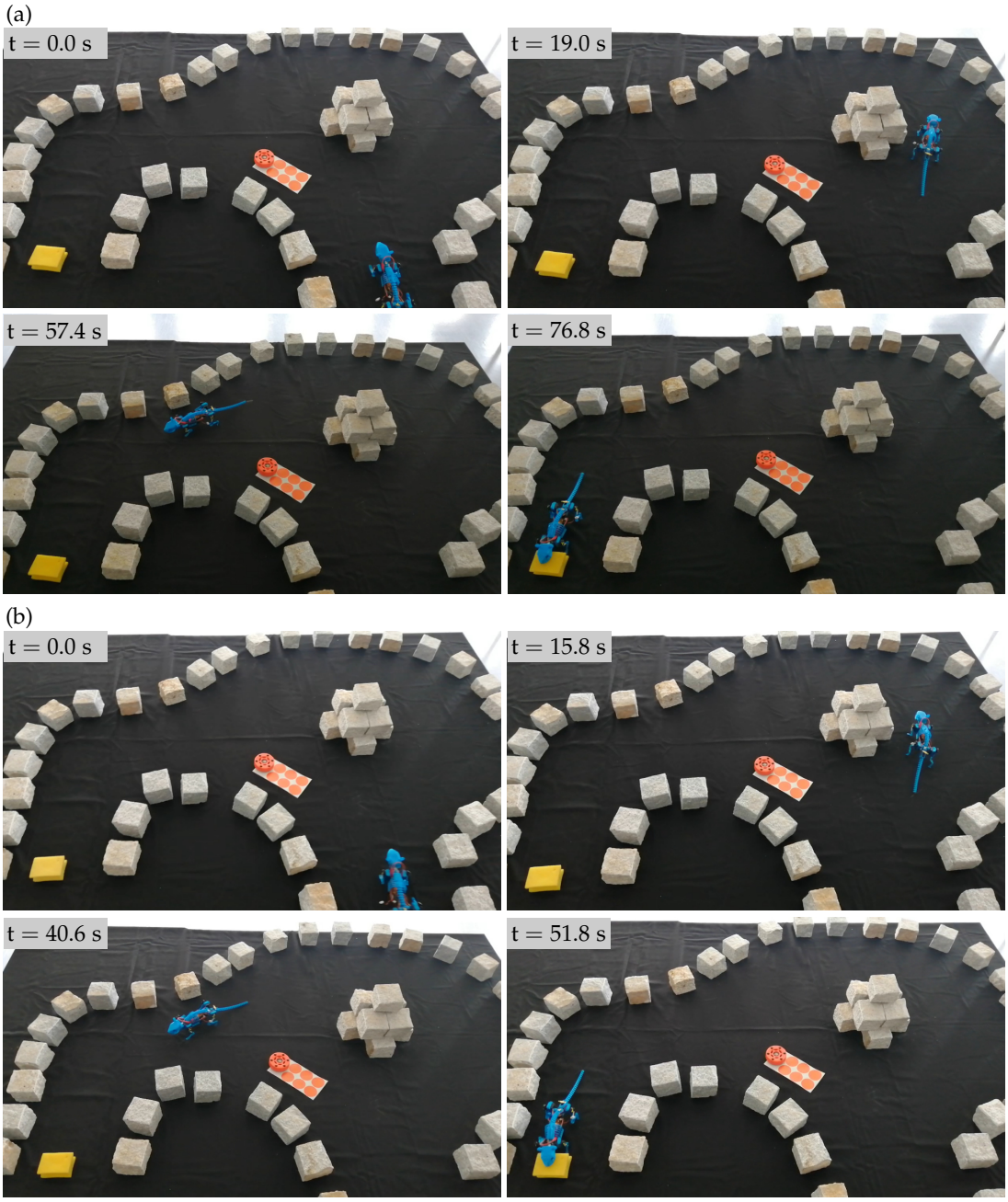


Figure 6.18.: Real world maze test with obstacle (orange dots) and goal (yellow sponge). (a) Nermo-Lite navigating the maze test without spine-based functions. (b) Nermo-Lite navigating the maze test with spine-based functions.

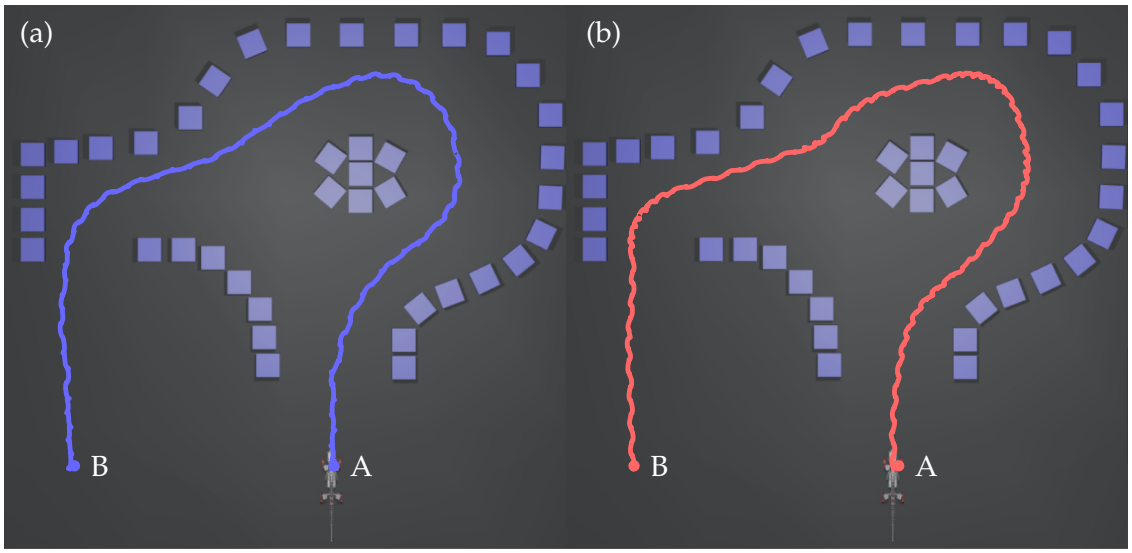


Figure 6.19.: Visual depiction of the trajectories of the digital twin during the maze test. (a) Shows the trajectory of with all spine functions enabled and (b) the trajectory with all spine function disabled.

7. Discussion

In the previous section, we discussed the results of the four test case scenarios focusing on the numerical performance metrics. This chapter discusses the implications of each spine-based function and the advantages and disadvantages of using the spine. We conclude this chapter by discussing the differences between theoretical values, simulation results, and real-world test results.

7.1. Implications of the Spine-Based Functionalities

In this thesis, we successfully demonstrated that a flexible spine can improve the locomotion performance of small underactuated quadrupeds in three key areas: 1) straight-line speed, 2) turning behavior, and 3) static stability. Our work shows that the gait speed can be increased with spine flexion in the frontal plane (akin to the walk of a salamander) in addition to the sagittal plane flexion explored in previous works [38], [39]. We have also shown that an additional spine controller needed to control the spine functionalities can be successfully integrated into existing quadruped control architectures. Therefore, developers of small quadruped robots have an additional design feature, a flexible spine, to improve the performance of their quadrupeds.

7.1.1. Advantages of the Spine-Based Functionalities

Gait speed increase: The simulation and real-world experimental results have successfully demonstrated that an actuated flexible spine can enable faster walking speeds by extending the effective stride length, compared with only leg-based walking. Therefore, the spine-based stride extension provides an additional method, beyond increasing the gait frequency and stride length, to increase the gait speed. The spine-based stride extension achieves this by carefully synchronizing gait compression (decreasing the distance between opposing leg pairs) and extension (increasing the distance between opposing leg pairs) to increase the effective stride length by up to 0.03 m per gait cycle. As a result, for a leg stride length of 0.06 m, the total effective stride length is 0.15 m (see (5.22)) per gait cycle for the ideal trot. Therefore, the spine-based stride extension increases the effective stride length per gait cycle by up to 25 %.

Turning performance: Another significant advantage of the flexible spine is the ability to use it for turning with underactuated quadrupeds. Quadrupeds are limited to leg-based turning without an active spine, which is remarkably inefficient in underactuated

quadrupeds. Leg-based turning is capable of tight turning radii at the cost of turning speed. With the addition of an active spine, we have shown that the turning speed of Nermo can be drastically increased by up to 40 % compared with leg-based turning, depending on the turning radius and gait type. The speed difference occurs because the spine-based turning does not need to modulate the leg stride length and takes advantage of the entire forward leg motion while turning. Furthermore, the minimum turning radius is improved by up to 30 % with the mix-based turning strategy compared with leg-based turning. Two mechanisms enable the improved turning radius of the mix-based turning compared with the leg-based turning. First, by applying frontal spine flexion during leg-based turning, the yaw turning moment of the COM is increased because the spine-flexion shifts the COM closer towards the turning radius. Second, the hip axis is tilted inwards, shortening the leg diagonal and increasing the yaw turn rate per gait cycle. To conclude, the spine significantly improves the turning speed and radius of Nermo.

Balance performance: We have shown that the spine improves the static stability of quadrupeds without the need for foot position readjustment. In all test cases, with the spine-based balance compensation enabled, Nermo remained balanced in three-foot contact scenarios that were initially statically unstable. The spine-based balance compensation enables the balance behavior by actively shifting the COM inside the static stability triangle. In this way, Nermo can adjust the foot positions of individual feet without the risk of falling over.

7.1.2. Disadvantages of the Spine-Based Functionalities

In the straight-line tests, we observed that the pitch, roll, and yaw orientation exhibited increased standard deviations and amplitudes with the enabled spine-based gait extension. The orientation behavior is essential because the visual perception system of Nermo, with the head-mounted cameras, is susceptible to unexpected orientation changes. These unexpected orientation changes introduced visual noise, which makes perception tasks and processing complex.

Pitch orientation: The undesired pitch orientation is primarily caused by undesired rear leg compression during the lift-off phase, as outlined in the results and visualized in Figure 6.10. The rear leg compression leads to a sudden upward pitch during leg lift-off, followed by a downward pitch during the front leg touch-down, which increases the pitch amplitude and standard deviation during a complete gait cycle. The pitch orientation behavior can be reduced by adjusting the gait parameters or increasing the stiffness of the rear leg. Furthermore, the neck of Nermo has a pitch-axis joint that can help mitigate the undesired pitch orientation behavior.

Roll orientation: The roll orientation amplitude and standard deviation are influenced by the asymmetric vertical shift and roll at the hip caused by the lateral flexion of

the spine. Additionally, the sudden shift in COM position during spine-based stride extension introduces dynamic instability at the legs and subsequent roll behavior. We have shown that the roll behavior caused by the spine-based stride extension can be reduced with the spine-offset compensation by modulating the leg lengths to counteract the roll and vertical shift in the hip (as shown in Figure 6.9(b)). The reduction of the roll behavior with spine-offset compensation is a promising result, as the head of Nermo cannot compensate in the roll direction.

Yaw orientation: The yaw orientation behavior is primarily influenced by the actuation amplitude of the spine along the yaw axis. In Figure 6.8 we observed that the yaw orientation amplitudes coincide with the spine flexion amplitudes. The undesired yaw orientation behavior can be reduced by decreasing the lateral spine flexion in scenarios that prioritize visual perception. Furthermore, like for the roll orientation, the yaw orientation can be mitigated by the yaw joint located at the neck of Nermo.

7.2. Implications of the Digital Twin

The use of the digital twin of Nermo has increased the speed at which we were able to design and test the spine-based functions. Furthermore, the simulation allowed for more granular data extraction, including contact forces at the feet during the balance tests. In the discussion of the results in section 6.2 we verified the accuracy of the digital twin to simulate real-world behavior and trends. Therefore, the digital twin can be used in future research to test model-based and learning-based control algorithms rapidly.

Furthermore, the digital twin is a valuable test platform for new mechanical designs. Although not covered in this thesis, we used a fully digital design process to redesign the rear leg of Nermo. The desired performance characteristics of the redesigned leg were validated using the digital twin. The validated leg design was then manufactured and tested on the real-world robot, where we were able to confirm the simulation performance. Therefore, the digital twin is a valuable alternative for quickly developing mechanical designs at a low cost.

Lastly, we observed a difference between the theoretical results, simulation results, and real-world results. Starting with the theoretical to simulation comparison, the simulation results deviated by 10 % in absolute values for the straight-line tests and by 20 % for the turning tests. However, we also proved that the spine-based stride extension speed improvement closely matched the theoretically expected improvement for the straight-line tests. Lastly, we observed that the real-world tests deviated by a further 10 % to 20 % compared with the simulation results. We believe several factors are contributing to the difference in results.

7.2.1. Theoretical to Simulation Difference

In legged locomotion, the feet are periodically in and out of contact with the ground. Consequently, the ground friction and foot contact pairing are crucial parameters for gait performance. For the theoretical results, we always considered the no-slip condition assumption. However, we observed different degrees of contact slip depending on the ground/foot contact pairing from the simulation and real-world tests. In particular, we observed excessive slip behavior of the rear leg during the lift-off phase. We believe two factors contribute to the observed slip characteristics. First, the neutral pitch orientation of the digital twin is forward biased, which results in a higher contact load and consequently contact friction at the front legs than the rear legs. Second, the rear leg moves away from the COM at the end of the stance phase compared with the front legs, which results in a further decrease of the contact force at the rear leg. Both effects decrease the rear leg vertical contact force during lift-off, reducing contact friction and leading to contact slip. Furthermore, during the lift-off phase, the compliant ankle and knee provide a sudden potential energy release stored during the stance phase, ideally requiring increased horizontal friction force. Therefore, the digital twin loses considerable speed due to contact slip when compared with the theoretical values.

However, we hypothesize that the pitch behavior of the digital twin also negatively influences gait performance. During the gait cycle, the pitch orientation fluctuates up to 4° upwards during lift-off and downwards during touch-down (refer to Figure 6.7(b)). This pitch behavior leads to a decrease in the forward stride length because: 1) the upward pitch during lift-off adds additional leg compression and delays the lift-off phase, and 2) the forward pitch results in premature leg touch-down. Both factors combined result in a decreased effective stride length. Observations of the gait behavior from experimental videos further confirms that contact slip and reduced stride length are possible reasons for the differences in the turning performance.

7.2.2. Simulation to Reality Difference

Contact slip is the primary reason for the difference between the theoretical and simulation results. In part, the influence of contact slip also translates to the real-world tests. Due to unforeseen environmental factors, we cannot guarantee that the ground/foot friction behavior matches the friction conditions of the simulation and vice versa. However, comparing the video results of the real world and simulation tests, we observed that the contact behavior appeared to be very similar. Therefore, we believe the simulation to real difference is not primarily caused by additional slip.

Instead, we observed that the spine in Nermo-Lite is considerably more compliant and constrained in its range of motion than the spine of the digital twin. Two factors cause the additional compliance of the spine and range of motion constraints: 1) the manufacturing tolerances leading to uneven pretension of the spine tendons, and 2) the interference from servo wires and additional cable components running from the main body to the back of Nermo-Lite. These two factors change the response behavior,

increase spine compliance, and limit the maximum spine deflection angle. We also believe that the additional compliance influenced the increased pitch behavior that affected the total stride length. The interference of the servo wires results in a decreased maximum spine deflection, which helps explain the reduced turning performance of Nermo-Lite for spine-based turning.

One other point to mention is the assembly of the Nermo-Lite. The models and performance of Nermo rely on the assumption that there is no loss of pretension in the leg tendons to provide an instantaneous response when actuating the servos. We cannot ensure proper tendon pretension due to the robot's stringing tolerances, part tolerances, and servo drift. As such, Nermo-Lite inherently has many asymmetrical features that influence gait performance. Lastly, we also observed material fatigue in the front legs during the extended gait tests. As a result, the front leg compliant knee joints lost performance, resulting in lost gait performance.

8. Conclusion And Outlook

In this thesis, we have outlined and explained the process of developing a digital twin for the bio-inspired robotic rodent Nermo. We developed the digital twin of Nermo in three steps: 1) modeling of the compliant components and spine based features, 2) design of a control architecture for quadruped robots, and 3) transferring the mathematical model of Nermo and mechanical properties (evaluated using FEM) to the simulation in MuJoCo. As part of the spine model, we proposed four unique spine modes: 1) spine-based gait extension to increase the stride length, 2) spine offset compensation to handle the leg offset induced by the spine actuation, 3) spine-based balance compensation to use the spine for improved static stability, and 4) spine lateral flexion turning.

The result of our work is a high-fidelity, tendon-driven, and dynamic digital twin of Nermo with pseudo-compliant joints. To the best of the authors' knowledge, the proposed control architecture is the first that integrates a model-based spine controller into a standard quadruped control architecture. The most crucial element is the capability of the control architecture to handle interdependencies between the spine and leg controllers, such as accounting for leg stance offset caused by spine lateral flexion (spine offset compensation controller). Furthermore, the spine controller can account for multiple spine-based modes without introducing control interference or deadlock. We have also deliberately chosen a fully modular design, allowing the digital twin of Nermo to be used far beyond the research conducted in this thesis.

We conducted four test cases - straight-line, turning, static balance, and maze navigation - with the digital twin to test and evaluate the gait performance of Nermo of the control architecture designed around the compliant legs and spine. The results of the experiments clearly show the effects of the compliant spine on the gait performance of a quadruped robot with compliant legs.

- **Spine-based gait extension:** The spine-based gait extension increases the gait speed by 5 % to 22 % depending on the gait type and frequency. As a trade-off, the spine flexion increases the maximum pitch, roll, and yaw orientation.
- **Spine-based lateral flexion turning:** With spine-based lateral flexion turning, the turning speed increased by 15 % to 30 % compared with leg-based turning. Further, combining leg-based and spine-based turning proved to be a highly successful turning strategy that leverages the speed of the spine-based turning and superior minimum turning radius of the leg-based turning.
- **Spine-based balance compensation:** With spine-based balance compensation enabled, Nermo was capable of balancing in three-foot contact scenarios that were

initially statically unstable. Additionally, the spine balance compensation reduced the pitch variation in stable stance by up to 30 %.

- **Spine-based offset compensation:** The spine-based offset compensation showed small improvements to the roll-behavior when using the spine.

We successfully validated the digital twin of Nermo by conducting a sub-set of the test cases in the real-world using Nermo-Lite. Concretely, we showed that Nermo-Lite's gait speed increased by 15 % with the spine-based gait extension and the turning performance drastically by combining leg vectoring and spine-based lateral flexion turning. These results are significant for further research as we can rely on the ability to transfer the results from experiments with the digital twin to the real world. Establishing a digital twin capable of exhibiting real-world behavior allows for a fast, iterative, and flexible research approach.

With the digital twin of Nermo, we can now investigate many new topics across robot control, learning, and neurorobotics. For a start, we believe in further research and development across the following three topics:

- **Model:** In this thesis, we only considered the spine flexion in the frontal plane. From research on rodent gaits, [6] we know that rodents also use the vertical flexion of the spine for their gait. Therefore in future work, we plan to extend the spine model by the vertical flexion. Next, we plan to expand on the single revolute discretized compliant joint model with an advanced compliant joint model. The advanced compliant joint model could, for example, consist of a series connection of revolute joints to closer approximate the angular and positional displacement of a compliant joint. We also believe that implementing a dynamic model will allow for more robust robot control over uneven terrain and help predict the passive wrist and ankle deflection during gait. Lastly, we also see much value in developing an empirical model of the servo motors better to understand the torque-to-speed and torque-to-current characteristics of the servos. Such a model can then be implemented in MuJoCo to simulate the motor dynamics accurately.

- **Controller:** Based on a dynamic model of Nermo, we plan to extend the controller feedback with dynamic control. We also plan to include Nermo's ground contact sensors to improve pose estimation and perform force control. Such changes can help improve the robustness of the controller to disturbances and obstacles. In addition, the controller can be tuned to different cost functions - for example, based on reducing roll orientation noise or motor power consumption - using the empirical servo model. Lastly, we believe there is a significant opportunity to use the digital twin to train and implement learning and hybrid-based controllers.

- **Nermo:** Using the digital twin, we see much potential in improving the performance of Nermo beyond its current state. We already outlined one such example in this thesis - the improved rear legs. Similarly, we plan to improve the design of

the compliant front legs and spine. We also plan to redesign the neck of Nermo to be compliant like the spine.

Developing a digital twin for the bio-inspired rodent robot Nermo has set the foundation for future research into designing, modeling, and controlling small, compliant, and underactuated quadruped robots. The digital twin can be used to investigate and explore the performance of novel bio-inspired learning and navigation algorithms. Furthermore, our work has shown practical ways to improve the performance of small, underactuated quadrupeds through a flexible spine. The compliance makes the quadrupeds more robust under challenging conditions and better mimics the behavior of rodents.

A. Appendix

A.1. Methodology Supplement Material

A.1.1. Results of FEA Simulation for Compliant Joints

This section covers the finite element simulations of the forelimb, hindlimb, and compliant spine. We conducted the simulations to extract the joint stiffness for the discretized compliant joints as described in Section 5.1.1. In each simulation case, we applied a torque T_j of 0.125 N·m at the joints and measured the angular displacement θ_j. We then solved for the joint stiffness k_j using the relation $T_j = k_j \cdot \theta_j$. We assume that the joint stiffness remains constant at the joint. It should be noted that the spine is a series connection of four joints such that $k_j = k_{j,v}/4$, where $k_{j,v}$ is the stiffness of the individual joints.

For the FEM simulations, we used the FEM solver integrated with Autodesk Fusion 360 [45]. Autodesk Fusion 360 is a cloud-based CAD/CAE/FEM software provided by Autodesk. We used the study type static non-linear stress and the material Organisol Invent Smooth PA12 [46] from the Autodesk Fusion 360 library. Figure A.1 shows the constraint and torque application point during each test. The results of the tests and joint stiffness are listed in Table A.1.

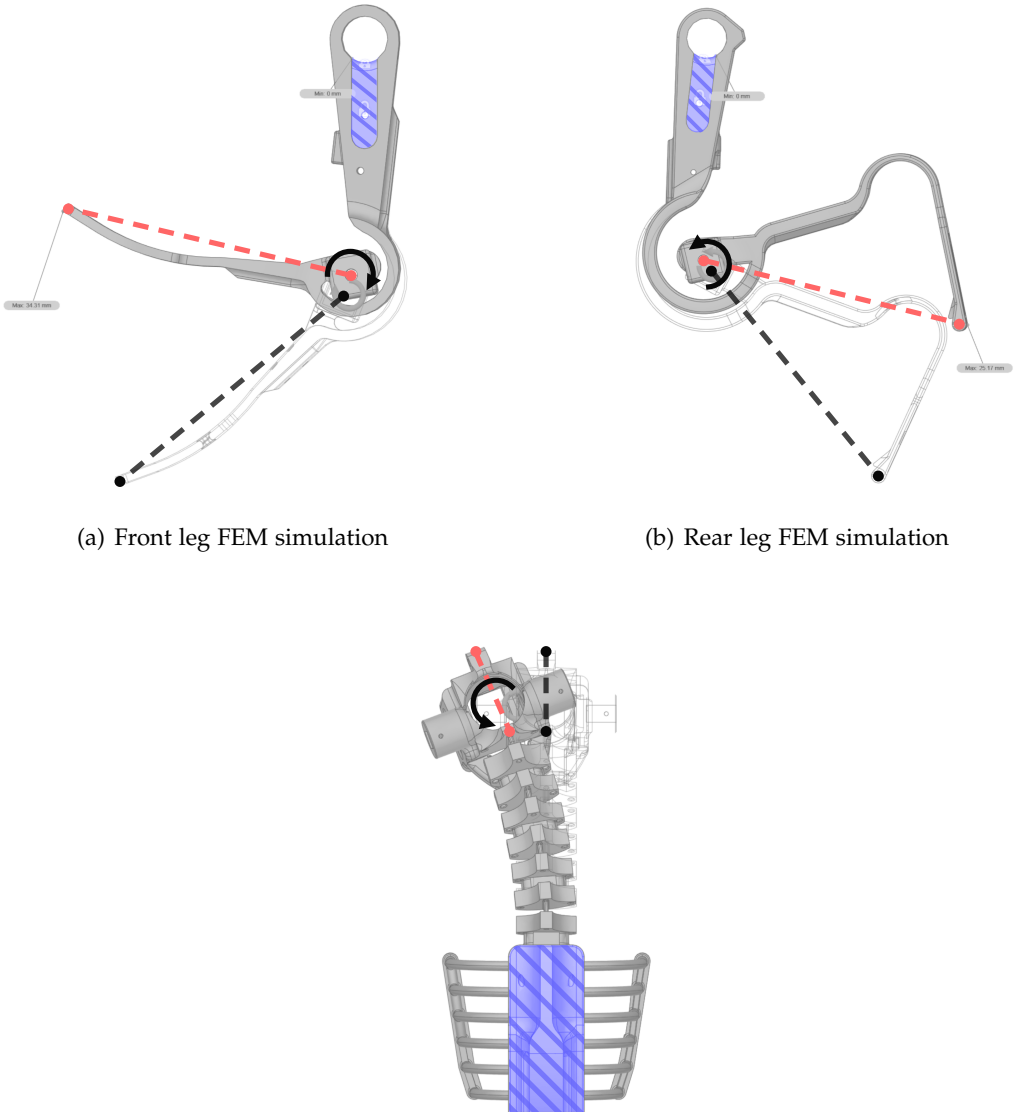
Table A.1.: Joint stiffness values for discretized compliant joints.

Test Case	θ_j [rad]	k_j [N·m/rad]
Forelimb	0.95	0.13
Hindlimb	0.65	0.19
Spine	0.42	1.20

The computed joint stiffness from the FEM simulation is used as the joint stiffness parameter in MuJoCo. The damping values c_j were evaluated through parameter tuning of the simulation. We also applied a joint armature value to each joint in MuJoCo to improve simulation stability (as recommended by the developers of MuJoCo, see section Armature in [47]).

A.1.2. Type 3 Tendon Configuration

The type 3 tendon configuration discussed in Section 5.1.2 is a close approximation for the front and rear leg designs in Nermo. Figure A.2 illustrates how the tendon in the real



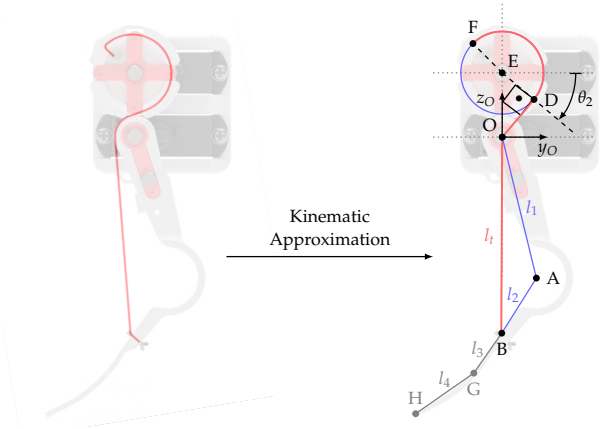
(a) Front leg FEM simulation

(b) Rear leg FEM simulation

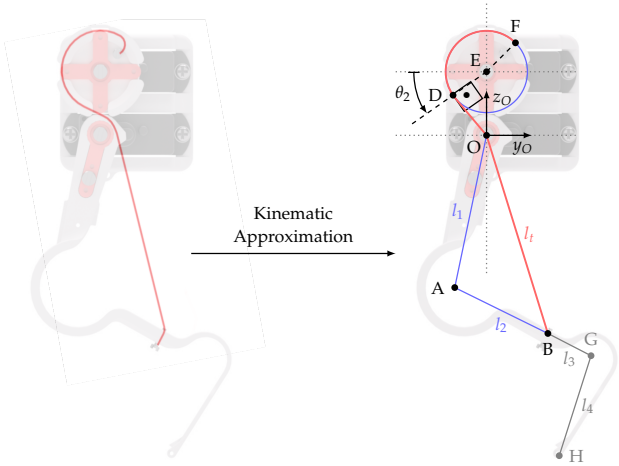
(c) Spine FEM simulation

Figure A.1.: FEM simulation of the compliant components for joint discretization. The red dashed line indicates the end configuration. The silhouette component shows the part before deflection. The black dashed line indicates the starting configuration. The blue striped area indicates the constraints, and the black arrow the torque application point.

front and rear leg designs is routed around the shoulder/hip axis joint and consequently the tendon pulley. In this specific tendon routing configuration the influence of the tendon wrapping around the shoulder/hip axis on the foot-end position is negligible compared to the wrapping around the tendon pulley. Therefore, the real front and rear leg designs can be approximated with the type 3 tendon configuration as depicted in the right side of Figure A.2.



(a) Front leg type 3 configuration



(b) Rear leg type 3 configuration

Figure A.2.: Approximation of the leg designs as type 3 tendon configurations.

A.1.3. Gait Factor

The gait factor is a multiplier that accounts for the varying duty factor and leg phase offset values between the different gait types. Consider a scenario where the stride length of a single leg is 0.06 m for one gait cycle. For the fictitious gait type with duty factor 1 and phase offset 0 the effective stride length for a complete gait cycle is 0.06 m.

The effective gait stride length is also 0.06 m if the phase offset is changed to 0.5 with a duty factor of 1. However, for the ideal trot with duty factor 0.5 and phase offset 0.5, the effective stride length for a complete cycle is 0.12 m. The reason for this difference is clear; in the ideal trot the half-gait cycles in swing and stance state occur in parallel, where as in the fictitious gait the half-gait cycles occur concurrently. The gait factors for the gait types used in this thesis are listed in Table A.2.

Table A.2.: Gait factor values for each gait type used in this thesis.

Gait Type	Gait Factor
Lateral Sequence Walk	1.47
Walking Trot	1.67
Ideal Trot	2.00

A.1.4. Control Architecture

This section provides details on the high-level controller and gait knowledge-base modules of the control architecture.

High-Level Controller

The high-level controller is responsible for two main functions: 1) handling the interface between upstream control or input devices and 2) processing the upstream control commands $\mathbf{c}_m$ and mouse sensor data $\mathbf{q}_{m,s}$ into mouse velocity v_m, turn-rate α_{tr} and gait mode $\mathbf{g}_m$ commands for the low-level motion module. Therefore, the high-level control commands the high-level behavior of Nermo.

The ability to follow a desired trajectory is one important behavior computed by the high-level control. For trajectory control, the controller is capable of being run in two modes:

- **Internal Trajectory Mode**: In this mode, the high-level controller does not require an upstream control signal. Instead, we define simple trajectories directly in the high-level controller (such as straight-line following).

- **External Trajectory Mode**: In this mode, the high-level controller receives control values c_m in velocity space from an upstream module - such as an input device like a game console controller.

The selection of the high-level trajectory control mode depends on the connected upstream device and the specific use case. We use the internal trajectory mode for basic trajectory following in the experiments.

Table A.3.: A comparison of physical and hardware specifications between Nermo and Nermo-Lite.

Physical Property	Nermo	Nermo-Lite
Mass [g]	275	270
Servo Count	13	11
Battery [mAh]	1000	500
Run Time	25 min.	15 min.
Main Electronics	RPi Zero	ESP-01s
Material of Skeleton	PA-12 (SLS)	PETG (FDM)
Approx. Cost [EUR]	1000	200

Gait Knowledge-Base

The gait knowledge base contains gait specific control parameters and, in future, heuristics for gait transition. The gait knowledge base currently contains parameter for three gait types: 1) the lateral sequence walk, 2) the walking trot, and 3) the ideal trot. The gait diagrams are covered in section 2.1. The ideal trot is a unique variant of the trot with a duty factor of 0.5. That means that only two feet are in contact with ideal conditions at any given point in the gait cycle. The numeric values for the gait parameters used in the tests can be found in section A.4.1.

A.2. Specifications for Nermo-Lite

This section covers all details for Nermo-Lite, the modified version of Nermo specific to the experiments carried out in this thesis. First, we cover the different areas of Nermo that were modified or redesigned for Nermo-Lite. Second, we provide a complete list of parameter values of Nermo relevant for all models.

A.2.1. Mechanical Redesign of Nermo

This section covers the necessary steps taken to reduce the sim-to-real gap by applying specific modifications to Nermo. The modifications were implemented in a new version of Nermo named Nermo-Lite. The difference in mechanical and electrical specifications between Nermo and Nermo-Lite are listed in Table A.3. Beyond the lower servo count, the servo motors in Nermo-Lite are also unmodified (i.e. without a custom PID control board). Furthermore, Nermo-Lite is designed to be fully printable using a desktop fused deposition modeling (FDM) printer compared to a far more expensive, industrial selective laser sintering (SLS) machine. With all the changes combined, the total cost of Nermo-Lite is approx. 200 EUR (800 EUR less than Nermo).

Mechanical Redesign of Spine

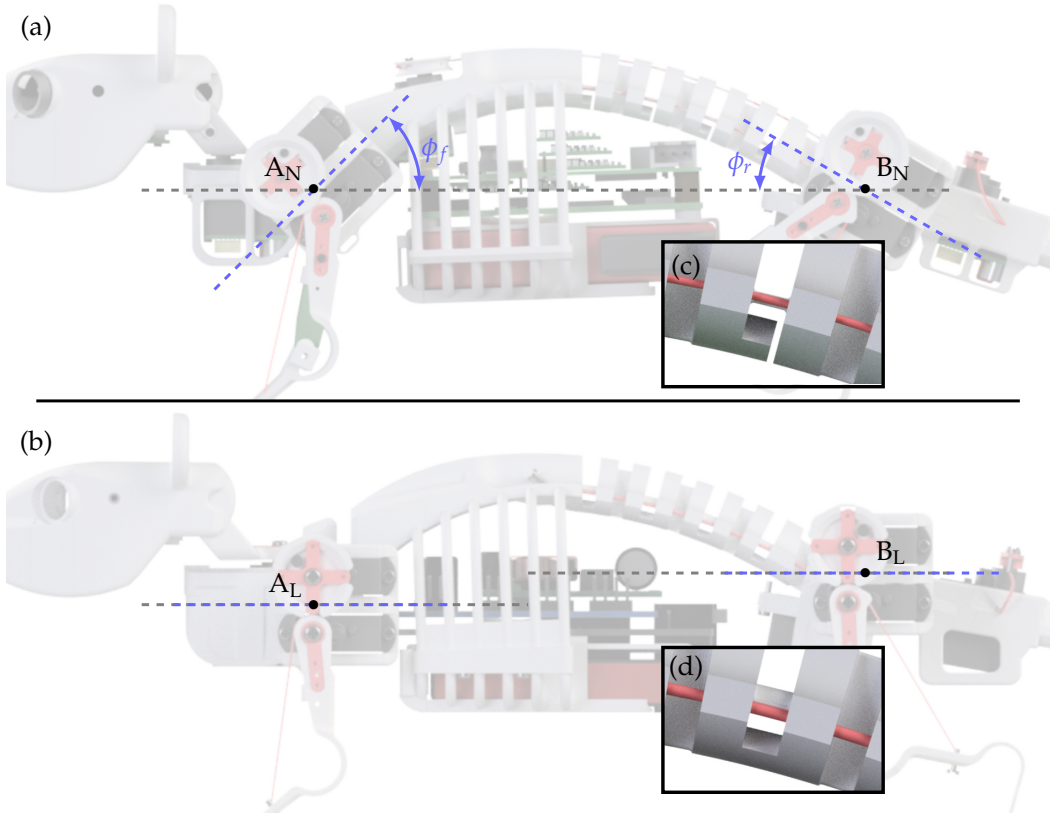


Figure A.3.: Comparison of design modifications between (a) Nermo and (b) Nermo-Lite. The front and rear leg caddies are rotated by 45 degrees and 30 degrees, respectively, to be fully horizontal. (c) The spine of Nermo with unmodified vertical flexing vertebrate. (d) The spine of Nermo-Lite with fused vertical vertebrates, which do not allow vertical flexion.

The two critical design differences are shown in Figure A.3. In Nermo, the front and rear leg mounting caddies are oriented by 45 and 30 degrees, respectively, as shown in Figure A.3(a). In Nermo-Lite, following the MuJoCo model, the front and rear leg caddies are horizontal. The new orientation of the leg caddies improved tendon routing around the shoulder and hip joints to closer replicate the type 3 tendon arrangement used in the MuJoCo and kinematic models. The second modification focuses on the degrees of freedom in the compliant spine. The spine in Nermo has vertebrates that can flex in the vertical and horizontal direction as seen in Figure A.3(c). However, as covered earlier, the kinematic and MuJoCo models of the mouse only evaluate the horizontal flexion of the spine. The additional vertical degree of freedom in the spine of Nermo introduces kinematics and dynamics not considered in the controller kinematic models and MuJoCo model. Therefore the spine in Nermo-Lite has the vertical vertebrate fused

as shown in Figure A.3(d). The fused vertical vertebrates no longer allow vertical spine flexion.

Lastly, we modified several surface contours of the skeleton to allow for manufacturing with desktop hobby 3D printers and to improve the tendon routing during assembly (see Figure A.4. For example, an improved tendon stringing system has been implemented to mimic the neutral tendon configurations more closely between the MuJoCo simulation and Nermo-Lite. The improved system uses 3d-printed leg braces that hold the legs in neutral configuration whilst stringing between the pulley servo and leg attachment point. Additionally, the nuts at the tendon mounting points allow fine-tuning of the tendon stiffness.

Mechanical Redesign of Rear Leg

One last step to reduce the real-to-sim gap is the redesign of the rear mouse leg to match the implemented kinematic models. The first versions of Nermo all used a bio-inspired three-bar leg consisting of the upper leg (femur), lower leg (tibia and fibula), and foot. However, with the switch to a compliant knee, the old three-bar rear leg lacked the necessary stiffness in the compliant knee joint for optimal performance. Subsequently, the rear leg was redesigned to mimic a four-bar linkage to provide additional support for the knee joint.

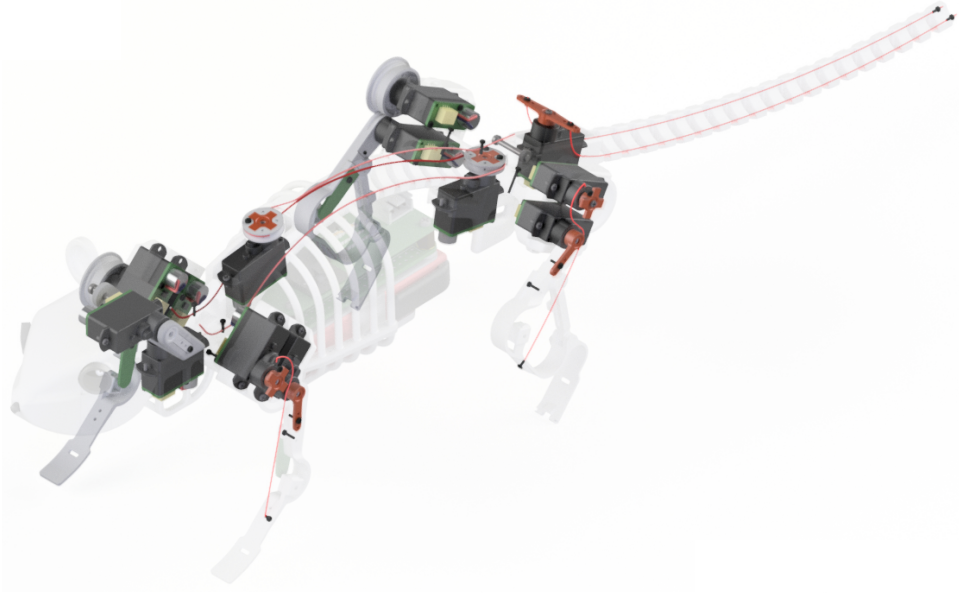
However, the four-bar linkage design suffered from three major drawbacks:

- The four-bar linkage mechanism no longer closely mimics the leg behavior of a biological rodent mouse. Therefore, it is difficult to relate metrics such as the range of motion between the biological and bio-mimetic leg designs from a modeling perspective.
- The four-bar linkage mechanism adds additional modeling complexity compared with the three-bar leg.
- The four-bar linkage mechanism adds additional asymmetry between the already asymmetric behavior of the front and rear legs.

Therefore, the rear leg was once again redesigned into a three-bar linkage mechanism to fix the drawbacks of the four-bar linkage.

The new rear leg retains the compliant knee and ankle joint of the previous version. We used the flat beam torsional spring model outlined in earlier sections, range of motion information from biological rodents, and FEM simulations to design the new rear leg. The design difference between the old and new rear leg are shown in Figure A.5. The redesigned rear leg is implemented in Nermo and Nermo-Lite, closely matches the leg models used in MuJoCo and the controller, and closes the sim-to-real gap further. However, detailed discussions regarding the new rear leg's design and performance are beyond the scope of this thesis.

(a)



(b)

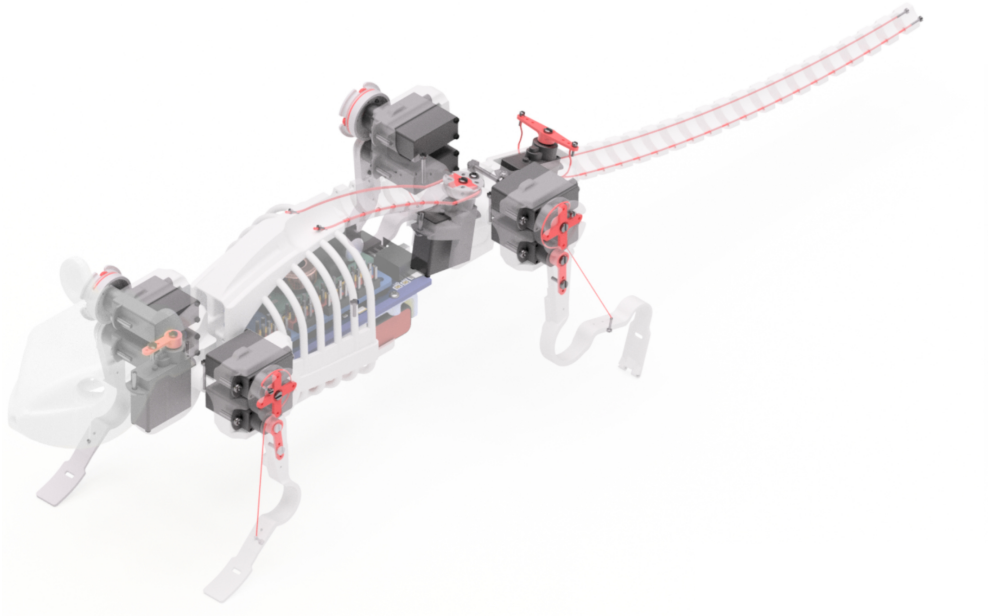


Figure A.4.: Comparison of the actuation systems in (a) Nermo and (b) Nermo-Lite. Nermo-Lite does not have the head tilt and spine vertical flexion servos. Furthermore, many of the tendon runs between actuator and component have been improved. Nermo-Lite also has the new rear legs.

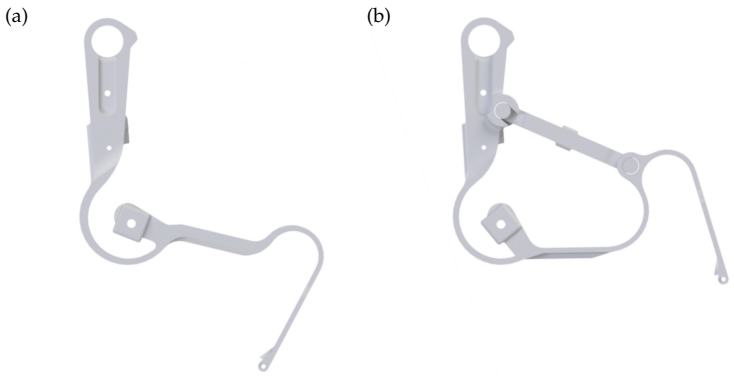


Figure A.5.: Design comparison between the (a) new three-bar linkage rear leg and (b) old four-bar linkage rear leg. Both legs are designed with compliant knee and ankle joints, and the knee joints are tendon-driven.

Redesign of Electrical System

Beyond the specific modifications outlined above, Nermo-Lite also contains four further simplifications. First, Nermo-Lite uses unmodified DSM-44 servos (i.e., using the inbuilt P-type controller). Second, Nermo-Lite uses an ESP8266 as the microcontroller compared with the Raspberry Pi Zero on Nermo. The difference in hardware controller means that Nermo-Lite has a real-time capability for servo control but lacks the additional compute and memory performance for advanced vision or on-chip gait computation. Third, Nermo-Lite omits the second head servo (which enables head pitch) and second spine servo (which enables vertical spine flexion).

A.2.2. Numerical Values for Leg and Spine Models

The numerical values used for the legs and spine kinematic models are given in the following tables.

Table A.4.: Numerical values for front and rear leg model parameters.

Parameter	Front Leg [mm]	Rear Leg [mm]
l_1	29.5mm	31.7mm
l_2	14.5mm	20mm
l_3	22.5mm	30.5mm
l_4	14.5mm	20.5mm
r_p	8mm	8mm
d_1	12.8mm	12.8mm
l_{t0}	33mm	32mm

The constant ankle/wrist angles are $\theta_{3_f} = 23^\circ$ for the front legs and $\theta_{3_r} = 73^\circ$ for the rear legs. In Table A.5 the positions of key modeling components are listed.

Table A.5.: Coordinate positions of critical mouse components for modeling with respect to the spine base frame {0}.

Component	Position (x,y,z) [mm]
Front Left Shoulder	(0.0368, -0.0615, 0.0024)
Front Right Shoulder	(-0.0368, -0.0615, 0.0024)
Rear Left Hip	(0.0428, 0.0604, 0.0095)
Rear Right Hip	(-0.0428, 0.0604, 0.0095)
Spine End ($\theta_{vert} = 0$)	(0.0, 0.0516, -0.0158)
COM Front	(0.0, -0.0417, -0.027)
COM Rear (w.r.t spine end)	(0.0, 0.0310, - 0.0160)

A.3. Digital Twin Setup

This appendix section covers additional details regarding the modeling techniques used to develop the high-fidelity digital twin in MuJoCo.

A.3.1. Modular Modeling Approach

Instead of modeling the digital twin in one monolithic XML file, we created several modular model blocks and combined them in a main model file. The modular model blocks were divided into the limb and tail components of Nermo. Further, as per the three core elements in MuJoCo, each model block contains three MuJoCo XML files defining the model geometry, the model tendons, and the model sensors and actuators. Therefore, the modular modeling approach closely matches the modular design of the real-world mouse.

In addition, the modular modeling approach has three key design and usability benefits. First, different leg and tail model blocks can be easily exchanged to evaluate different joint, geometry, tendon, and motor parameters. Second, the model blocks are re-usable across various test problems and setups, such as simplified walker setups to test new algorithms. Third, the model structure is easier to understand and debug since issues can be isolated to individual model blocks.

A.3.2. Functional Component Modeling

Joints

The compliant joints of Nermo are approximated as virtual torsion spring-damper components applied at the joint. The respective dynamics of the virtual spring-dampers

are evaluated from FEM simulations to approximate the joint stiffness. We chose the damping values from isolated leg experiments in MuJoCo to ensure a stable simulation.

Actuators

The motors are modeled as position-controlled, proportional servos. Each motor in MuJoCo also has a control and torque range that mimics the performance of the real-world servos.

Tendons

We modeled the tendons as spatial without stiffness. Each tendon was given a fixed length range, such that the tendon allowed for force transfer during pull behavior but exerted no force during push behavior.

A.3.3. Integration of CAD Model With MuJoCo

To create the visual model of Nermo in MuJoCo, we imported the entire CAD model as STL-meshes for MuJoCo geometries. The use of the STL converted CAD components provide two significant benefits: 1) the development of a high-fidelity visual model of Nermo and 2) the ability to cross-check active components for collisions in simulation.

However, importing the STL model files is not entirely straightforward. Simply importing the model STLs into MuJoCo and applying them into a geometry body creates a convex mesh around the body for collision detection. This is done automatically by MuJoCo. MuJoCo also infers mass and inertia properties for these components. One drawback of this approach is the added simulation complexity and computation created by the more complex convex mesh contact checking required for the STL meshes than the sphere and capsule geometries.

To solve this problem, a low-fidelity basic geometry model of Nermo runs parallel to STL based model. The dual model approach is visualized in Figure A.6. The basic geometry model handles all contact, self-collision, and dynamics. The low-fidelity model has the alpha channel set to 0 % to avoid rendering artifacts and thus is not visible in the renderer. The visual STL model is only used for visual purposes and does not influence the model dynamics. The lack of influence on the model dynamics is achieved by turning off the contact parameters for the geometry bodies containing the STL models and setting the mass, and thus automatically the inertia matrices, to zero.

A.4. Supplementary Details for Experiments

A.4.1. Test Scenario Gait Parameters

This section contains the numerical values used for the different gait parameters in the simulation and real-world tests. Note that we keep all gait parameters listed here

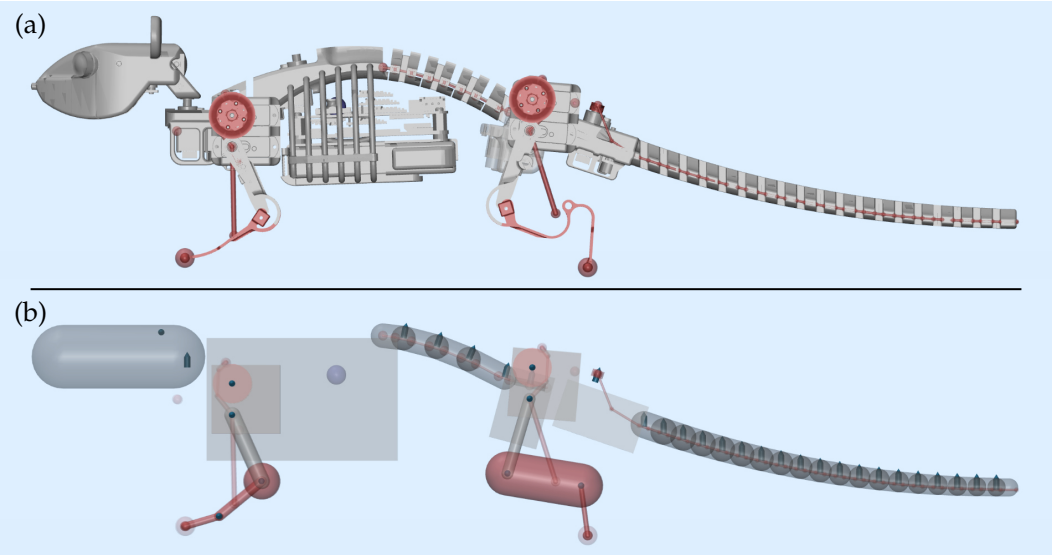


Figure A.6.: Comparison between high-fidelity (a) and low fidelity (b) models used in the MuJoCo simulation. In (b) the joints (blue arrows) and tendon arrangements are also visible.

constant across simulation and real-world tests. Table A.6 contains the gait parameters. The duty cycle refers to the ratio of time between the swing and stance phase. The offsets per leg is a stance phase offset value for each leg in the order: front left leg, front right leg, rear left leg, and rear right leg. Gait frequency values are dependent on the individual test cases.

Table A.6.: Numerical values for the gait parameters used during testing.

Gait Type	Duty Cycle	Offsets per Leg
Ideal Trot (trt)	0.5	[0.5,0,0,0.5]
Walking Trot (trt1)	0.6	[0.5,0,0,0.5]
Dynamic Walk (lat)	0.68	[0.63,0.14,0,0.54]

Table A.7 lists the leg specific stride parameters. It should be noted that the neutral stance and stride position values are given for the local leg coordinate system and hence the negative values.

A.4.2. Test Scenario Control Parameters

In this section, we outline further details regarding the simulation and real-world tests. First, each simulation test with a specific set of parameters is repeated three times. The same is applied to the real-world tests. The final results are the average of the data.

Table A.7.: Numerical values for the leg stance parameters during testing.

Parameter	Front Leg [mm]	Rear Leg [mm]
Swing Amplitude	0.01	0.01
Stance Amplitude	0.005	0.005
Max Stride Length	0.06	0.06
Neutral Stance Position	-0.045	-0.05
Neutral Stride Position	-0.005	0.0

Beyond the independent and dependent variables, we control for several additional variables in each test scenario. These are listed below for each test case scenario.

Straight-Line Test Constant Parameters

- The high-level controller velocity output to the motion module such that the maximum theoretical stride length of 0.06 m is achieved.
- The leg neutral parameters across all gait types (see Appendix for the full list of parameter values).
- The maximum spine flexion is kept constant throughout all tests to 0.1 rad per spine joint.
- The normalized maximum stride length is kept constant at 0.08 (except for the special stride length tests).

Turn Test Constant Parameters

- The high-level controller velocity output to the motion module such that the maximum theoretical stride length of 0.06 m is achieved.
- The leg neutral parameters across all gait types (see Appendix for the full list of parameter values).
- The maximum spine flexion is kept constant throughout all tests to 0.1 rad per spine joint.
- The normalized maximum stride length is kept constant at 0.08.

Balance Test Constant Parameters

- The leg neutral parameters across all test runs. This ensures the same effective height of the COM of 0.057 m above the ground plane.

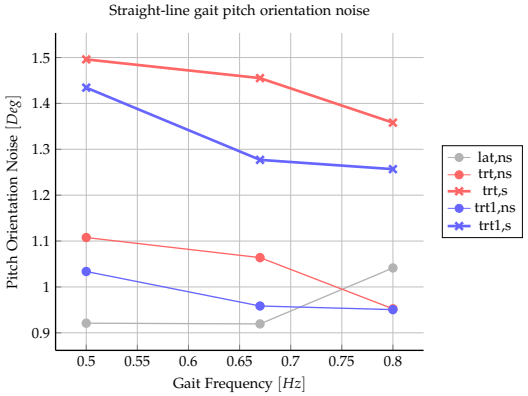
- The maximum spine flexion during the balance compensation is limited to 0.1 rad per spine joint.
- The compression for the rear legs is set to 0.7 rad of the knee servo with a sweep range of -0.7 rad $< q_1 <$ 0.7 rad for the hip servo.
- The compression for the rear legs is set to 0.6 rad of the knee servo with a sweep range of -0.7 rad $< q_1 <$ 0.7 rad for the shoulder servo.

Maze Test Constant Parameters

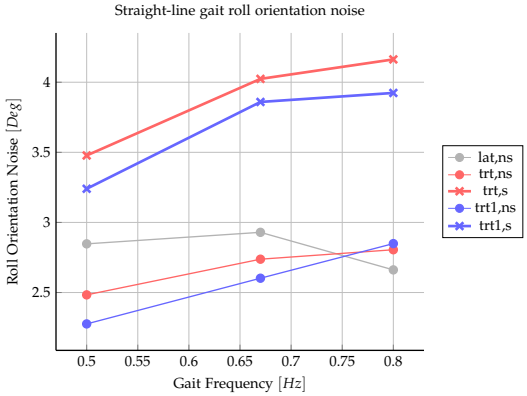
- The high-level controller velocity output to the motion module such that the maximum theoretical stride length of 0.06 m is achieved.
- The leg neutral parameters (see Appendix A.4.1 for full values)
- The maximum spine flexion is kept constant throughout all tests to 0.1 rad per spine joint.
- The normalized maximum stride length is kept constant at 0.08.
- The gait type used is ideal trot (trt) at 0.67 Hz.

A.4.3. Supplementary Experimental Results

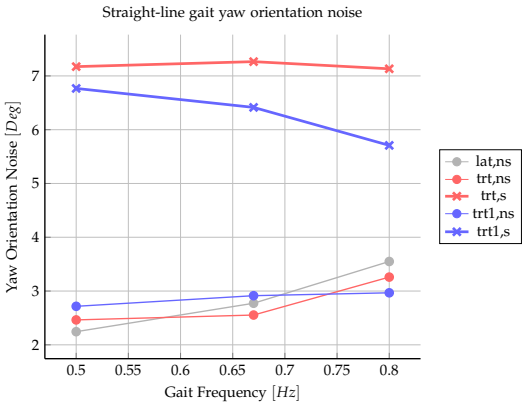
This section contains some additional graphs that help summarize the orientation noise induced by the various gait types, frequencies, and spine actuation. Figure A.7 summarizes the straight-line tests and provides absolute values on the difference in orientation noise across the different test types. Orientation noise depicts the standard deviation of the orientation during the test. Figure A.8 is similar to Figure A.7 but focuses purely on variation of the normalized stride length. Orientation noise depicts the standard deviation of the orientation during the test. Figure A.9 shows the absolute orientation noise with offset compensation enabled during the ideal trot with spine actuation test. Orientation noise depicts the standard deviation of the orientation during the test.



(a) Pitch orientation noise (standard deviation)

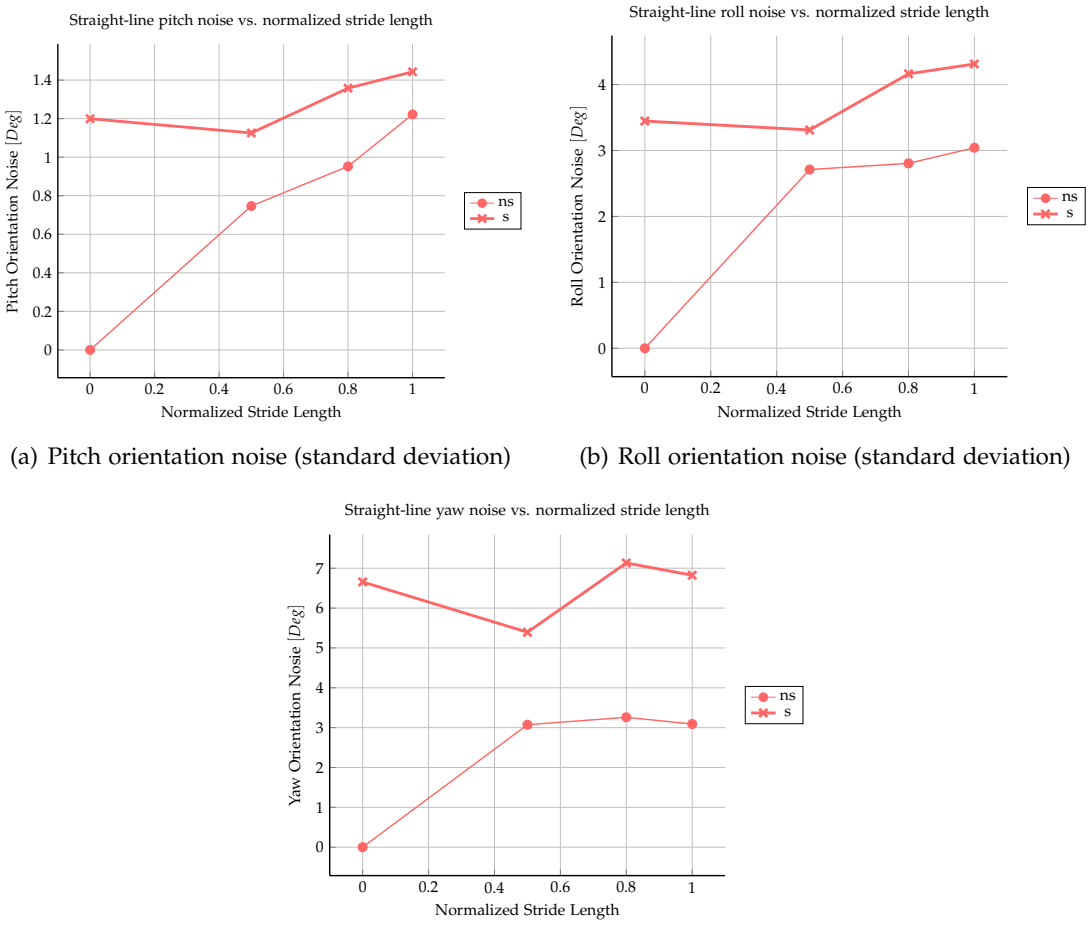


(b) Roll orientation noise (standard deviation)



(c) Yaw orientation noise (standard deviation)

Figure A.7.: Speed and orientation noise of the digital twin for different gait types, gait frequencies, and spine modes. The gait modes used are dynamic walking (lat), walking trot (trt1), and ideal trot (trt). The label (ns) denotes no spine gait extension, and (s) denotes spine gait extension.



(a) Pitch orientation noise (standard deviation)

(b) Roll orientation noise (standard deviation)

(c) Yaw orientation noise (standard deviation)

Figure A.8.: Orientation noise for varying normalized stride lengths. The gait type used is ideal trot (trt) at 0.8 Hz. Results are from simulation.

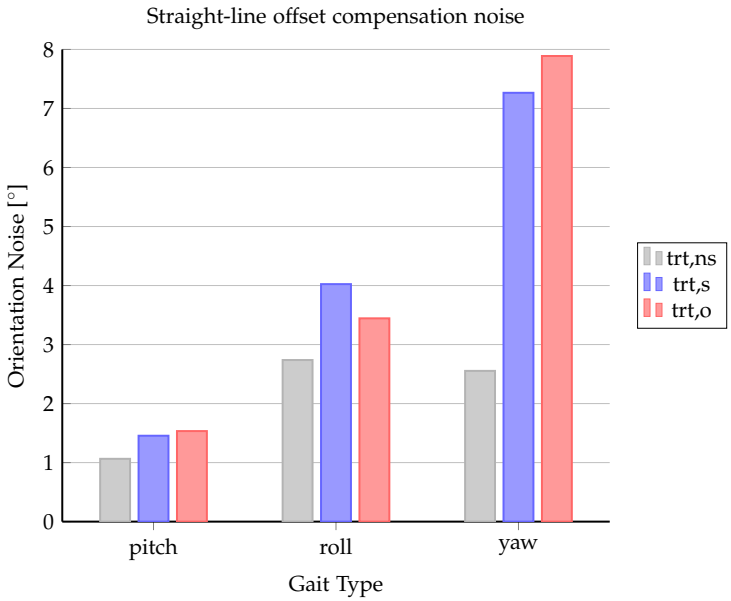


Figure A.9.: Pitch, roll and yaw orientation noise in straight-line tests for gait type with spine offset compensation denoted by the subscript (o), spine gait extension without offset (o), and no spine flexion (ns). Results are from simulation. Gait type ideal trot at 0.67 Hz.

List of Figures

List of Tables

Bibliography

[1] R. D. Beer, "Biologically inspired robotics," *Scholarpedia*, vol. 4, no. 4, p. 1531, 2009.

[2] R. Pfeifer, M. Lungarella, and F. Iida, "Self-organization, embodiment, and biologically inspired robotics," *science*, vol. 318, no. 5853, pp. 1088–1093, 2007.

[3] D. Floreano, A. J. Ijspeert, and S. Schaal, "Robotics and neuroscience," *Current Biology*, vol. 24, no. 18, R910–R920, 2014.

[4] M. Goulding, "Circuits controlling vertebrate locomotion: Moving in a new direction," *Nature Reviews Neuroscience*, vol. 10, no. 7, pp. 507–518, 2009.

[5] O. Kiehn, K. J. Dougherty, M. Hägglund, L. Borgius, A. Talpalar, and C. E. Restrepo, "Probing spinal circuits controlling walking in mammals," *Biochemical and biophysical research communications*, vol. 396, no. 1, pp. 11–18, 2010.

[6] C. Bellardita and O. Kiehn, "Phenotypic characterization of speed-associated gait changes in mice reveals modular organization of locomotor networks," *Current Biology*, vol. 25, no. 11, pp. 1426–1436, 2015.

[7] C. Laschi, B. Mazzolai, F. Patanè, V. Mattoli, P. Dario, H. Ishii, M. Ogura, S. Kurisu, A. Komura, and A. Takanishi, "Design and development of a legged rat robot for studying animal-robot interaction," in *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.*, IEEE, 2006, pp. 631–636.

[8] F. Patanè, V. Mattoli, C. Laschi, B. Mazzolai, P. Dario, H. Ishii, and A. Takanishi, "Biomechatronic design and development of a legged rat robot," in *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2007, pp. 847–852.

[9] D. S. Touretzky, H. S. Wan, and A. D. Redish, "Neural representation of space in rats and robots," *Computational Intelligence: Imitating Life*, pp. 57–68, 1994.

[10] N. Burgess, J. G. Donnett, K. J. Jeffery, and J. O–keefe, "Robotic and neuronal simulation of the hippocampus and rat navigation," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 352, no. 1360, pp. 1535–1543, 1997.

[11] M. Milford, A. Jacobson, Z. Chen, and G. Wyeth, "Ratslam: Using models of rodent hippocampus for robot navigation and beyond," in *Robotics Research*, Springer, 2016, pp. 467–485.

[12] Q. Shi, C. Li, K. Li, Q. Huang, H. Ishii, A. Takanishi, and T. Fukuda, "A modified robotic rat to study rat-like pitch and yaw movements," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 5, pp. 2448–2458, 2018.

[13] H. Ishii, Y. Masuda, S. Miyagishima, S. Fumino, A. Takanishi, C. Laschi, B. Mazzolai, V. Mattoli, and P. Dario, "Design and development of biomimetic quadruped robot for behavior studies of rats and mice," in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, 2009, pp. 7192–7195.

[14] (Nov. 9, 2021). Wr(waseda rat) series, [Online]. Available: `http://www.takanishi.mech.waseda.ac.jp/top/research/rat/robot-WR.html`.

[15] S. Heath, C. A. Ramirez-Brinez, J. Arnold, O. Olsson, J. Taufatofua, P. Pounds, J. Wiles, E. Leonardis, E. Gygi, E. Leija, *et al.*, "Pirat: An autonomous framework for studying social behaviour in rats and robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 7601–7608.

[16] C. Li, Q. Shi, Z. Gao, M. Ma, Q. Huang, H. Ishii, A. Takanishi, and T. Fukuda, "Bioinspired phase-shift turning action for a biomimetic robot," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 1, pp. 84–94, 2019.

[17] P. Lucas, F. Walter, and A. Knoll, "Design of a biomimetic rodent robot," 2018.

[18] M. Reckhaus, N. Hochgeschwender, J. Paulus, A. Shakhimardanov, and G. K. Kraetzschmar, "An overview about simulation and emulation in robotics," *Proceedings of SIMPAR*, pp. 365–374, 2010.

[19] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, 2021.

[20] R. M. Alexander, "Walking, running and jumping," in *Locomotion of animals*, Springer, 1982, pp. 81–113.

[21] M. Hildebrand, "Symmetrical gaits of horses: Gaits can be expressed numerically and analyzed graphically to reveal their nature and relationships.," *Science*, vol. 150, no. 3697, pp. 701–708, 1965.

[22] K. Clarke and J. Still, "Gait analysis in the mouse," *Physiology & behavior*, vol. 66, no. 5, pp. 723–729, 1999.

[23] C. S. Mendes, I. Bartos, Z. Márka, T. Akay, S. Márka, and R. S. Mann, "Quantification of gait parameters in freely walking rodents," *BMC biology*, vol. 13, no. 1, pp. 1–11, 2015.

[24] R. McNeill Alexander, "Energetics and optimization of human walking and running: The 2000 raymond pearl memorial lecture," *American journal of human biology*, vol. 14, no. 5, pp. 641–648, 2002.

[25] N. C. Heglund, C. R. Taylor, and T. A. McMahon, "Scaling stride frequency and gait to animal size: Mice to horses," *Science*, vol. 186, no. 4169, pp. 1112–1113, 1974.

[26] M. S. Fischer, N. Schilling, M. Schmidt, D. Haarhaus, and H. Witte, "Basic limb kinematics of small therian mammals," *Journal of Experimental Biology*, vol. 205, no. 9, pp. 1315–1338, 2002.

[27] M. F. Bonnan, J. Shulman, R. Varadharajan, C. Gilbert, M. Wilkes, A. Horner, and E. Brainerd, "Forelimb kinematics of rats using xromm, with implications for small eutherians and their fossil relatives," *PloS one*, vol. 11, no. 3, e0149377, 2016.

[28] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[29] J. K. Hodgins and M. H. Raibert, "Adjusting step length for rough terrain locomotion," *Dynamically Stable Legged Locomotion*, p. 27, 1991.

[30] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 236–258, 2011.

[31] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 2245–2252.

[32] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2016, pp. 38–44.

[33] Verge, *Boston dynamic spot*, [Online; accessed Nov. 9, 2021], 2019. [Online]. Available: `https://cdn.vox-cdn.com/thumbor/U_XCbVRFppYvnLxwZ5Sshcg3_yI=/800x0/filters:no_upscale()/cdn.vox-cdn.com/uploads/chorus_asset/file/19224524/bfarsace_190919_3680_0095.jpg`.

[34] N. Kau and S. Bowers, "Stanford pupper: A low-cost agile quadruped robot for benchmarking and education," *arXiv preprint arXiv:2110.00736*, 2021.

[35] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019.

[36] S. Wang, Q. Shi, J. Gao, Y. Wang, F. Meng, C. Li, Q. Huang, and T. Fukuda, "Design and control of a miniature quadruped rat-inspired robot," in *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, IEEE, 2019, pp. 346–351.

[37] R. Kawasaki, R. Sato, E. Kazama, A. Ming, and M. Shimojo, "Development of a flexible coupled spine mechanism for a small quadruped robot," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2016, pp. 71–76.

[38] M. Khoramshahi, A. Spröwitz, A. Tuleu, M. N. Ahmadabadi, and A. J. Ijspeert, "Benefits of an active spine supported bounding locomotion with a small compliant quadruped robot," in *2013 IEEE international conference on robotics and automation*, IEEE, 2013, pp. 3329–3334.

[39] P. Eckert, A. Spröwitz, H. Witte, and A. J. Ijspeert, "Comparing the effect of different spine and leg designs for a small bounding quadruped robot," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 3128–3133.

[40] Q. Zhao, B. Ellenberger, H. Sumioka, T. Sandy, and R. Pfeifer, "The effect of spine actuation and stiffness on a pneumatically-driven quadruped robot for cheetah-like locomotion," in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2013, pp. 1807–1812.

[41] B. Chong, Y. O. Aydin, C. Gong, G. Sartoretti, Y. Wu, J. M. Rieser, H. Xing, P. E. Schiebel, J. W. Rankin, K. B. Michel, *et al.*, "Coordination of lateral body bending and leg movements for sprawled posture quadrupedal locomotion," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 747–763, 2021.

[42] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," in *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 4397–4404.

[43] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, F. Golemo, M. Mozifian, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, *et al.*, "Perspectives on sim2real transfer for robotics: A summary of the r: Ss 2020 workshop," *arXiv preprint arXiv:2012.03806*, 2020.

[44] (2021). Power-hd digital servo dsm44 - engel modellbau + technik. [Online; accessed Nov. 15, 2021], [Online]. Available: `https://www.engelmt.de/mtrc-electronic/power-hd-servos/power-hd-digital/power-hd-digital-servo-dsm44`.

[45] (2021). Fusion 360 - cloud powered 3d cad/cam software for product design - autodesk. [Online; accessed Nov. 11, 2021], [Online]. Available: `https://www.autodesk.eu/products/fusion-360/overview`.

[46] (Feb. 2014). Pa12 materials - inventor - autodesk knowledge network. [Online; accessed Nov. 11, 2021], [Online]. Available: `https://knowledge.autodesk.com/support/inventor-products/learn-explore/caas/CloudHelp/cloudhelp/2014/ENU/Inventor/files/GUID-38CC917B-418C-4F27-BA0A-451D42DEE8E5-htm.html`.

[47] (2021). Xml reference — mujoco documentation. [Online; accessed Nov. 11, 2021], [Online]. Available: `https://mujoco.readthedocs.io/en/latest/XMLreference.html`.