Trajectory Generation for Flight Phase of a Quadruped Robot Jump

by

Beste Bahçeci

Submitted to
the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

SABANCI UNIVERSITY

July, 2022

Trajectory Generation for Flight Phase of a Quadruped Robot Jump

Beste Bahçeci

ME, Ph.D. Thesis, 2022

Thesis Supervisor: Assoc. Prof. Dr. Kemalettin Erbatur

Keywords: Quadruped robots, angular momentum, reinforcement learning, reference trajectory generation, free-fall manipulators, flight stability, landing stability, jumping motion, legged robots

Abstract

Legged robots excel in navigating challenging natural environments, such as steep obstructions or wide gaps in the ground. Apart from rough terrain, they may confront unexpected impact forces during their leaping gaits. While facing external disturbances, legged robots should maintain and restore their stability while completing their gaits. External disturbances and body orientation errors should be identified. Appropriate actions have to be taken to restore the balance of the robot and provide advantageous landing circumstances.

This dissertation examines the robot body orientation errors during the flight phase and first offers a unique posture control method that uses reinforcement learning to build reference trajectories for a quadrupedal robot with waist joints during a long jump flight phase. Then, another novel algorithm for posture recovery is provided, this time based on angular momentum. The same algorithm is altered to account for perturbations in the flying phase caused by a push on the robot's body. We describe a push recovery method that uses angular momentum to build reference trajectories for the long jump. This work also contains a more detailed angular momentum-based reference generating approach for posture recovery. Real-time centroidal dynamics computation is employed in this second technique.

These approaches provide reference trajectories for the waist and rear hip joints of the quadrupedal robot in order to acquire the desired orientation of the robot in the air. PID joint position control is used to track reference trajectories. The robot model used in the calculations is comprehensive since each component of the robot's body—the leg links and three torso portions—is represented by individual parameters. The suggested techniques for trajectory creation are computationally efficient, making them suited for use in real-time applications. The proposed posture control and push recovery approaches are tested on the model of a quadrupedal robot during the flight phase of a long jump via simulations. The findings reveal that the proposed methods are accurate in terms of angular position and angular velocity regulation and can achieve successful landing postures.

Dört Bacaklı Robotun Uzun Atlayışı için Uçuş Evresi Referans Yörüngesi Sentezi

Beste Bahçeci

ME, Doktora Tezi, 2022

Tez Danışmanı: Doç. Dr. Kemalettin Erbatur

Anahtar kelimeler: Dört bacaklı robotlar, açısal momentum, destekli öğrenme, referans yörüngesi sentezi, serbest düşüş manipülatörü, uçuş dengesi, iniş dengesi, atlama hareketi, bacaklı robotlar

Özet

Bacaklı robotlar, dik engelleri veya zeminde geniş boşlukları olan zorlu doğal arazilerde hareket etmek için kullanışlı özelliklere sahiptirler. Engebeli arazilerde hareket etmenin zorluklarının yanı sıra, sıçrama hareketleri sırasında beklenmedik darbelerle karşılaşa- bilirler. Dışarıdan gelen denge bozabilecek etkilerle maruz kalırken planlanan hareketlerini tamamlamalı, dengelerini korumalı ve düzeltebilmelidirler. Harici darbeler veya vücut yönelim hataları tespit edilebilmeli ve robotun dengesini yeniden sağlamak ve optimum iniş koşullarını sağlamak için uygun önlemler alınmalıdır.

Bu tezde ilk olarak uçuş aşaması sırasında robot gövdesindeki yönelim hatalarını düzeltmek üzerinde durulmaktadır. Bel eklemlerine sahip dört bacaklı bir robotun gerçekleştirdiği uzun atlamada tamamen havada olduğu süre için referans yörüngeleri oluşturulmakta, bu amaçla destekli öğrenme tekniği kullanılarak yeni bir vücut konumu kontrol yöntemi sunulmaktadır. Ardından, açısal momentuma dayalı bir yöntem kullanılarak, bozulan vücut konumunun düzeltilmesine gidilmektedir. Ayrıca, aynı algoritma yardımı ile robotun gövdesine uçuş aşamasında isabet eden bir itme darbesinin neden olduğu düzensizliklerin giderilmesi için çevrimiçi referans yörünge oluşturan bir itme kurtarma yöntemi sunulmaktadır. Tezde açısal momentuma dayalı ikinci bir robot yörüngesi düzeltme tekniği de geliştirilmektedir. Bu teknik atalet değerlerini gerçek zamanlı hesaplaması açısından ilk önerilen yöntemle farklılıklar içermektedir.

Bu yaklaşımlar robotun havada istenen oryantasyonunu elde etmek için dört bacaklı robotun bel ve arka kalça eklemleri için referans yörüngeler üretmektedir. Referans yörüngelerini takip etmek için PID eklem konum kontrolü uygulanmaktadır. Hesaplamalarda kullanılan robot modeli, robotun gövdesinin her bir bileşenini yani bacak parçaları ve üç gövde kısmı kütle değeri ile temsil edildiğinden kapsamlıdır. Yörünge oluşturma için önerilen teknikler hesaplama açısından verimlidir ve bu da onları gerçek zamanlı uygulamalarda kullanıma uygun kılmaktadır. Önerilen algoritmalar, uzun bir atlayışın uçuş aşaması sırasında dört bacaklı bir robotun simülasyonu ile test edilmiştir. Bulgular, yöntemlerin açısal konum ve açısal hızı düzenlemede etkili olduğunu ve uygun bir iniş pozisyonu sağlayabildiğini ortaya koymaktadır.

Acknowledgments

I would like to thank and express my gratitude to everyone who guided and supported me throughout my journey toward earning my Ph.D.

First and foremost, I would like to express my deepest gratitude to my Ph.D. thesis supervisor, Dr. Kemalettin Erbatur. His endless support towards overcoming obstacles I faced, paved the way through this accomplishment. His encouraging guidance and warm attitude made this thesis possible. During thesis writing period, he has been extraordinarily tolerant and supportive. I will always be indebted to him for shaping my future in academic life.

I would like to thank the members of my dissertation jury, Asst. Prof. Dr. Melih Türkseven, Prof. Dr. Albert Levi, Assoc. Prof. Dr. Ahmet Onat, and Asst. Prof. Dr. Özkan Bebek for offering their time and opinions in order to assist me in improving the quality of my thesis. It is an incredible chance and honor for me to have distinguished professors participate on the dissertation jury for my dissertation.

I would like to express my gratitude to all of my research colleagues, Dr. Ömer Kemal Adak, Dr. Ahmet Selim Pehlivan, Dr. Mehmet Mert Gülhan, Orhan Ayit, and Koray Erkekli. A particular thanks to Dr. Ömer Kemal Adak for his encouragement and support as we collaborate to overcome academic obstacles and Dr. Barış Pekerten for his guidance, encouragement and support.

I would like to express my gratitude to my lovely friends İdil Özgün, Dr. Barış Pekerten, Soner Ulun, Serhat Onur Yavaşi, Fırat Tınç, Mina Bayolken, Merve Sayar, Sinem Koç, Evrim Kurtoğlu Can, Bilge Akdemir, Emre Tankal, Tunç Öndemir, Ahmet Fatih Tabak and Çağdaş Erk, who are there for me on both my best and darkest days. Their emotional support made me keep going on this journey.

Additionally, I would like to thank to Cafer Yılmaz, Suat İngin, Ersan Tekson, Füsun Dalamalı, Murat Şakar, Bernis Koyuncuoğlu, Bavver Ergin and Duygu Demir who were with me during the whole writing period of this dissertation and supported me sometimes with a good conversation, sometimes with a good laugh, sometimes with beautiful desserts, sometimes delicious meals and drinks. This would not be possible without them.

I would especially like to thank Zahide Kamışlı for her advice from my childhood. Her effect on my character and her support cannot be denied. Her influence made me a stronger and better person. She will always be a role model for me.

Finally, I would like to thank to my family, my father Hakan Bahçeci, my mother Özdem Bahçeci, my lovely brother Egemen Bahçeci who always finds a way to make me laugh, my grand parents Ülker Bahçeci, Tahir Bahçeci, Kadir Üzer, Emriye Üzer, my uncles Veli Üzer and Egemen Bahçeci, my cousins Sıla Üzer and Sude Üzer. They made me who I am today and they continue to show incredible support in so many ways that anything I can write cannot possibly be enough.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Popular images that come to mind when people hear the word ‘robot’ are Star Wars’ C3PO and R2D2, the Jetsons’ Rosie, and Disney’s Wall-e. They are all intelligent human-inspired machines created to serve humans, and this definition of robot purpose is not far from reality. The term ‘robot’ originated in the 1920s and was initially used to refer to a sort of slave; robots are often defined by their ability to undertake tedious, strenous, or dangerous activities in place of people [23].

In a more recent definition, the Robot Institute of America describes a robot as “a reprogrammable, multipurpose manipulator intended to move material, components, tools, or specialized equipment through several programmed movements to execute a variety of tasks [24].” As technology advances rapidly, so does our understanding of the notion of robots. Nowadays, robots can sense their environment, think logically using a variety of inputs, and act on the physical world [23].

Although the first modern robot examples are industrial robots, popular robot images are all in the form of humans. The area of designing human or animal-like robots is called biologically inspired robotics. We turn to nature for kinematic arrangements in biologically inspired robotics because, in comparison to present robots, animal behavior is exceptionally adaptable and resilient in the face of environmental contingencies. Biological inspiration can help robots gain degree of adaptability and durability similar to animals. Biological inspiration may be sourced by various elements of animals, including their behavioral manners, physical and nervous system structure. In the 1950s, advances in technology resulted in constructing a diverse array of electromechanical devices meant to simulate biological processes and systems. Elsie and Elmer, W Gray Walter's robotic "tortoises" are perhaps the most well-known and immediately related to early bio-robotics [25]. His tortoises were tiny mobile robots encased in a tough shell. The robots were propelled by steerable motorized wheels and equipped with a headlamp, a light, and a touch sensor that reacted to contact with the shell. Between the mid-1980s and the mid-1990s, modern bio-inspired robots started to emerge. Rodney Brooks' [26] study on behavior-based robotics was a defining moment in this era. Although the work was not closely focused on biology, Brooks proposed that non-trivial and adaptive behavior might be formed in a robot via the relationship between basic control hardware and its environment, validating his thesis with robots performing tasks such as insect-like walking. Another contribution is the study on hopping and legged robots by Raibert [27]. It underlined the critical role of energetics in animals' dynamic balance and movement. Furthermore, in 1990 Arkin [28] created a schema-based control scheme for reactive robots. Hirose [29] produced a variety of snake-like motion patterns and manipulators based on research on the serpentine motion. Beer et al. [30] constructed several hexapod robots relying directly on the body structure and neurological control of walking cockroaches and stick insects.

Legged robots make one of the concentration areas in bio-inspired robotics. Depending on the application, robots can have one,[31],[32],[33] two,[34] four,[35] six,[36], [37] legs. Bipedal and quadrupedal robots are some of the most often employed structures of legged mobility in bioinspired robotics. In terms of speed, the two superior robots to date are the Rhex robot [36] and Cheetah [38].

This thesis is interested in mammal-like machines such as quadrupedal robots. Since mammals possess an incredible capacity to plan and execute complex behaviors suited for the environment and task at hand, they are an excellent inspiring point for robotics research. They can jump over a gap in the terrain or change their orientation rapidly in a case of perceived danger. While animals can navigate rugged terrains effectively, nature-inspired robots must plan and execute dynamic actions to negotiate challenging landscape. Legged robots are well-suited for utilization in hazardous natural scenes.

In this thesis, reference generation algorithms for the flying phase of a quadruped robot performing a long jump are presented. We first inspect the reinforcement learning (RL) technique for the posture recovery problem. Then an angular momentum approach is created for posture control and push recovery scenarios. Finally, a more comprehensive angular momentum-based algorithm with real time centroidal dynamics computation is proposed for posture control of quadrupedal robots during the flying phase of a long jump.

1.1 Motivation and Objectives

Legs provide substantial capability for crossing uneven terrain, especially steep obstacles or wide terrain gaps. Along with challenging terrain, robots may encounter unanticipated impact forces when executing leaping gaits. External disturbances include being pushed by other robots or live beings, colliding with trees or rocks, and being struck by objects. When conducting their gaits in an irregular natural environment or the absence of contact points, legged robots must effectively retain and reestablish their stability in the face of external effects. External disturbances should be detected quickly, and necessary actions should be taken to maintain the robot's balance and to create a posture suitable for landing. Hence, a posture control algorithm may significantly improve the functionality of a quadrupedal robot. Managing robot's balance and posture when free-flying or performing gaits with vitally stable sections is a challenging problem. Due to the absence of contact points, these scenarios require balance or posture control techniques.

1.2 Contributions

We propose novel reference generation approaches for in-air stabilization of robots that lack ground contact in the flight phase of a long jump. We concentrate on a quadruped robot with waist joints. These approaches are:

- A reference generation strategy based on RL for posture control,
- A reference generation strategy based on angular momentum for posture control,
- A reference generation strategy based on angular momentum for push recovery, and
- A reference generation strategy based on angular momentum with real time centroidal dynamics computation for posture control.

The RL-based approach generates reference curves for the waist joints of the quadrupedal robot in order to achieve the proper roll and pitch orientation in the air. Proper orientation is described as suitable to achieve final posture before landing.

The first angular momentum approach builds reference curves for the quadrupedal robot's waist and rear hip joints in order to obtain the appropriate roll pitch and yaw orientation in the air.

The angular momentum approach discussed secondly provides references for a small set of generalized coordinates and speeds that need another small set of coordinates and speeds (such as the orientation of the robot in flight) to reach a specified reference value swiftly and stably at the conclusion of the flight. We discretize and linearize the continuous time rate of change equation for angular momentum in order to provide a minimum set of linear equations for reference trajectory construction. Because the angular momentum equations are linearized, the reference generation approach may use an arbitrarily detailed robot model with no influence on computing performance.

These techniques are intrinsically more applicable than other prior models [21, 39] since they do not need the addition of a gyroscope or a tail to the robot. The robot model utilized in the computations is detailed, with each link of the robot, such as the legs and three torso pieces, represented as a separate mass. Additionally, these approaches for creating the trajectories are computationally efficient and can be employed in real-time applications. The proposed approaches are implemented in the simulation of a quadruped robot in the flight phase to validate posture recovery and push recovery methods.

1.3 Roadmap of this Dissertation

This introduction chapter discusses bioinspired robotics, legged robots, our motives and contribution goals. Past contributions are emphasized.

The following chapters discuss balance control for quadruped robots in detail and reference generation mechanisms for push and posture recovery operations.

Chapter 2 presents the background information and related work. This chapter is divided into three main sections. In the first section, machine learning strategies employed on legged robots are covered. This section is organized according to the different tasks achieved by learning techniques: Collision detection, terrain classification, path planning, motion planning, and posture recovery. The second section reviews the literature on balance and posture control strategies implemented on quadrupedal and bipedal robots. The last section introduces the angular momentum approaches employed in balance and posture control problems. This section of Chapter 2 is more comprehensive than previous ones since this field is closely related to the thesis. This section has three subsections: Bipedal robots, quadrupedal robots, and miscellaneous robots.

Chapter 3 describes articulated quadruped robot parameters. Simulation model of the quadruped robot, simulation and motion parameters and the control framework used in following chapters are described.

In Chapter 4, the learning-based posture control method for the quadruped robot during the flight phase of a long-jump is introduced. First, the methodology of this RL based reference generation approach is described. Secondly, the method is employed in a simulation environment, and results are presented. Lastly, simulation results and methodology are discussed.

Chapter 5 introduces the first reference generation method based on the angular momentum approach. This method is applied for posture recovery scenarios with and without the presence of external disturbance. The organization of this chapter is similar to the previous one. The methodology is described, simulations are performed with the quadruped robot during the in-air phase of a long jump, and methods and simulation results are discussed.

Chapter 6 is about a mathematically more complex angular momentum algorithm for posture control. This method is more detailed since there are fewer assumptions and a more detailed dynamics computation than in previous chapters. First, the methodology of this approach is introduced and application on a spined quadrupedal robot during the flight phase of a long-jump is discussed. Then, simulation results are presented. Lastly, simulation results and the performance of the algorithm are discussed.

Chapter 7 presents the overall conclusion of this dissertation, and possible future work is discussed.

Chapter 2

Background and Related Work

2.1 Machine Learning Strategies

2.1.1 Collision Detection

Machine learning algorithms provide an intuitive and generalizable approach for achieving a more stable motion for quadrupedal robots. Collision detection is one of the tasks that quadrupedal robots can perform using learning techniques. Collisions with the ground or self-collisions while conducting leg action are vital concerns in these scenarios. These impacts may cause the robot to lose balance and tumble, posing a chance of damage or significantly slowing the robot's progress toward the desired point. Some examples of machine learning approaches for achieving collision detection tasks are presented in this subsection.

Doshi et al. [40] offer a method for identifying collision-free swing-foot trajectories for walking gait rugged terrain. Rather than recreating swing trajectories and testing for collisions along with them, the method employs machine learning algorithms to predict if a swing trajectory is collision-free. Supervised learning is employed to develop a classifier to predict collisions utilizing terrain data collection.

Marco et al. [41] make three contributions to this field. They first provide a new Bayesian optimization framework with unknown constraints. This framework tackles the issue of learning with crash constraints by transferring the modeling effort associated with robot failures to the constraint and reducing failures during learning. Next, they offer a unique single-output Gaussian process for categorized regression with the constraint capable of handling hybrid data (discrete labels and actual values) and consider the constraint threshold as a hyperparameter, eliminating the requirement for expert knowledge.

Schperberg et al. [42] present an online path planning architecture in their study which expands the model predictive control (MPC) formulation to account for future position uncertainties. This allows for more secure navigation in congested situations. Their method incorporates an object detection process with a recurrent neural network that implies the covariance of state estimations at each step of the short time horizon of the MPC. The recurrent neural network algorithm is trained on robot and landmark positions are obtained by inertial odometry utilizing camera pictures and inertial measurement unit readings. They employ a trained deep learning algorithm combined with a feature extractor to recover the 3D centroid and radius bounds of surrounding obstacles to identify and extract their positions for avoidance.

2.1.2 Terrain Classification

Terrain classification is another task that quadrupedal robots can perform using learning techniques. Terrain categorization is critical for legged robot control since it enables the robots to adapt to the changing environment. This modification can be facilitated by providing data on the surface material or fundamental qualities such as stiffness. Specific gaits are better suited to different terrains. Therefore, a robot should be capable of switching gaits in response to a change in terrain. Some examples of machine learning approaches for achieving terrain classification tasks are presented in this subsection.

Degrave et al. [43] determine which sensors provide the most valuable data on the landscape. They experiment with various combinations of sensors often found on robots and assess the effectiveness of supervised or unsupervised machine learning approaches. They also determine which machine learning algorithms are most effective in classifying the landscape utilizing these sensors and which attributes are required for the techniques to operate.

In a different study [44], two different sensors are integrated to boost robustness. The oscillation power characteristic of the x, y, and z accelerometer dimensions provide higher-level data, while other aspects explain how much the paw sensors are stimulated by ground contact during movement. The constructed feature vectors are fed into machine learning algorithms and many classifiers are evaluated to determine which performs the best.

A difficulty in providing rapid posture adaptation across various terrains is the need for adaptive motor control. In order to overcome this problem, a distributed-force-feedback-based response with online learning is presented in [45]. It blends force sensor feedback, reflexes and learning to provide adaptive motor instructions in collaboration with central pattern generators (CPG). The learning approach is based on a primary neural network that utilizes a rapid dual integral learner's online modulation of plastic synapses.

Kim and Lee propose a quadruped robot gait adaptation strategy based on terrain classification and gait optimization for adaption on various surfaces [46]. A classification technique is employed to learn the adaption surfaces and a particle swarm optimization approach is utilized to optimize a locomotion parameter on each surface. After learning and optimizing, the classifier identifies the surface, and an optimum gait parameter is picked for adaption depending on the classification result.

Another solution to the terrain classification issue is proposed by Li et al. [47]. They develop a foothold determination model and establish its parameters by expert advising. The topographical elements affecting robot's stability are identified and then a foothold determination model is developed that takes both terrain attributes and kinematic constraints into account. A support vector machine is employed to learn model parameters, and the training data is stored in the simulation platform together with the topographical properties of candidate footholds and their rank orders.

Kolter et al. offer a hierarchical apprenticeship learning approach in [48], enabling the algorithm to receive isolated suggestions at various hierarchical levels of the control task. This method extends the applicability of the apprenticeship learning model to more challenging fields. This method is applied for quadruped mobility across rugged terrain.

Yao et al. [49] offer a hierarchical terrain-aware control system that combines deep reinforcement learning (DRL) and optimum control for high-level planners and low-level controllers. The global altitude map of the terrain provides visual information to the DRL, which utilizes it to identify the needed footholds for the robot's leg swings and body position. In order to maintain balance, optimal control determines the torque of the joints in the standing legs.

There are still difficulties in ensuring the motion stability of DRL-based robots' locomotion, particularly on rugged terrain. Achieving posture stability and agile movement pose challenges. In order to address this problem, Zhang et al. [50] propose a terrain-aware teacher-student regulator that incorporates a risk assessment network. During the training stage, the risk assessment network analyzes the risk associated with previous observations or the present state and guides the policy's update, supporting the policy in picking more appropriate actions and avoiding dangerous ones. The controller receives a real-time altitude map as visual input, allowing it to comprehend the terrain and generate higher-performance locomotion.

2.1.3 Path Planning

Path planning is the process through which a legged robot plans a lengthy path across recognized, potentially challenging terrain. Path planning is another task that quadrupedal robots can perform using machine learning techniques. One application area for quadrupedal robots is transportation over irregular terrain. This task needs path planning to maneuver the machine so that it takes a suitable course. In this section, some examples of studies concentrated on this particular task are presented.

Blum et al. [51] propose an RL approach based on a path planning and motion controller training algorithm. The algorithm teaches a simulated quadruped robot to respond to various commands and travel to designated areas using a combination of observable goals and goal randomization during training. A customized reward function is employed. They highlight two essential components of route planning and motion control, namely, region enabled travel and multi-point travel. Region enabled travel is the capacity to go to any site within a specified area. Multi-point travel is the ability to travel to many locations sequentially.

Medeiros et al. [52] discuss a method for guiding a quadruped robot in such a manner that it follows a course indicated by a collection of waypoints. Each leg of the robot contains three servo motors. Steering is accomplished by rotating the robot's frontal axis, sustaining its frontal legs, with an additional servo motor. The suggested route following algorithm begins by selecting a target point based on the present location of the robot and two touchpoints of interest. The robot's frontal axis orientation is then altered to point in the direction of the target location. A new reference point is chosen as the robot advances.

The primary objective of [53] is to demonstrate the application of a multiclass learning approach to challenges involving robotic grasping and quadruped locomotion. From a machine learning perspective, these tasks exhibit an unexpected fundamental similarity. In both situations, the desired policy entails complicated scoring functions, yet expert operators may deliver demonstrations of desirable behavior very inexpensively. Both issues offer many possible actions and may be easily improved to find the ideal option.

2.1.4 Motion Planning

Motion planning or gait planning is another task that quadruped robots can handle using machine learning techniques. Quadrupedal robots should perform navigation tasks like ascending stairs and traversing rough terrain. The fundamental robotic locomotion challenge is to determine the optimal sequence of movements for the robot to execute in order to reach its objective effectively. The motion planning approach focuses on this problem. Approaches to overcome this challenge involve changing between gaits, optimizing gait speeds or footholds and learning gaits autonomously from the start. Some examples of studies on this particular task are presented in this section.

Chernova and Veloso offer a method for walking gait optimization based on an evolutionary approach in [54]. The gait optimization technique makes no effort to estimate the gradient of the multidimensional space. This optimization method improves walking gait robustness to parameter assessment noise and prevents early converging to local optimum.

The purpose of [55] is to describe the planning and implementation of a coordination framework for quadruped walking robots capable of learning and performing soccer-playing actions. The author designs a hybrid method that combines reactive responses with deliberative thinking. Reactive behaviors translate location data acquired from sensors directly into actions. Fuzzy logic controllers are utilized for encoding adaptive behaviors to provide real-time and optimal control performance. A two-stage learning strategy is employed to keep the fuzzy logic regulators adaptable to complicated scenarios.

DeFazio and Zhang employ RL to quadruped locomotion in [56], addressing difficulties of learning efficiency and style definition for quadruped motion. The usage of incentive machines alleviates the problems. Reward machines provide task definition employing human-designed automata over linear temporal logic formulae, allowing the specification of a wide variety of locomotion behaviors. Additionally, reward machine methods have been created to maximize learning efficiency using available human knowledge.

Ishii et al. [1] design a simple quadruped robot with one degree of freedom (DOF) legs (Figure 2.1) and their goal is implementing different gaits to examine the implications of mechanical components to gait pattern transitions. They develop an experimental system capable of extracting and integrating movement-related feature quantities from sensor data collected online across several devices. They build a technology that allows them to obtain latent movement information from locomotion patterns using machine learning.

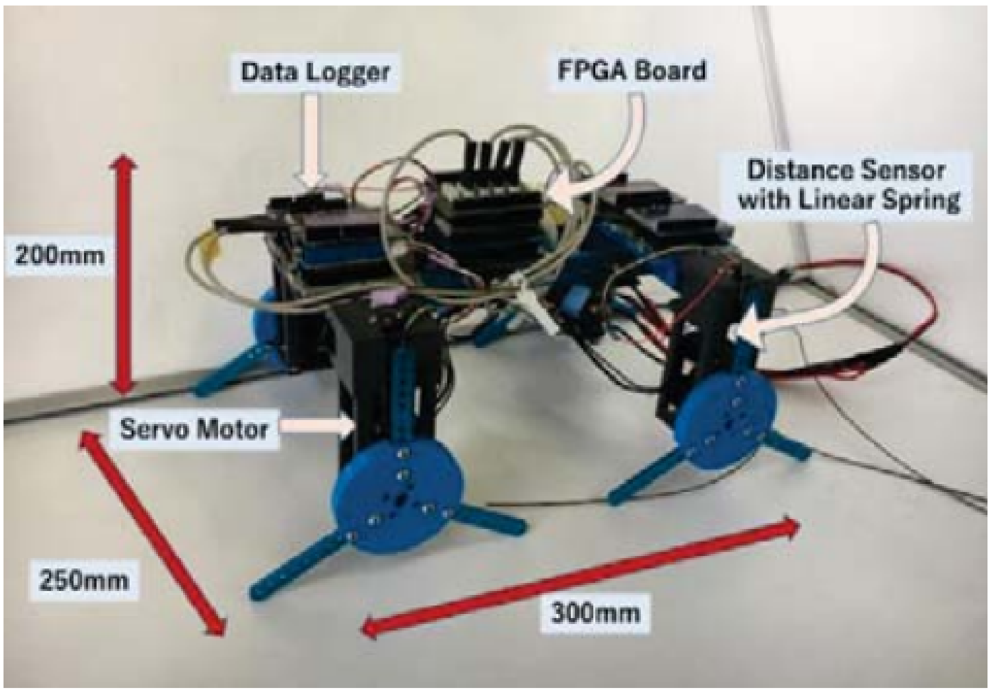

Figure 2.1: The quadruped robot with one DOF legs [1].

Two gait transition models for a four-legged robot are developed in [57] using gait kinematics. The training and generalization capabilities of a cerebellar design articulation regulator neural network are next investigated in learning gait transitions. Two gait transfer models (change between two periodic gait patterns and shift between periodic gait patterns) and a continuous follow the leader motion pattern are explored. These nonlinear models necessitate heuristic methods or parallel solution of many nonlinear equations.

Another study on gait transition is [58]. The purpose is to provide a novel intelligent control strategy for rapid four-legged robot bounding and galloping gaits. The controller can learn the leg touchdown orientations and foot thrusts necessary to achieve the appropriate sprinting height and speed in a single stride. The controller is trained using basic principles based on the heuristic understanding of quadruped kinematics.

Shao et al. provide a method for training a fundamental control strategy to enable a four-legged robot to perform various gaits in [59]. Two distinct stages are the bridge between the gait reference generator and the regulation policy that defines the quadrupedal movement. The quadruped robot, guided by these phases, displays locomotion by the created gaits, such as walking, trot, pace, bounding, and transitions between them.

Kalakrishnan et al. [60] demonstrate a control framework for high-speed quadruped motion across rugged terrain. They approach the task by breaking it down into numerous subsystems, each of which is subjected to learning, planning, optimization and regulation techniques in order to achieve robust, rapid locomotion. The control strategy is distinguished by several features, including a system that learns optimal foothold decisions from expert demonstrations using terrain layouts, a body reference optimizer depending on the zero moment point (ZMP) stability analysis, and a floating-base inverse dynamics regulator that, when combined with force control, enables reliable, compliant mobility over unanticipated obstacles.

The purpose of [61] is to present findings on the policy gradient method in training a stable, rapid gait. The experiments employ an objective function that prioritizes stability and head movements compensation. The policy gradient method discovers a stable stride with a negligible speed penalty in both situations.

Sato et al. [62] present a technique for CPG controller acquisition employing a mix of GA and RL. The three-step approach is capable of learning adaptive controllers. They demonstrate that the quadrupedal robot can adjust to new surroundings only via sensory input and oscillator connections. This finding implies that learning by trial and error can be conducted after identifying the right CPG coefficients. This is advantageous for adapting robots to new environments.

Tan et al. offer a method for automating the design of agile locomotion processes via DRL principles in [63]. The system can create the quadruped movement from scratch. Additionally, users may specify reference values to assist the learning process when additional control over the learned gait is required. Control rules are developed on a physics model and then implemented on actual robots. Simulation-trained rules often do not translate to the actual world. [63] closes this gap between reality and simulation by enhancing the simulator and developing solid regulations. They improve simulation quality by identifying the system, creating an accurate actuator model, and modeling delay. They develop resilient controllers by randomizing physical settings, introducing perturbations, and condensing the observation space.

Zhang et al. propose a neural network design for managing quadruped robots in [64]. The system is built of two networks: One for motion prediction and another for gating framework. The motion prediction system calculates the current frame's robot state at each stage using the previous frame's state and the user-supplied control signals. The gating system dynamically adjusts the scores of the motion prediction system by choosing and combining specialized networks. Due to the additional flexibility, the system may acquire consistent expert values for a wide variety of non-periodic/periodic actions using unstructured motion capture data. Additionally, users are relieved of executing sophisticated phase labeling in various gaits.

2.1.5 Posture Recovery

Posture recovery is the task of achieving the desired angular position of the robot body during performing a gait when the angular position of the robot body is different from the desired one. It is another task that legged robots can perform using learning techniques. Due to the lack of a ground connection, balance regulation during legged robot locomotion is crucial for navigating challenging static or dynamic environments. Since there is a lack of contact points for certain gait phases, such as in jumping, body orientation control is essential for motion stability. Posture control is the main issue handled in this thesis; some other examples of approaches below handle the same issue for quadrupedal or bipedal robots. However, they all have ground contacts and therefore they do not apply to long-jump tasks.

2.1.5.1 Quadrupedal Robots

Agrawal et al. offer a system in [65] centered on posture adjustment for stable quadruped motion across uneven terrain. Stability is achieved by using stable postures throughout gait transitions, chosen depending on the environment, foothold reachability, and gait sequence. They evaluate posture by utilizing value functions that mimic stability and kinematic factors. The value functions are created by learning using regression approaches, which reduce the need for extra sensors and computing for posture assessment.

In addition, Gay et al. [66] provide a framework for learning a model-free feedback controller for motion and balance regulation of a quadruped robot traversing challenging terrain. They employ a neural network to describe sensory input inside the CPG dynamics after designing an open-loop gait stored in a CPG. This neural network receives sensory input from a camera or a gyroscope and learns its weights unsupervised.

Kertesz and Turunen [2] construct a machine learning model that can distinguish four states of a Sony AIBO robot (Figure 2.2): Usual, pick-up, fall over, and pushed. A deep neural network classifier using these predictors achieves 98 percent accuracy on a new dataset. Actual trials on the robot demonstrate the practical usage of the classifier with real-time computational efficiency.

FIGURE 2.2: Sony AIBO robot [2].

The purpose of [67] is to examine the dynamic postural balance optimization and regulation of four-legged robots with compliant and flexible joints when external forces are perturbed. They begin by developing a limited dynamic model of joints for robots. They build a reduced-order dynamic model that considers the robot's relationship with the environment through numerous contacts. A dynamic force distribution approach based on a quadratic objective function is proposed to determine the optimal contact forces required to cope with an external wrench. A fuzzy logic based adaptive controller for compliant and flexible joints for four-legged robots is presented to suppress uncertainties in the robot's and actuator's dynamics. The proposed framework combines techniques of dynamic surface control and fuzzy learning algorithms.

Another solution for quadruped robot push recovery based on RL is proposed in [68]. This technique utilizes a reduced model of a quadruped robot to minimize the dimensionality of the action and state-space for the RL system and then improves the efficiency by employing the simplified robot model's previous knowledge. This technique may offer a foot placement estimation to the quadruped robot throughout the learning phase, allowing it to regain equilibrium while being pushed.

2.1.5.2 Bipedal Robots

Machine learning is a widely employed balance and push recovery strategy for bipedal robots. Ferigo et al. [69] use model-free DRL to learn a broad and resilient humanoid push-recovery policy in a simulation setting. The approach is verified on the iCub humanoid and aims for high-dimensional bipedal robot control. Fast learning of many robust actions by the same policy, encompassing the full-body, is possible with reward components that include expert knowledge on bipedal robot control.

Several bio-inspired approaches are modeled using the Dynamical Movement Primitives (DMP) [70]. The DMP, which comprises a series of differential equations, provides a unified parameterized representation for modeling a motion strategy, resulting in a strategy model that can be solved using machine learning techniques. The learning process for the DMP modeled bio-inspired methods is performed in this study by using stochastic policy gradient RL and imitation learning independently. Furthermore, using Gaussian process regression, impact recovery tactics may be developed using the DMP model's invariance qualities. As a result, an adaptive online push recovery control approach is realized.

Semwal and Nandi [71] collect humanoid push recovery data using an accelerometer sensor. Human push recovery by fusing data at the feature level using a physics toolbar accelerometer is studied through the experiments. The subjects for the experiments are selected as right handed and left handed. Pushes are induced from behind with closed eyes to observe the motor action and open eyes to observe learning-based reactive behaviors. A learning vector quantization-based classifier is developed to identify the coordination between various pushes and hip and knee joints. Semwal et al. utilize a deep neural network to categorize different types of pushes, and the total accuracy is 89.28 percent [72]. The first classifier is based on a feed-forward back-propagation neural network, whereas the other is based on a deep neural network. Small, medium, moderately high, and high pushes are used to test and assess the suggested deep neural network-based classifier.

Seo et al. [73] suggest that a push recovery controller is composed of an inertia measuring unit sensor component, a high-level recovery regulator, and a low-level recovery regulator. The linear velocity and angular velocity are measured by the inertia measurement unit sensor and sent to the high-level push recovery regulator. Based on the stability region, the high-level push recovery regulator determines the low-level push recovery regulator's strategy. The RL method's Deep Q-Network results in an increased the stability region. The low-level push recovery regulator involves the ankle, hip and step techniques. The actuators for each method examined by the linear inverted pendulum model (LIPM) are regulated based on the analysis.

A viable hierarchical push recovery approach applied to various bipedal robots is described by Yi et al. in [3]. The system comprises low-level controllers that conduct basic, biomechanically driven push recovery measures and a high-level regulator that integrates the low-level controllers based on sensory information and the present state of the robot. RL is employed to modify the control scheme to improve the robot's stability under various external effects. The controls are implemented and learned through a physical simulation on the Darwin-HP bipedal robot platform (Figure 2.3) In a different study, Yi et al. describe an alternate method for stabilizing the walking of a position-controlled bipedal robot challenging terrain [74]. They assess the inclination of the surface utilizing swing foot dynamics and sensors and then train an adaptive terrain model employing an online learning method. A hierarchical push recovery regulator rejects perturbations caused externally and modeling errors by modulating three biomechanically driven push recovery regulators per current estimated condition. They also focus on learning in a high-level push recovery technique, employing simulated robot models with various degrees of complexity and then a real robot [75]. They discover a shared low-dimensional technique for high-level push recovery from state trajectory data gathered using several models and a real robot. Push recovery can be efficiently learned online from a limited range of experiments on a real robot.

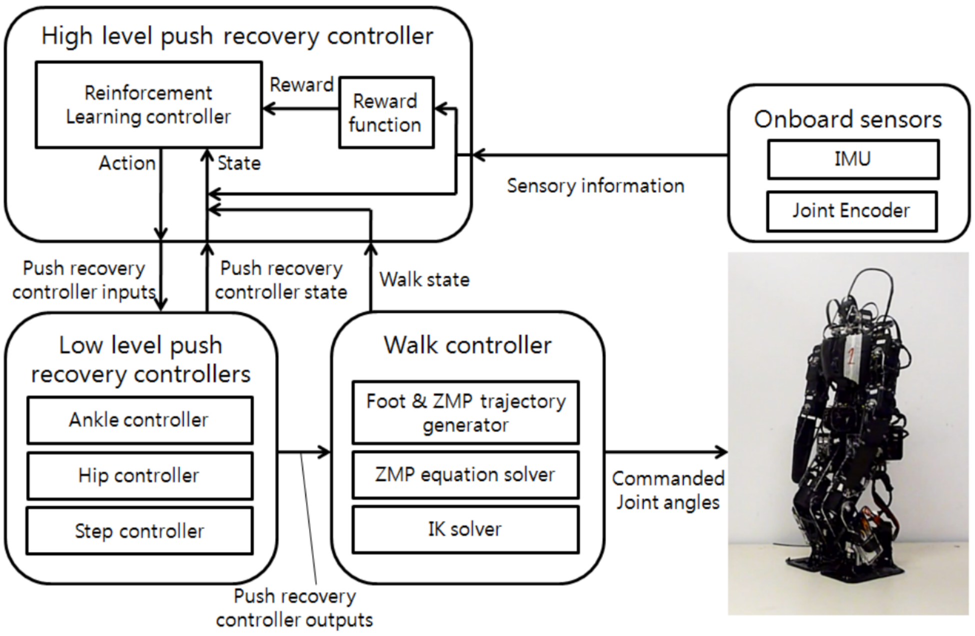
High level push recovery controller
Reinforcement Learning controller
Reward
Reward function
Action
State
Onboard sensors
IMU
Joint Encoder
Sensory information
Push recovery controller inputs
Push recovery controller state
Walk state
Low level push recovery controllers
Ankle controller
Hip controller
Step controller
Walk controller
Foot & ZMP trajectory generator
ZMP equation solver
IK solver
Commanded Joint angles
Push recovery controller outputs

FIGURE 2.3: The hierarchical push recovery controller scheme [3].

2.2 Balance and Posture Control Strategies

Legged robots need balance and posture control strategies while performing dynamic gaits or being subjected to external disturbances such as an external push.

There are studies on posture control and dynamic gaits with ground contact [4, 5, 76, 77, 78]. They do not apply however to long-jump tasks.

2.2.1 Quadrupedal Robots

A reference trajectory generating approach for a quadruped robot for pacing gait on a horizontal plane is presented in [76]. The method is based on the LIPM and the ZMP stability criterion. It presents ZMP reference trajectories for pace employed for the generation of quadruped robot center of mass (COM) references. Preview control is applied. Inverse kinematics is used to calculate reference positions of leg joints with the COM trajectory.

Asadi et al. [77] present a Cartesian CPG-based regulator for gait creation and transition between gaits. The suggested method generates rhythmic signals using nonlinear linked oscillators . A programmable excitation signal can change these patterns by correspondingly altering the frequency, magnitude, and coupling variables among oscillators. The length of the step, frequency of swing and stance periods of all legs, relative timing between the legs, and order of leg lifts may all be modified smoothly based on these differences, notably the length of gait transition. Then, using the data from the CPG controller, trajectories of swing legs along the x and z-axes are determined. The footprints of legs are used to design the reference ZMP route. After that, the center of gravity (COG) trajectories are determined via a preview servo controller.

Di Carlo et al. [4] demonstrate how MPC can calculate ground reaction forces (GRF) for quadrupedal locomotion. The robot dynamics of Cheetah (Figure 2.4) are reduced to be formulated as a convex optimization problem while still capturing the system's complete 3D characteristics. Reaction force planning issues are created and solved utilizing the simplified model. The quadruped robot is capable of robust movement at various speeds despite having a simpler model.

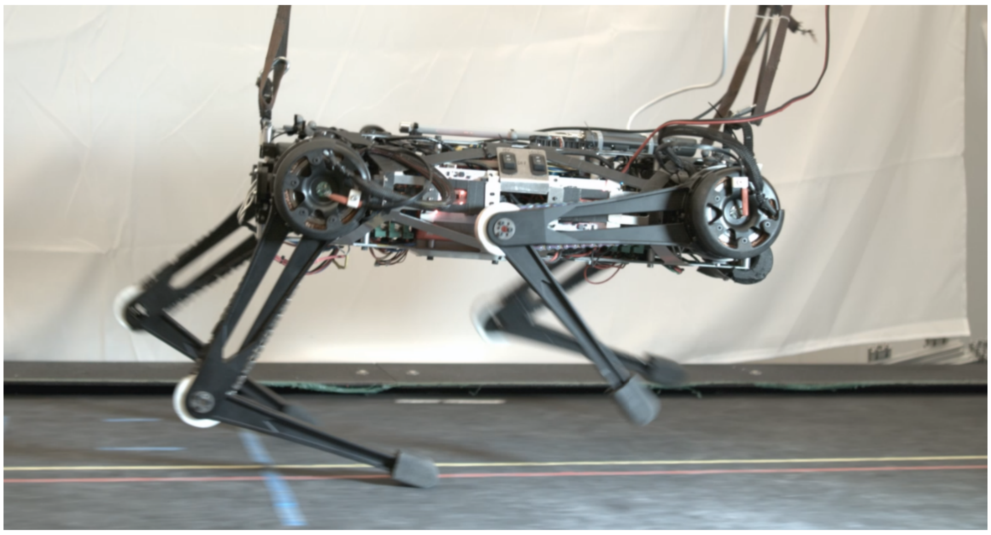
FIGURE 2.4: The MIT Cheetah 3 quadruped robot [4].

In [78], it is investigated how to generalize algorithms that provide balance for one leg to operate machines with many legs. When multilegged systems operate with gaits that employ supporting legs one at a time, the generalization is relatively straightforward. Multilegged running may be controlled using one-leg algorithms for specific gaits. In order to further expand the method to gaits that employ the legs in pairs (pace, bound, and trot), the idea of a virtual leg is presented.

A trajectory generation method and an active compliance control technique for a hydraulically actuated quadruped robot HyQ (Figure 2.5) is proposed by Ugurlu et. al. [5]. Techniques are combined in a framework to generate trot-walking motion cycles. First, a center of pressure (COP) based trajectory generator generates viable and balanced motion trajectories. For symmetrical locomotion patterns, initial conditions are set individually, ensuring that references are smoothly coupled in position, velocity, and acceleration, independent of the support phase. The active compliance regulator, which is utilized in tandem, ensures adequate joint displacement and force regulation.

Some of the studies that are focusing on posture control under external disturbances are presented below.

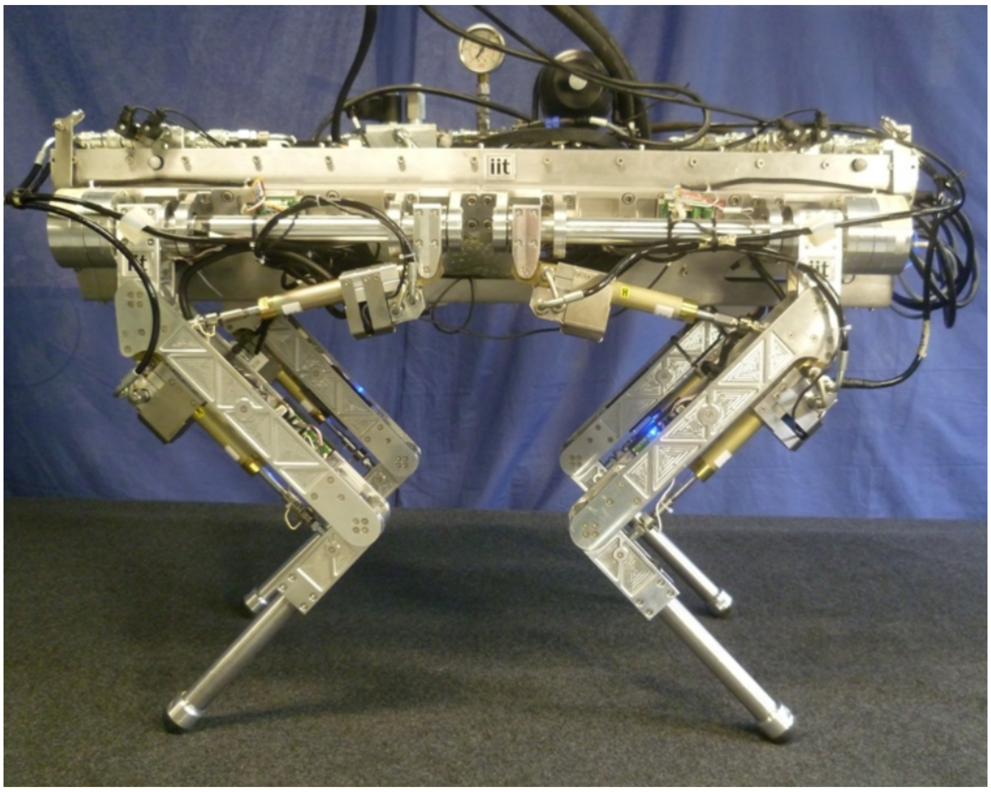

FIGURE 2.5: The quadruped robot HyQ [5].

The primary/secondary gait method is offered as a mechanism of gait creation for both straight and circular body trajectories [79]. The main gait is a set of leg transfers with adjusted foot kinematic constraints in response to obstacles, whereas the secondary gait is a dynamic gait created to change the foot position. The main gait is created by considering the following constraints: Stability, kinematics, sequential, and surrounding constraints. The impact of the impediment on primary gait metrics changes them. Regardless of the motion mode, all limitations and obstacle effects are stated by a single set of equations. The suggested technique allows for the efficient generation of free gait while considering body trajectory planning.

A flying trot gait control system is developed to increase the quadruped robot's dynamic motion capacity in [80]. The robot can accomplish a steady flying trot movement by planning the motion of the torso and transferring it to the feet. In order to achieve stability under external disturbances, the motion controller is employed based on the spring-loaded inverted pendulum concept. The suggested approach can shift between trot and flying trot gaits, with the flying trot gait improving the speed of the robot and the trot gait providing robustness.

A push recovery framework is presented in [81] to recover the stability of the robot in the face of unknown external perturbations. The entire body dynamic model is employed for calculations in order to improve the robot's ability to restore its posture by utilizing all of the DOF. The posture controller is designed to calculate the main body's proper acceleration. After the disturbance, it is estimated to restore the robot to a target location. Regarding the stability and friction parameters, desired acceleration is chosen. An optimization model is designed to determine body acceleration, with stability and friction as constraints.

In another study, Khorram and Moosavian present a balance controller for a quadruped robot that will regain its posture stability in the face of external forces [82]. Posture stability is accomplished by creating a model of the quadruped robot traveling across both flat and irregular terrains. Proportional-derivative (PD) control is suggested to calculate the appropriate accelerations to bring the robot back into balance. A disadvantage of the method is that, these accelerations can cause the robot to become unstable and cause the stance feet to slip. As a result, the maximum acceptable accelerations are computed using an optimization procedure. The requirements that assure the robot's stability and prevent stance foot slipping are the restrictions of the optimization issue. The preferred optimization technique is a real-time linear least-squares based method.

2.2.2 Bipedal Robots

One of the balancing strategies employed on bipedal robots is hip and ankle approach. A study that classifies different sorts of pushes introduces a push detector and provides control techniques for dealing with constant powerful pushes that occur while walking gait is described in [83]. Due to biological and biomechanical studies, more efficient procedures, such as a hip and ankle approach or a bent-knee strategy, can be adapted to biped control systems. A push recovery regulator is suggested, generating required torques and regulating joint positions to regulate a bipedal robot.

A control scheme and a push recovery controller for a bipedal robot walking are developed in [84]. The bipedal robot and the algorithm are subjected to various types of pushes in order to modify the walking stage and maintain walking in a simulation environment. This work discusses feature selection strategies for predicting hip, ankle, and knee joint push recovery. The algorithm is trained to utilize K-Mean classification and crouch data.

Stephens utilizes basic humanoid-specific models to large-scale push recovery [6]. The author builds analytic decision surfaces, which are functions of reference points like the COM and COP, that forecast whether or not a fall is unavoidable. Three recovery strategies, namely ankle torques, internal joint movement, and taking a step are explored (Figure 2.6).

In [85], a three-mass inverted pendulum model (Humanoid Open Architecture Platform) is constructed, and the model's efficacy is justified using the a simulation

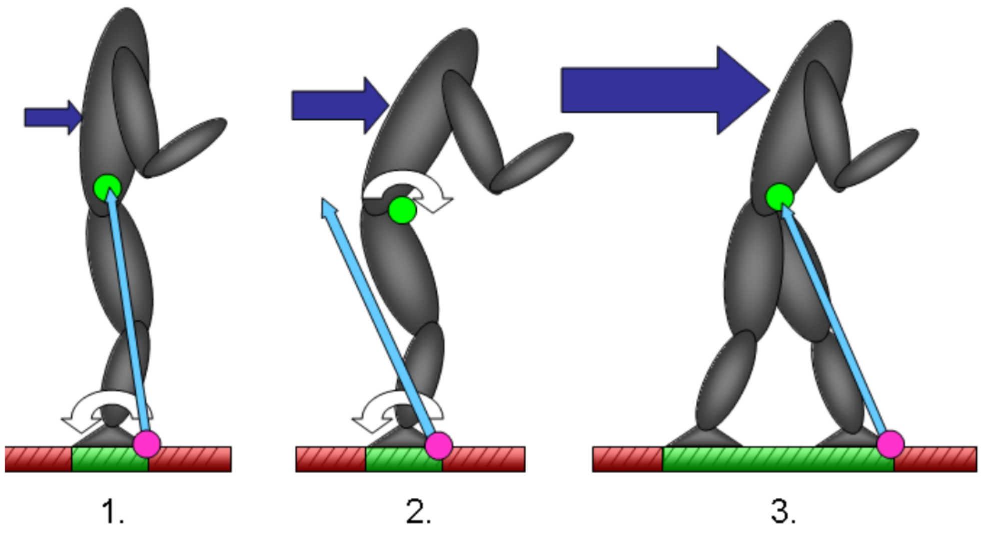

FIGURE 2.6: The three balance techniques [6]. The COM is represented by the green dot, the COP by the magenta dot, and the GRF by the blue arrow. 1. Balancing the COP (”Ankle Strategy”) 2. CMP Balancing (also known as the ”Hip Strategy”) 3. Take a step outside.

environment. It is created as a testbed for advanced push recovery procedures for the Humanoid Open Architecture Platform robot. This model’s functionality is broad and extendable to any three mass models. In order to confirm that the proposed model is valid, the ankle strategy approach was employed.

Kamioka et al. [86] present a strategy for re-planning steps and time and locomotion mode, including walking, running, and hopping. The locomotion mode re-planning technique uses parallel computation and a rating system with a new cost function. Push recovery studies are conducted in order to validate the strategy, which involved pushing in the forward and lateral directions in walking gait.

A humanoid walking trajectory generation technique with push recovery that can be planned online is described by Shafiee et al. in [87]. The suggested approach is particularly well suited to control systems in which the Divergent-Component-of-Motion is pre-planned. It includes a step adapter to change the planned trajectories and ensure push recovery. The step adapter creates new coordinates and timings for the next step assuming that the bipedal robot is in a single support state. The step adapter is active during single support phases. However, the suggested torque control system considers single and double support phases. The concept behind the step adapter's design is to use an exponential interpolation of the time-variable ZMP trajectory to establish both final and initial Divergent-Component-of-Motion step values. This method enables the push recovery task to be recast like a Quadratic Programming problem, which may be solved online using optimizers.

The planning of stepping movements is extended to bipedal robots having force controlled limbs (such as the Sarcos humanoid robot (Figure 2.7)) in [7]. Push Recovery MPC is a linear MPC that does step planning. A basic model and a specified objective function and constraints are utilized to obtain this model. Force control is employed to add feed-forward joint torques.

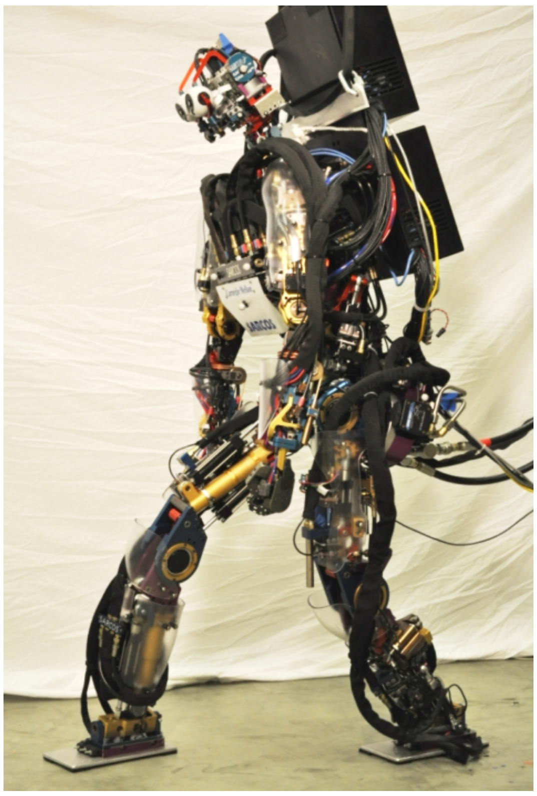

FIGURE 2.7: The humanoid robot Sarcos [7].

Urata et al. present a novel online walking trajectory development approach in [88]. In order to adjust the placements of foot-steps sequentially, this approach utilizes a limited version of the preview controller. Foot-step generation to alter walking position and velocity with a slight delay, push recovery under undetermined external force, and incorporation of full-body dynamics bipedal robot model in trajectory development are among advantages of this technology.

A fuzzy dynamic gait pattern generator that enables a teen-sized bipedal robot to develop an appropriate gait pattern in real-time when impacted by an unanticipated force is presented in [8] by Wu and Li. The ideal ZMP is used to design the COM's trajectory in traditional gait pattern generators, and a cycloid creates steps. On the other hand, pre-planned gait patterns cannot deal with unforeseen conditions, particularly when the robot is confronted with an unknown force. As a result, they develop a dynamic gait pattern generator that uses the Virtual Force LIPM to alter the COM's trajectory and detects balance by calculating the ZMP's trajectory employing eight high-precision load cell pressure sensors attached to the robot's soles. They use a fuzzy controller with an accelerometer and pressure sensors to respond immediately to environmental influences and develop an appropriate gait pattern. When the robot is unexpectedly pushed, it replaces its present walk with a pre-planned gait pattern. The fuzzy controller generates the recovery gait simultaneously, with proper strides and inclination angles to soften the impact. The proposed approach is tested on David Junior II (Figure 2.8), a teen-sized bipedal robot.

Another balance control strategy for a biped robot is proposed in [89]. This work involves adjusting COM position with the assistance of the correct foot positioning technique in response to external disturbance. The proposed method detects the robot's current posture and stability state using sensory information and recovers the robot's posture and balance in the event of an external push by placing the robot's feet in the proper direction. The foot placement is done using one of the alternate or rescue trajectories previously stored in its memory. Sagittal and lateral recovery trajectories are designed for transitions from the present trajectory to the recovery trajectory.

A balancing control strategy is described in [90] for compliantly adjusting the COM location and torso orientation of a bipedal robot. When an unknown external

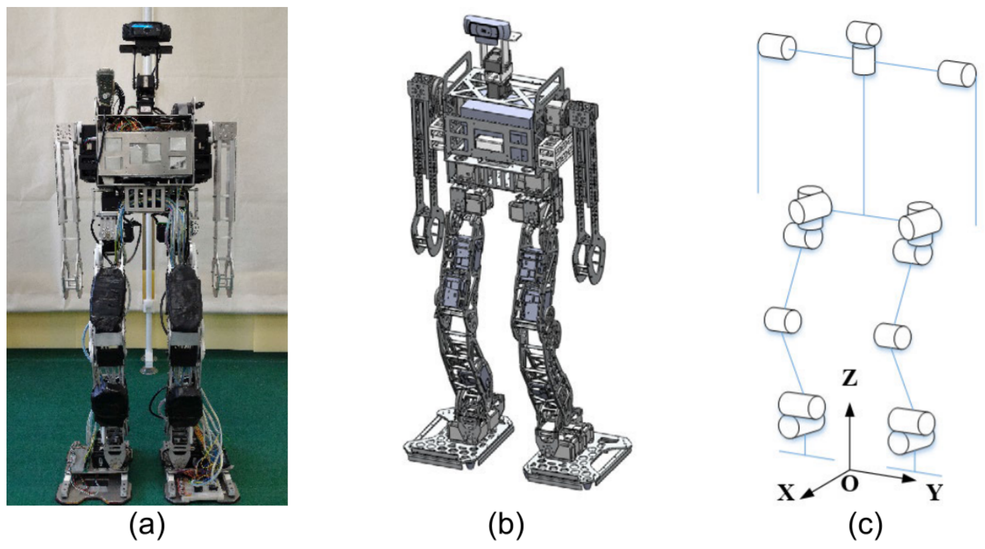

FIGURE 2.8: Humanoid robot David Junior II [8]. (a) An accurate representation of the robot. (b) The structure of the robot. c) The arrangement of the joints.

perturbation disturbs the robot's posture, the controller that maintains both the position and orientation calculates the desired force and torque required to restore the posture. This force and torque are then applied at predetermined contact sites using a constrained optimization algorithm that aims to achieve the desired force and torque while reducing contact forces.

The capture point (CP) idea is used to create and propose a control strategy [9]. Rather than depending on position control, as most CP strategies do, the proposed strategy produces references for the iCub (Figure 2.9) robot's momentum based torque controller, extending its ability to respond to external perturbations while preserving the benefits of torque regulation when engaging with the environment.

Hodgins and Raibert describe a control algorithm to make a bipedal robot do forward flips [91]. The approach is based on several steps. First, they adjust the pitch rate to maximize flight time. In order to convert horizontal speed to vertical speed, legs are extended forward. They modify the initial pitch torque for take-off. Legs are shortened after take-off to increase pitch rate. Reducing energy in the landing phase is achieved by reducing thrust and using an attitude control algorithm. The method is nature-inspired, and mentioned steps are based on human knowledge of the mechanism of the flip movement.

Figure 2.9: iCub humanoid robot [9].

The virtual leg model is provided as the essential contribution in [10] for recovering the NAO humanoid robot (Figure 2.10) from external force effects. The control purpose is to absorb the external force and restore the robot's original configuration. A PD controller is employed for joint torque control to fulfill the control goal. Webots validates the performance of these methods and the model. In each case study, an impulsive force of varying magnitudes is delivered on NAO's torso.

Finally, a novel fuzzy logic-based regulator for impact recovery is introduced [92]. The fuzzy inference system requires two inputs (the force and the direction of the motion), which are fuzzified before being subjected to a series of rules, after which

FIGURE 2.10: NAO bipedal robot [10].

the result is defuzzified and converted back to a crisp value. They utilize fuzzy rules to recreate the model in an unknown environment. The fuzzy logic-based controller can forecast if the robot will bounce back or fall and implements the required push recovery technique. The architecture is organized hierarchically. The first fuzzy inference system detects small, medium, and large forces in roll and pitch effects on the body. These are the input parameters used by the second fuzzy inference system to decide upon the push recovery approach that will be employed, and the robot will be able to recover from a push or a fall.

2.3 Balance and Posture Control Strategies with Angular Momentum Approach

Angular momentum is an important variable in balance and posture stability analysis and control for legged robots. Angular-momentum-based approaches are also implemented in this thesis for posture control and push recovery purposes. Some works in the literature utilize the angular momentum approach for balance and posture control. Some of them are listed in this section. However, they do not apply to long-jump scenarios (the main problem in this thesis) where there is no ground contact.

2.3.1 Quadrupedal robots

SCOUT, a basic mechanical design for a quadruped robot with only one DOF per leg, is suggested in [93]. Despite its mechanical simplicity, the initial prototype SCOUT-1 can walk, turn, and climb a step. The dynamic operation focuses on controlled momentum transfer is the basic principle. A momentum transfer occurs when a leg contacts the ground, leading to step changes in linear and angular velocities. The idea of conservation of angular momentum concerning the impacted toe is used to determine these changes.

Chung et al. [94] provide a posture stabilization approach for creating a steady trot gait. A foot positioning approach is designed to create steady locomotion. The step planning technique uses conservation of the angular momentum to attain the postural stability of the quadruped robot. The trotting gait can be regarded as a virtual humanoid gait since the diagonal legs contact and depart simultaneously. The stepping point that provides the robot's stability is found by employing a dynamic model of a humanoid gait.

Mita and Ikeda [95] introduce the notion of variable constraint control and demonstrate how to implement it. The main contributions of their work are the expression of constraints with equations of motion and integrating a differential equation to develop the control methodology that embodies these limitations as decoupled motions. They propose that the movements of the COG of the overall robot and angular momentum around COG, and the position of the foot provide the objective restrictions in the touchdown phase of gaits such as running and jumping.

Ugurlu et al. [11] propose a control system for quadruped robot RoboCat-1 (Figure 2.11) leaping and trotting over challenging terrain that is actively compliant and balanced. Two control schemes are created in order to demonstrate such movement abilities: active compliance control (force feedback) and angular momentum control (gyro sensing). Using Jacobian and admittance blocks, the first regulator yields the joint displacements due to the error of GRF as an output. These joint displacements are inputs of the second controller (local servo controller), together with position restrictions, allowing the robot to complete the assigned motion in an actively compliant way.

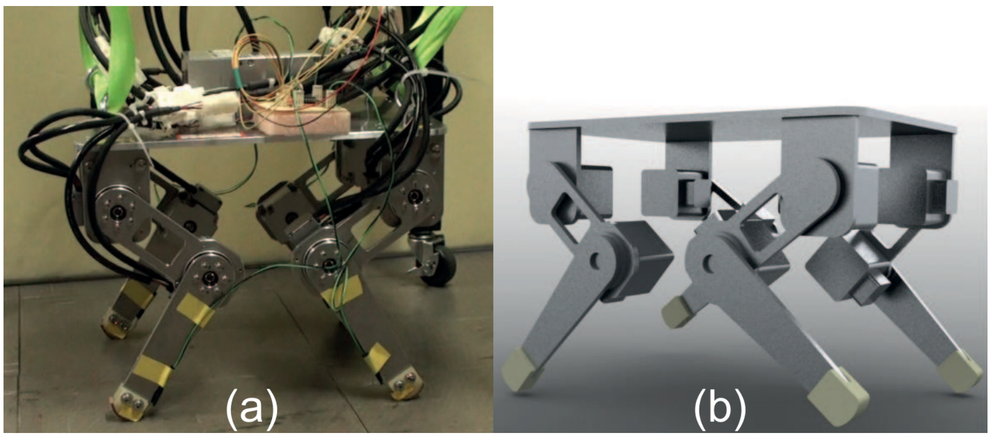

FIGURE 2.11: a) Quadruped robot RoboCat-1. b) CAD model of RoboCat-1 [11].

Control algorithms for a quadruped standing leap over uneven topographical obstacles are explored in [96], [97]. Simple open loop leg forces are utilized to eliminate the robot's significant linear and angular momentum upon landing. In order to choose foot touchdown angles, real time simulation is performed. Landing parameters are estimated based on the simulation employing a simplified quadrupedal robot model. Leg forces at take-off are calculated by the symmetry principle and compared to those expected during landing.

The balance recovery of quadruped robots performing trotting gait is the subject of [12]. Because of the dynamical nature of this gait, the robot is treated as a humanoid robot, with two cross legs of the robot represented as a virtual leg (Figure 2.12). The virtual model is simplified to a 2-D LIPM with a flywheel. Furthermore, by constructing a two-dimensional CP estimator, the required positions for the COP of the legs for regaining the desired posture of the robot are computed using the idea of CP for the two-legged model. The rotating motion of the two-legged robot's body and non-contact limbs is employed to produce angular momentum around the COM of the robot. The use of an MPC changes footstep positions and produces a new walking gait pattern. Modified joint positions of the virtual humanoid robot model are then converted back to joint angle values of the actual quadruped robot model.

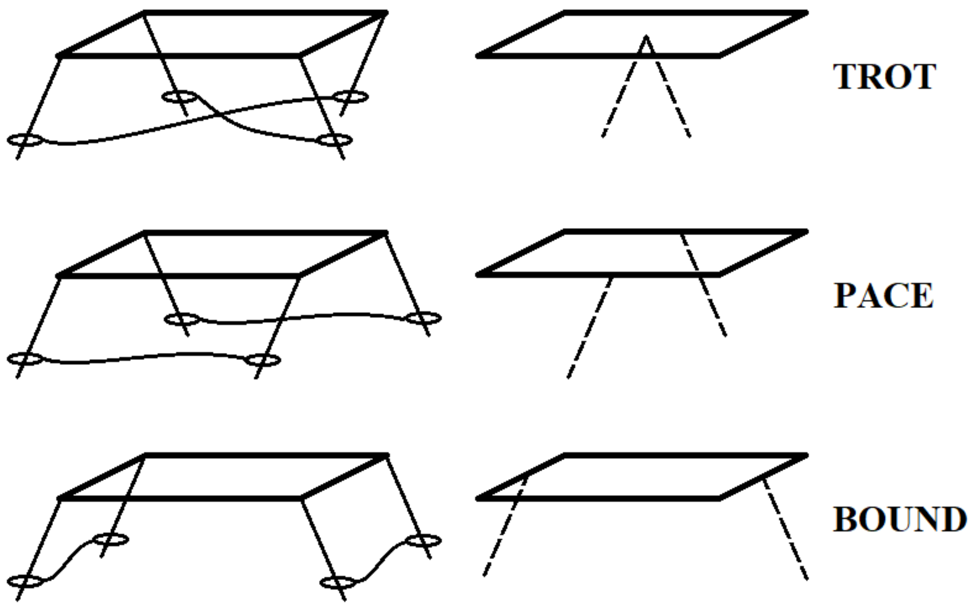

FIGURE 2.12: The three distinct dynamic gaits (trot, pace, bound) utilize pairs of legs and their virtual leg counterparts [12].

Lee et al. [98] offer a closed-loop full-body controller that regulates angular momentum about COM for quadruped robots. The joint torque caused by contact with the ground is calculated using the torque sensor data and the inverse dynamics equations. They propose a technique and full-body control criteria for gaits that employ two or fewer legs in contact with the ground. A centroidal moment pivot trajectory is developed based on this technique, which includes the angular momentum rate of change translated from the observed joint torque to maintain the balance of the robot. A push recovery approach that relies on CP dynamics synthesized of a mechanism for foothold generation and linear momentum is also applied by the controller.

A foot placement estimator is presented in [99] to assist a quadrupedal robot to regain stability. The robot can stabilize its posture by taking extra steps. The approach provides a practical strategy to calculate step positions for the quadruped robot utilizing angular momentum.

2.3.2 Bipedal Robots

ZMP is one of the tools widely employed in literature for posture control and push recovery purposes. A capturability-based walking stability control is presented in [100]. Moving ZMP in the support polygon, landing location adjustment, landing time modification, angular momentum control, and fall detection and control are the five tactics included in the suggested approach. The ZMP is calculated so that after the double support phase, the CP approaches the location of the supporting foot. In order to avoid falling, the torque around the COG is generated using body inverse kinematics with angular momentum restrictions.

Luo et al. [13] suggest feedback and feed-forward controls to create a bipedal robot walking trajectory generator based on a five-mass with angular momentum model for Renbo humanoid robot (Figure 2.13). This method intends to reduce modeling error and improve frequency characteristics resulting from nonminimum phase nature. In order to decrease modeling error and improve walking performance, the suggested method concentrates on angular momentum contributions from arm and leg rotation. It can counteract abrupt change of the natural ZMP reference related to frequency properties in the nonminimum phase control system with pole-zero cancelation and a series approximation approach. Based on the suggested model, a humanoid robot can verify and demonstrate the bipedal robot walking pattern generator. In an earlier study, Luo et al. [101] also present a method for humanoid robots for disturbance rejection and push recovery. Maintaining walking gait stability and preventing falling are the main objectives. The approach they employ to manipulate ZMP uses the angular momentum created by reaction mass. The torque concentration among the response mass is calculated in a biomimetic manner to cope with disturbance. When external disturbances are eliminated, the balance recovery process starts with online adjustment of the walking gait pattern, which involves angular momentum for adaptive trajectory development.

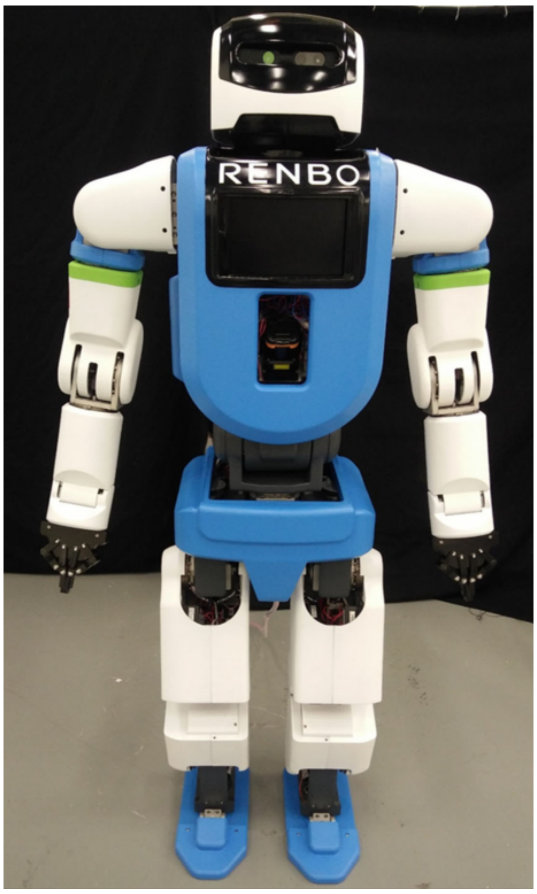

Figure 2.13: RENBO humanoid robot [13].

The stability control of a humanoid robot is demonstrated utilizing three mass modeling (Figure 2.14) and the ZMP criteria [14]. Because predictive control strategy and real-time implementation are harder to achieve, a predictive proportional-integral-derivative (PID) controller is proposed for tracking references. In order to decrease modeling mistakes, the system is simulated using three mass points (the torso's, the right leg's and left leg's COMs) and angular momenta.

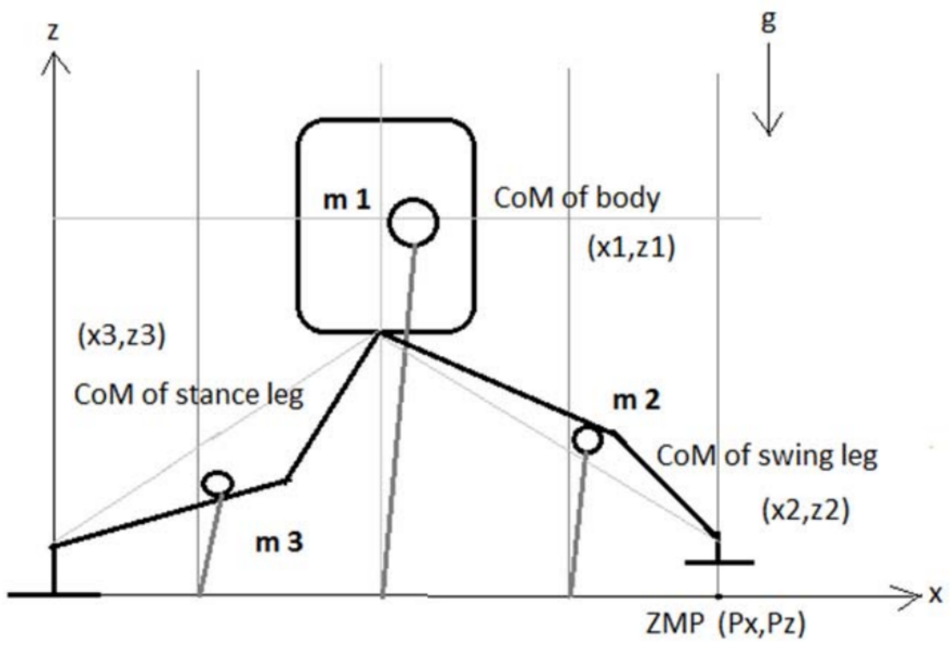

FIGURE 2.14: Three-mass model of the bipedal robot in the sagittal plane [14].

An MPC and CP-based push recovery algorithm is proposed in [102]. The ankle, hip, and stepping techniques are three approaches that are employed for balance improvement. There are various situations in which a humanoid robot cannot step. In this case, regaining equilibrium by regulating the torso's angular momentum or the ZMP is critical. While modulating the centroidal moment pivot and the ZMP, a single MPC technique guides the CP to the desired location. As a result, the purpose of the suggested algorithm is to control the CP using the centroidal moment pivot while the CP is outside of the support polygon and the ZMP when the CP is within.

[15] aims to offer a method for creating viable and dynamically stabilized ZMP-based COM trajectories for bipedal robots. The ZMP approach in the spherical coordinate frame is employed in order to fully leverage its features since this method allows to efficiently mix internal angular momentum rate change components with inertial force terms. Bipedal walking tests on the humanoid robot MARI-3 (Figure 2.15) are conducted in order to test the proposed method. Consequently, they created repeated, continuous, and dynamically stabilized walking cycles with minimal

unfavorable torso angles. Furthermore, since the robot's inertial characteristics are known, the ZMP error diminishes.

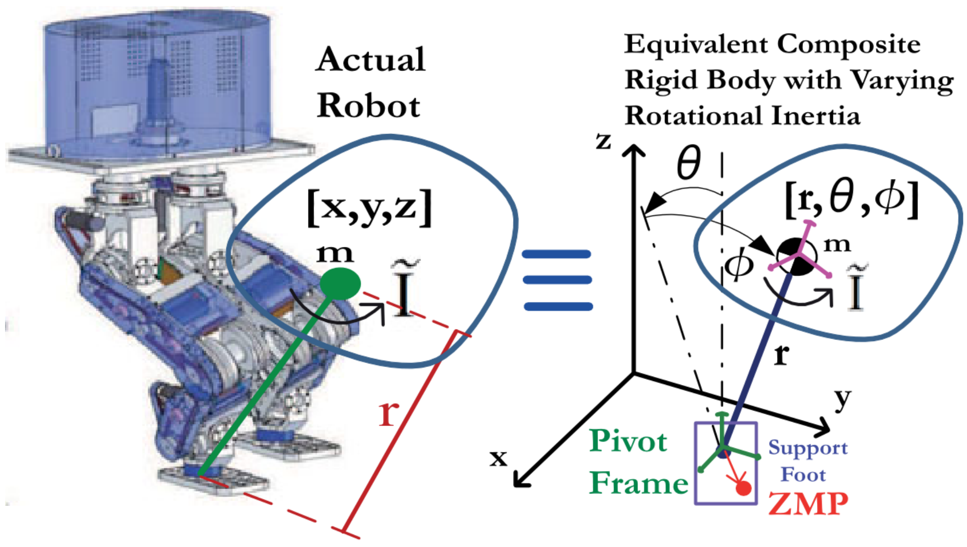

FIGURE 2.15: Bipedal robot model and spherical coordinates of MARI-3 [15].

Luo and Chatila [103] present a five-mass angular momentum approach to model a bipedal robot to decrease modeling errors and increase ZMP stability. In order to assess the collision stage, they create the safe bound investigation and a state estimator for the COM. In order to ensure a stable walking gait, the safe bound test is utilized. A state estimator for the COM is employed to assess whether or not the external disturbance is complete.

Various pendulum models are used to simplify the bipedal robot models and represent the robot as a point mass. Simulations of a basic planar biped robot walking to the capture region and employing angular momentum to regain stability following a push are presented in [104]. This work proposes a computing method for CPs and the capture region, the area on the ground where a bipedal robot must walk to come to a complete halt. The intersection of the capture region and the support base determine which method the robot should use to stop in a particular scenario effectively. The calculation of the capture region is achieved by using simpler walking models. They explain a method to compute accurate capture region solutions for this model by extending the LIPM to incorporate a flywheel body.

Rebula et al. [105] propose a method for generating CP by learning offsets from the CP calculated by the LIPM, which models a humanoid robot as a point mass with a constant height. A 3D bipedal robot simulation with 12 lower body DOF, distributed mass, and articulated joints is used to test this approach. Compared to employing the LIPM without a learning algorithm, resilience to disturbances is enhanced when utilizing the learning algorithm.

The LIPM with angular momentum is employed in [106] to create a posture recovery approach based on stepping strategy (hip and ankle strategy). In the stepping strategy, the duration and number of each step are significant. In addition, a stepping-out approach is offered for the posture recovery process, which brings vertical COM motion into consideration.

Whitman et al. [107] propose a LIPM dynamics-based modification for a wide range of controllers. A simple model and a change of variables are used to leverage the connection between COM and angular momentum dynamics. In order to create complete body torques, appropriate COM and torso accelerations are used. A change of variables to control COM unaffected by upper body angular accelerations is employed. They employ upper body rotation and COP modulation as alternative tools of the controller, allowing us govern with both upper body rotation and COP modulation. Simulated stance and walking studies show that enhanced control authority improves resilience to external forces.

The reaction mass pendulum model, a 3D version of the well-known reaction wheel pendulum, is introduced in [108]. The response mass pendulum approach improves the previous models by compactly capturing the robot's centroidal angular momentum through its rigid body inertia. This model presents an overview of legged robot dynamics, particularly rotational motions, leading to a straightforward set of control rules.

Kasaei et al. [109] offer a walking gait trajectory generation technique that considers push recovery. The algorithm is organized in a hierarchical framework that attempts to conceal the intricacies of dynamic walking motion. They improve the LIPM-Plus-Flywheel by considering the angular momentum around the COM to alleviate the COM's height limitation. This improvement creates room for a more natural motion and more steady walking motion.

The method of angular momentum control and the inverted pendulum model based on the virtual mass ellipsoid (Figure 2.16) to recover from orientational push are suggested in [16]. These approaches have three primary characteristics: They can recover push during walking gait, are suited for irregular terrains and eliminate the constant COM height and constant centroidal angular momentum limitations.

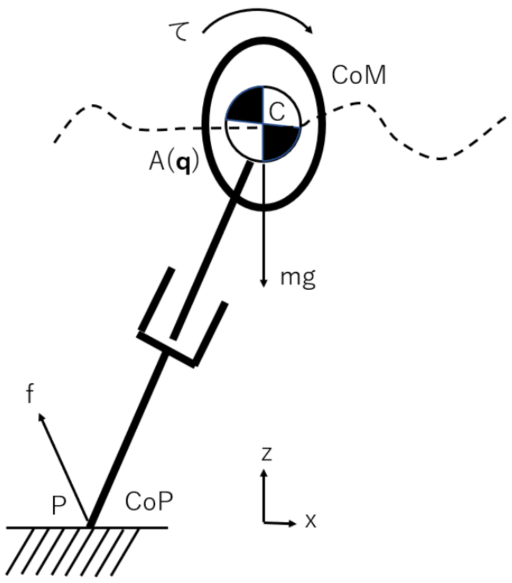

FIGURE 2.16: Inverted pendulum model with virtual-mass-ellipsoid [16].

A middle ground between a complicated, whole body dynamic model comprising every connection and actuator of the robot and a substantially simplified representation of the robot as a point mass is investigated in [110]. While the fundamental dynamics of bipedal robots are complex, the development of angular momentum and the COM position is significantly more straightforward. They arrive at an approach with simpler dynamics while still having the extensibility required to handle kinematic constraints. Robot COM and angular momentum estimated from joint trajectories to match the centroidal dynamics is also required for the algorithm.

A high-dimensional model of the humanoid robot (Figure 2.17) may be simplified during a turn to a lower number of primary components that contribute to the spin angular momentum about the COM is demonstrated in [17]. Separating various tasks is one of the method's limitations, and proposed enhancements utilizing Bayesian principal component analysis are discussed.

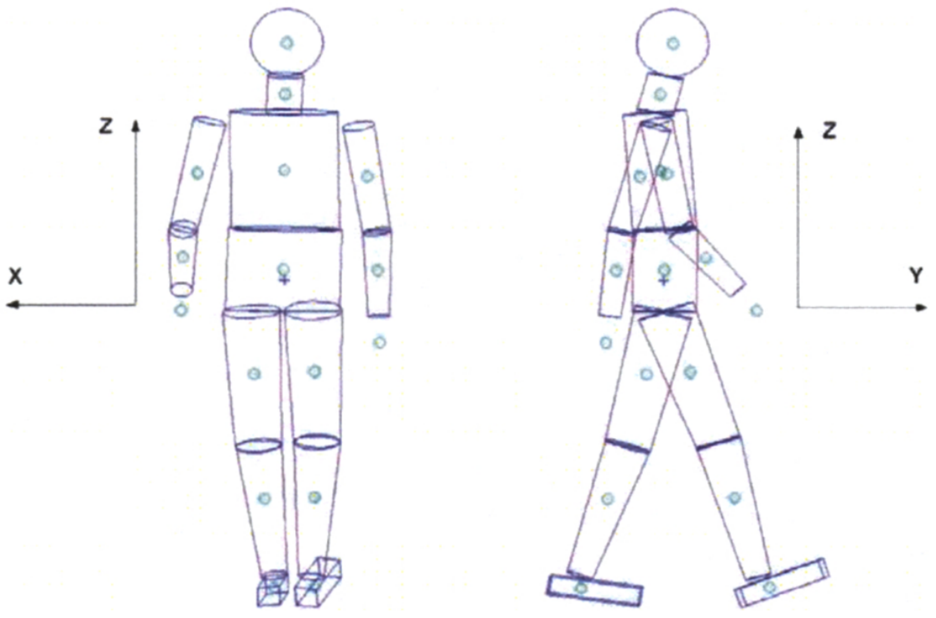

FIGURE 2.17: A high-dimensional model of the humanoid robot was used to estimate the spin angular moment about the COM. The model features 38 DOF externally and 32 DOF internally, 12 for the legs, 16 for the arms, and six for the remainder of the body [17].

The purpose of [18] is to develop generic stability criteria by studying the underlying physics of rotational stability of multi-body systems. The rate of change of a robot's centroidal angular momentum, as the physical quantity conveying its stability information, is the research subject. Three control techniques for biped robot stability recapture that exploit the robot's rate of change in centroidal angular momentum are presented. A derived condition for free walking on horizontal ground refers to a location on a robot's foot surface where the entire GRF would have to operate so that the robot's rate of change in centroidal angular momentum equals zero. The resulting GRF, denoted by R, flows through the COM, represented by G, as shown in Figure 2.18. As a result, rate of change in angular momentum is zero, and the robot is rotationally stable. The GRF produces a non-zero net moment about the COM in Figure 2.18. Therefore, the robot has a propensity to fall over. The rate of change of angular momentum would be zero if we laterally shifted the GRF to act along a new line of action going through the COM, and the robot would be stable. The contact point of the ground and the shifted GRF are shown at point A in Figure 2.18.

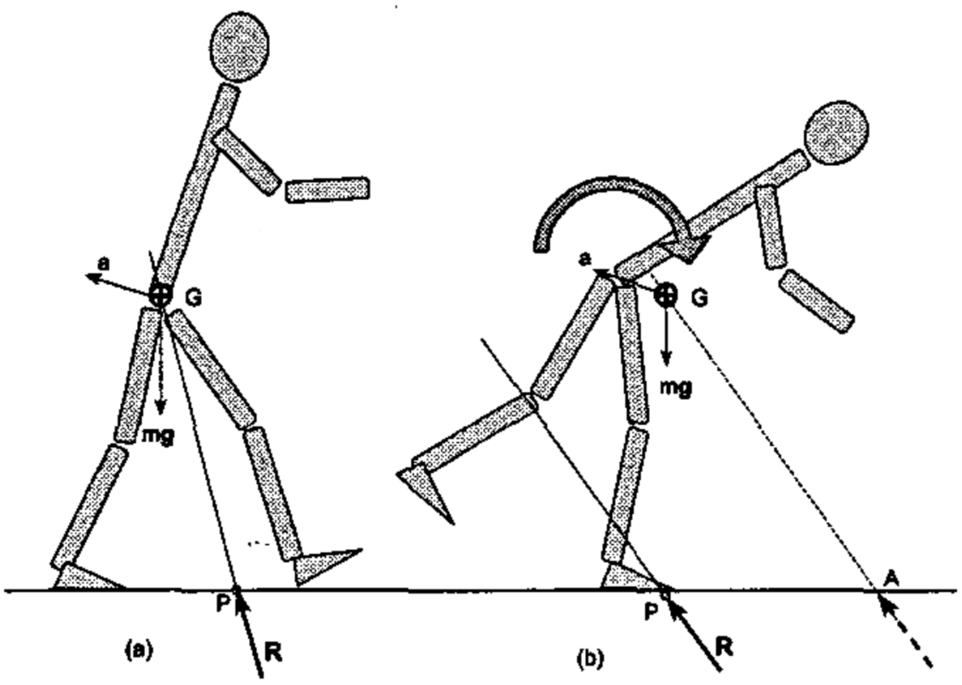

FIGURE 2.18: This diagram depicts the basis of stability analysis based on the rate of change of a robot's centroidal angular momentum and presents the notion of the angular momentum point with zero rate of change [18].

In [111], a motion-embedded COM Jacobian algorithm with angular momentum resolution is used to provide a walking algorithm for bipedal robots. The angular momentum equation is only employed for upper body motion determination in this technique, and no additional subject variables are used. Whole body cooperative motion is accomplished with walking restrictions defined by motion embedded COM Jacobian.

Chang et al. [112] propose a push-recovery approach for stabilizing the robot in the face of external disturbances. COG angular momentum regulator, COG state estimator, and stepping control are integral to the the technique, which alters the COG and swing leg trajectories in real time. The COG angular momentum regulator regulates the dynamics of the COG as an impedance system using the centroidal-moment-pivot criteria and feedback from a Kalman filter based COG state estimator. The stepping control chooses the best balancing reply before the robot's reactions to external disturbances.

A balance control technique for a bipedal robot standing on one leg is presented in [113]. First, the regulation of a contact torque acting from the robot's foot to the earth's surface is addressed. They also stress the relevance of back drivability of the contact torque controller. It is demonstrated that a two-DOF feedback controller produces a satisfactory outcome in this regard. Second, a novel balance control concept is presented based on direct feedback of total angular momentum and COG location.

A bipedal robot whole-body motion method that results in predefined total linear and angular momentum values is proposed in [19]. A linear equation that calculates a robot's overall momentum based on its physical properties, such as base link and joint velocities, is created. Constrains that are between legs and environment are also taken into account. A pseudo-inverse of the inertia matrix determines whole body motion from a particular momentum reference. Kicking and walking movements are tested on the HRP-2 humanoid robot (Figure 2.19).

The angular momentum created by the lower body movement of a bipedal robot is estimated in [114], and the torso and arm rotations are planned in such a way to counteract the lower body angular momentum. A bipedal robot upper-body mechanism that resembles the human arm length and mass characteristics is also constructed. The humanoid robot achieves angular momentum adjustment in the yaw direction while it is in the air.

A biologically inspired walking control technique that explicitly controls the system's angular momentum is provided in [115]. They calculate the distribution of angular momentum across the human body using human kinematic locomotion data at slow and self-selected walking speeds. According to principal component analysis, three angular momentum primitives describe 99 percent of the walking variables for the sagittal plane body rotations. The findings reveal that the angular momentum primitives are unaffected by walking speed. A morphologically accurate humanoid model walking in the sagittal plane to represent human walking

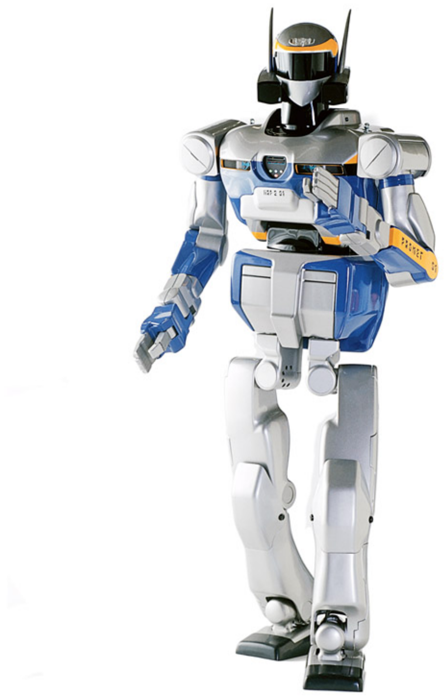

FIGURE 2.19: HRP-2 humanoid robot [19].

in the single support phase is used. The desired gait motion has minimum predetermined specifications. The resultant joint kinematics model is qualitatively similar to human gait data, suggesting that invariant angular momentum primitives in bipedal robot control might be crucial for establishing biological realism in legged robots and prosthetics. The angular momentum primitives architecture cis successful in gait synthesis, allowing the controller of a bipedal robot or powered limb prosthesis to adjust stride length and walking pace conveniently.

In [20], a model-based and efficient link-to-link distance computation method is proposed for robots. In order to prevent collisions, the findings are put into the inverse kinematics null space term, which is based on LIÉGEOIS' redundancy resolution methodology. A unique strategy for reducing vertical angular momentum for a walking humanoid robot is also suggested. This method employs arm motion. The approach may be coupled with the suggested collision avoidance strategy because it also works in null-space. Finally, findings from simulations and trials with the robot Lola (Figure 2.20) are presented with the suggested methodologies.

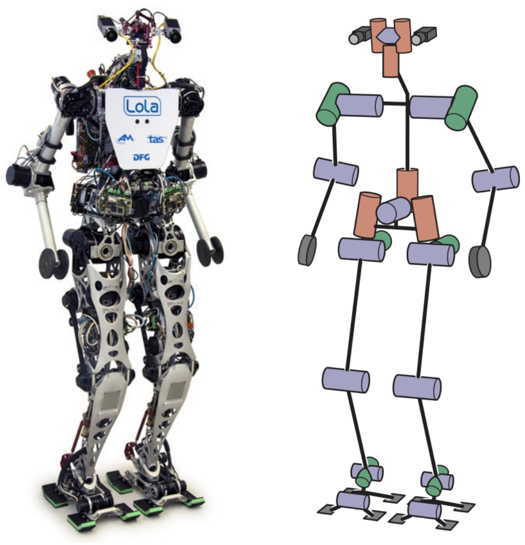

FIGURE 2.20: The humanoid robot Lola. Photo and kinematic structure of the robot system [20].

Several push forms are examined, and a controller for dealing with a sudden push on a bipedal robot during a walking gait is proposed in [116]. The controller senses a push, simultaneously determines the robot's required step locations to recover from the impact, and computes the joint motion needed to balance the robot effectively. Balance and posture control strategies are based on angular momentum.

In another work, Adiwahono et al. present a control system for bipedal robot walking and a push recovery. When the robot is subjected to an impact force, the algorithm adjusts the walking phase to keep the robot walking while taking the constraints into consideration [117].

Two methods for disturbance rejection are discussed in [118]. The first method is a disturbance observer and PI controller combination that extends the divergent component of motion and CP tracking controllers. Employing the error of momentum rate-of-change, transient disturbances are computed. An optimization-based divergent component of motion dynamics is provided for more substantial disturbances. The method employs a quadratic algorithm to obtain the appropriate step position and GRF. The optimization allows for the design of the angular momentum rate of change algorithm to assist shortening the length of the recovery process.

Hosseinmemar et al. [119] offer a closed-loop feedback controller for a bipedal robot with an accelerometer and gyroscope to balance the robot during walk and recover from external disturbances. Three balancing methods are investigated regarding humanoid robots: COP, centroidal moment pivot, and stepping strategy. In order to study recovery from impacts, experiments are conducted with three closed-loop feedback controller configurations: Employing only the gyroscope, only the accelerometer and a combination of both sensors. In order to classify pushes of varying strengths, each sensor is discretized into four distinct domains.

A multi-level postural control integration problem on a bipedal robot is discussed in [120]. Hyon et al. provide a comprehensive perspective of postural stability, including ankle and hip strategies. They employ two converters for required ground response force to whole-body joint torque. Due to the joint redundancy, an angular momentum controller is also proposed to regulate the internal movements. This article shows that replacing COM feedback with local joint stiffness improves the stability of the bipedal robot for specific rapid maneuvers.

Kojio et al. present a step modification strategy that considers the available moving range in [121]. Stable posture is achieved even on steps where the available moving range is strongly constrained. The proposed approach alters the step location, the step time and angular momentum. The available moving range is expressed as convex polygons. This allows gait characteristics to be determined analytically.

In [122], linear equations of motion for the location of the COP and the rate of change the angular momentum about COM are obtained linear. This is achieved by considering that the bipedal robot is following a predefined reference trajectory for the height of COM. MPC is utilized to solve for control inputs across a receding horizon. This yields reference trajectories for the COP and the rate of change of angular momentum about the COM.

Lee and Goswami describe a stability approach that manages the robot's linear and angular momenta [123]. The goal of the controller is to achieve the desired momentum. Which allows regulation of the posture of the bipedal robot. This technique works on uneven terrains and varying frictional characteristics at each contact between foot and ground by directly calculating the GRF and COP at support foot to obtain the appropriate momentum. In case the robot is not achieving the necessary linear and angular momentum values simultaneously, the controller prioritizes linear momentum at the expense of angular momentum.

The focus of [124] is to manage lateral plane disturbances and integrate them with a planar direction method. The humanoid robot is balanced in a walking gait employing different recovery techniques. Moving a leg or turning the torso are employed as angular momentum generators.

Angular momentum of a humanoid robot is utilized in [125] to achieve push recovery. In order to alleviate the disturbance and achieve posture stability, the proposed approach actively creates angular momentum references depending on the magnitude of the force and direction of the push. The reference of the angular momentum about COM is created by first computing angular momentum around COM for counterbalancing the push, then for decreasing angular momentum to stop the bipedal robot and finally for posture recovery to return to the desired posture reference.

A reactive stepping regulator for bipedal robot posture recovery is proposed in [126]. The regulator is based on momentum and can assist the robot in regaining equilibrium with or without taking another step by carefully managing linear and angular momentum combinations. The desired stepping position is determined by modeling the robot as a wheel. Arriving at the desired position causes the wheel to stop completely. The period of a step may be calculated from the position of the reference point.

2.3.3 Miscellaneous Robots

One of the tools for manipulating the angular momentum of a robot is adding a tail to the structure of the robot. Machairas and Papadopoulos [39] investigate the posture stability and control of robots with tails for the flight phases of dynamic quadruped gaits. The dynamics of a robot body whose posture is regulated by a spinning tail are represented employing a two-body template. Model based controllers are developed with the differential equations for a tail and a response wheel. The differential geometry of the system depends on initial angular momentum. The tail and the reaction wheel are compared for performance and fundamental steps and equations for selecting essential parameters in the design of such systems are suggested.

Xiaoyun et al. [21] propose a robot with a three DOF tail (Figure 2.21). A stabilizing mechanism based on the tail and the concept of angular momentum conservation is proposed. The balancing mechanism works with the sudden rotation of the tail in the same coordinate axis of the fall causes the robot body with the angular momentum in the opposite direction. The tailed robot can recover its posture to its initial position and prevent tipping over after a significant disturbance by using an angular momentum strategy and employing PD controller with a feedforward term.

The angular momentum approach can also be employed on robots lacking contact with the ground. Examples are space robots. The angular momentum control of the combined system after catching the object in orbit is investigated in [80]. The contribution of this study is the development of a regulated electromagnetic damper compatible with a space robot's joint and a combined angular momentum control approach depending on the damping joint. The kinematics equations of the robot base, manipulators with damping joints, and target spaceship are developed with the Kane technique. A framework for the electromagnetic damper is developed, and output characteristics of the damper are evaluated using analytical equations. This approach reduces the spin-axis angular momentum of the robot base while softening the impact.

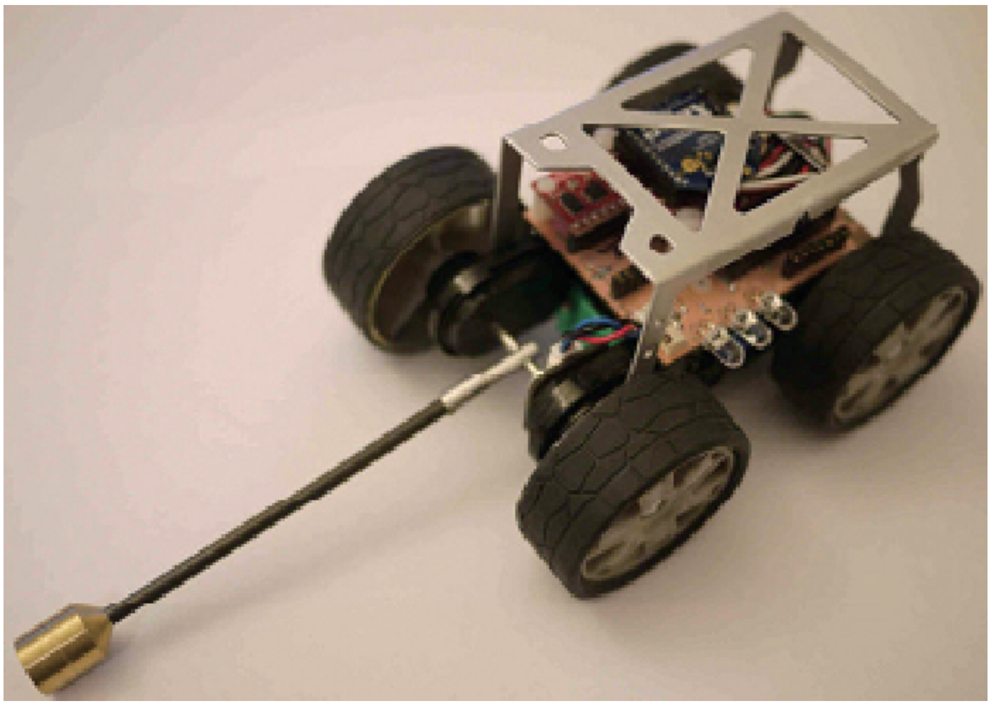

FIGURE 2.21: Robot with 3 DOF tail [21].

Nohmi et al. [127] discuss a space robot system consisting of a spaceship and a robot attached to it via a cord. The attached robot moves away from the spaceship wing with known tether tension regulation approaches in the gravitational field. Link movement of the attached robot, on the other hand, is challenging since the momentum is not conserved because of the existence of external forces. This study concentrates on attached robot momentum regulation in particular. They demonstrate that the proper motion of the cable attachment site can regulate the attached robot's rotational momentum. A control approach is presented for the attached robot link motion divided into two tasks: End-effector motion and cable attachment site motion.

A genetic algorithm is employed by Tang and Chen to handle the challenge of nonholonomic trajectory planning for a free floating space robot platform with two arms in [128]. The mathematical framework of the model for control system design is constructed by utilizing the space robot system's linear and angular momentum conservation. The control strategy for the system is investigated, and a genetic algorithm is employed to achieve optimal control parameters. The suggested optimum motion planning technique can achieve desired angular positions of the base attitude and arm joints by regulating the motion of the arm joints.

Ikeda et al. [129] present a control approach called variable constraint control for regulating the posture of free flying robots like space robots subject to angular momentum conservation. A holonomic constraint is employed on the nonholonomic system, called a control constraint. The original constraint is replaced by a holonomic constraint. The integrals of these constraint functions act as constant manifolds. Then, they regulate zero dynamics variables while preserving the control constraints.

Gupta et al. [130] present a method for a quadrotor with variable pitch propellers to increase the rate of change of thrust creation for active maneuvering. In combination with the momentum approach, the blade element approach is utilized to calculate aerodynamic loads, which are necessary for constructing the dynamics model of the quadrotor. In addition, a nonlinear controller designed for trajectory tracking via dynamic inversion. Three loops are used in the controller. The outer loop computes the translation dynamics in order to create thrust, pitch, and roll angle references necessary to attain a specific state or trajectory. The inner loop optimizes the rotational dynamics to produce necessary angular velocities by utilizing the references produced in the outer loop.

Mita et al. [131] derive an analytical solution for a two-link free flying robot to regulate the initial angular momentum condition. They demonstrate a mathematical solution to the issue of time optimum control. Simple closed loop control law equations are obtained. It is also shown that the problem results in a particular optimum control issue depending on the initial conditions.

The angular momentum approach can also be employed to robots lacking contact with the ground almost always during their gait cycles. Hopping robots (Figure 2.22) make examples in this category. A hopping motion controller structure is suggested in [22]. In order to disperse the torque load of motors, the hopping robot presented in this study includes a parallel link topology. The controller building blocks offered are collision relief control, hopping attitude and velocity control utilizing conservation of angular momentum, and equipment control for the touchdown. The first controller uses the compliance control to reduce the collision force during touchdown. The second controller has two parts for velocity and attitude of the motion. By using the conservation of kinetic energy, the velocity command is generated in hopping velocity control. The order for hopping attitude control is obtained from the variation in posture during the airborne phase. The next touchdown is planned by equipment control for landing.

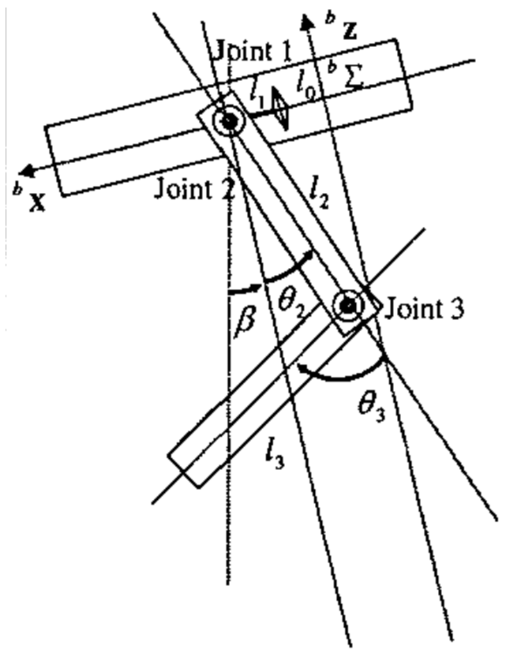

FIGURE 2.22: Sagittal plane of the hopping robot [22].

In another study with the angular momentum approach walking gait stabilizers for general legged robots are investigated. Powell and Ames [132] propose stabilizing underactuated walking robots. Employing continuous time control and hybrid system models that record impacts during footstrike, the regulator stabilizes the transmission of angular momentum from one leg to another. Angular momentum conservation upon contact enables the precise calculation of the angular momentum transfer function that depends on the stride length of the robot and vertical COM velocity right before foot contact. Reference trajectories for the step length of the legged robot and vertical COM coordinate can be computed.

Finally, in [133], a general method is proposed for creating dynamically stable gaits for legged robots. A walking pattern generator that can generate a stable COM trajectory by using angular momentum on difficult terrain conditions is presented. The solver is sufficient enough to serve as an MPC. In addition, a contact planner is employed to generate the contact pattern from a simulation of the system and the desired posture. The algorithm then computes a stable walking gait pattern, and inverse kinematics is utilized to generate a dynamically stable full body trajectory. This approach is efficient enough to design a step while the previous step is performed.

2.4 Contributions of This Thesis

Machine learning strategies for quadrupedal robots are categorized into collision detection, terrain classification, path planning, motion planning and posture recovery subjects. Since posture recovery is the main concentration of this thesis, the literature survey on posture recovery is expanded to bipedal and quadrupedal robots. Although many studies are proposing machine learning methods for legged robots, gaits with no ground contact (such as a long jump) are not studies in the research surveyed in Section 2.1. Posture recovery methods mentioned Section 2.1 depend on ground contact points or stepping strategies in order to balance the posture of the robot.

Balance and posture control strategies without machine learning or angular momentum approaches for bipedal and quadrupedal robots are surveyed as well. The main concentration of these studies is balancing legged robots under external disturbances or performing dynamic gaits. Running and walking gaits are considered in Section 2.2. Therefore, ground contact forces are available for posture control.

Studies on balance and posture control strategies with the angular momentum approach are investigated in the literature. Since this topic is highly related to this thesis, this section is expanded on quadrupedal, bipedal and miscellaneous robots. Similar to Section 2.2, the studies on posture control with the angular momentum approach for bipedal and quadrupedal robots include ground contact points. For miscellaneous robots, when ground contact is absent, adding a tail to the robot system is used to balance the robot. Other approaches include adding a cord or a damper to the robot to control the posture of a space robot or utilizing propellers.

This thesis focuses on the posture control of legged robots with no ground contact and without additional mechanical apparatus to manipulate angular momentum of the system.

Chapter 3

System Modeling and Control Methods

3.1 Floating-Base Multi-Body Dynamics

In order to characterize the motion of floating base systems, the concept of generalized coordinates is employed. In this approach, system kinematics and dynamics are defined in terms of n-dimensional vector functions.

$$\theta = \begin{bmatrix} \theta_g \\ \theta_j \end{bmatrix}, \tag{3.1}$$

where

$$\theta_g = \begin{bmatrix} x & y & z & \alpha & \beta & \gamma \end{bmatrix}^T . \tag{3.2}$$

Here x, y and z are unactuated robot positions with respect to world coordinate frame x, y and z axes (3.1), α, β and γ are underactuated robot orientation angles about world coordinate frame x, y and z axes and θ_j represents actuated joint positions (front and rear spine joints (θ_b and ϕ_b) and leg joints). The front torso component serves as the base of the robot. The location and orientation of this robot base are specified in terms of the underactuated floating base coordinates relative to the world frame. Accordingly, equations of motion are expressed as,

$$M(\theta)\,\ddot{\theta} + C(\theta,\dot{\theta}) + G(\theta) = S^T\tau, \tag{3.3}$$

where $M(\theta) \in \mathbb{R}^{n\times n}$ denotes the inertia matrix, $C(\theta,\dot{\theta}) \in \mathbb{R}^{n\times 1}$ represents the Coriolis and centrifugal forces, $G(\theta) \in \mathbb{R}^{n\times 1}$ stands for the gravitational effect, and $S \in \mathbb{R}^{n-6\times n}$ denotes the selection matrix of the actuated joints. The selection matrix S distinguishes between actuated and unactuated floating base coordinates. The vector of actuated joint torques is denoted by $\tau \in \mathbb{R}^{n-6\times 1}$.

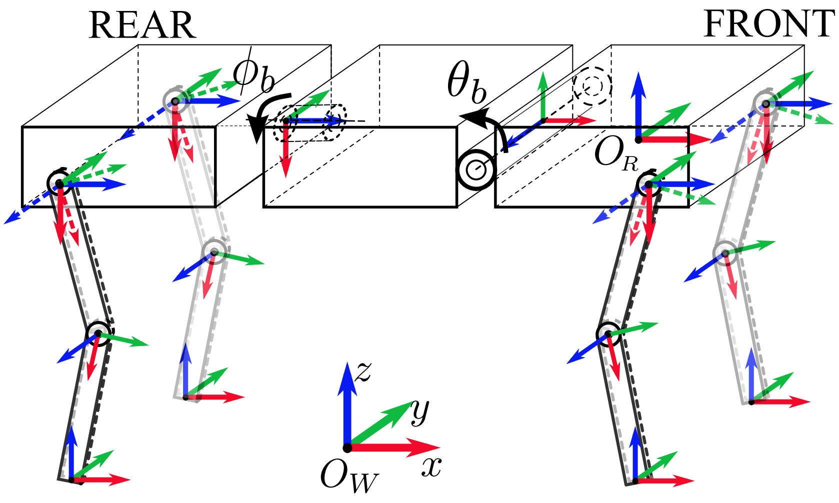

FIGURE 3.1: The full-body quadruped's frame positions. The x, y, and z axes are shown by red, green, and blue arrows, respectively.

3.2 Quadruped Robot Simulation Environment

3.2.1 Quadruped Body Kinematics

The quadruped robot model features a total of 20 DOF, with three DOF in each leg and two DOF in the spine. The remaining six DOF belong to the floating-base in three-dimensional space. Leg joints are rotational. Each leg has a hip adduction/abduction joint, as well as hip and knee flexion/extension joints. Figure 3.2 provides an example. Three distinct body sections and two joints are used to generate dynamic spine motion (Figure 3.3). The quadruped's full-body frame positions are shown in Figure 3.1.

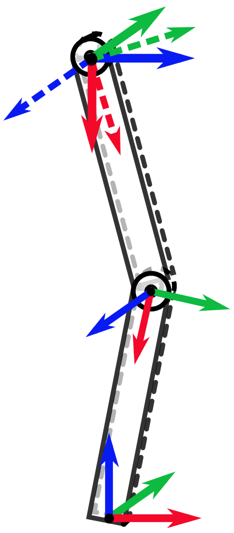

FIGURE 3.2: The robot leg's kinematic layout and frame positions. The x, y, and z axes are shown by red, green, and blue arrows, respectively.

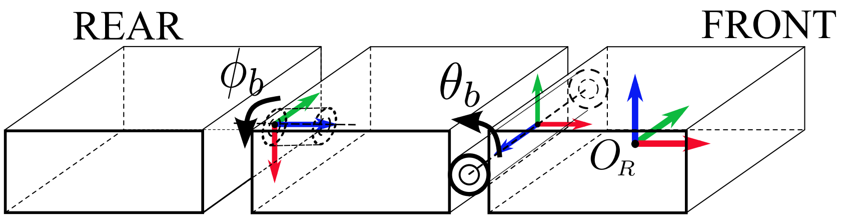

FIGURE 3.3: The robot body's kinematic layout and frame positions. The x, y, and z axes are shown by red, green, and blue arrows, respectively.

Coordinate transformation matrices for the front and hind body frames are as follows:

$$ {}^{B}_{B_m}T = \begin{bmatrix} \cos(\theta_b) & -\sin(\theta_b) & 0 & -\frac{l_{b1}}{2} \\ 0 & 0 & -1 & 0 \\ \sin(\theta_b) & \cos(\theta_b) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.4} $$

$$ {}^{B_m}_{B_r}T = \begin{bmatrix} 0 & 0 & 1 & -l_{b2} \\ \sin(\phi_b) & \cos(\phi_b) & 0 & 0 \\ -\cos(\phi_b) & \sin(\phi_b) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.5} $$

Here θ_b is the pitch angle of the front spine joint and ϕ_b is the roll angle of the rear spine joint. l_{b1} is lenght of the front body and l_{b2} stands for the lenght of the middle body (Table 3.1).

TABLE 3.1: Simulation parameters.

Simulation Parameters			
Definition		Symbol	Value (Unit)
Body (Front-Middle-Rear)	Lenght	l_{b1}, l_{b2}, l_{b3}	0.4 – 0.2 – 0.4 (m)
	Height	h_{b1}, h_{b2}, h_{b3}	0.15 – 0.15 – 0.15 (m)
	Width	w_{b1}, w_{b2}, w_{b3}	0.6 – 0.6 – 0.6 (m)
	Mass	m_{b1}, m_{b2}, m_{b3}	20 – 10 – 20 (kg)
Leg (Upper - Lower)	Lenght	l_{ul}, l_{ll}	0.4 – 0.4 (m)
	Height	h_{ul}, h_{ll}	0.06 – 0.06 (m)
	Width	w_{ul}, w_{ll}	0.1 – 0.1 (m)
	Mass	m_{ul}, m_{ll}	5 – 5 (kg)
Gravitational acceleration		g	9.81 (kgm/s^2)
Sampling time		t_s	0.0005 (s)

3.2.2 Orientation Representation

The following procedure is used to represent the angular position of the robot body in three-dimensional space. Given that the orientation of the robot body has three DOF, three principal rotations are employed to describe orientation. A rigid body's orientation may be defined as the orientation of a fixed reference frame attached to the robot body. A mapping matrix between the world frame and the frame connected to the quadruped robot's main body can be defined. Three sequential rotations around the world frame are chosen to describe the robot's body orientation. These are roll, pitch, and yaw rotations. Finding a mapping matrix between the Euler angle rates and the robot body's angular velocity is critical for the building of a 3D robotic simulation environment. The mapping matrix is calculated as in [134]:

$$\begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = E^{-1}\, {}^{B}_{W}R\, {}^{W}w_{B}, \tag{3.6}$$

Here, ${}^{B}w_{W}$ is angular velocity of robot body with respect to world coordinate frame. $\dot{\alpha}$, $\dot{\beta}$ and $\dot{\gamma}$ are angular velocities of the robot body along x, y and z axes respectively. E is the matrix that maps the body's angular velocity with respect to the body frame to the Euler angle rates. ${}^{B}_{W}R$ is rotation matrix between world coorditane frame to body coordinate frame.

$$E^{-1}\, {}^{B}_{W}R = \begin{bmatrix} 1 & \frac{\sin(\alpha)\sin(\beta)}{\cos(\beta)} & \frac{-\cos(\alpha)\sin(\beta)}{\cos(\beta)} \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & \frac{-\sin(\alpha)}{\cos(\beta)} & \frac{\cos(\alpha)}{\cos(\beta)} \end{bmatrix}. \tag{3.7}$$

When the pitch angle (β) equals 90 or 270 degrees, the mapping matrix is singular. However, these values are unlikely to be reached in quadruped motion. A negligible overflow number might be introduced to avoid singularity.

3.2.3 Quadruped Body Dynamics

The Lagrangian technique is used to generate the dynamic equations of a quadruped robot. This technique makes use of the Euler-Lagrange equations, which are derived by the kinetic and potential energy of the system. The difference between the kinetic and potential energy of a system is defined as the Lagrangian function for the purpose of creating dynamic equations.

$$\mathscr{L}(\theta, \dot{\theta}) = K(\theta, \dot{\theta}) - U(\theta). \tag{3.8}$$

In (3.8), $\mathscr{L}$ denotes Lagrangian function, K and U stand for the kinetic and potential energy terms of the system, respectively. The equations of motion of a quadruped robot with generalized coordinates ($\theta \in \mathbb{R}^n$) and a Lagrangian are formulated as,

$$\frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{\theta}_i} - \frac{\partial \mathscr{L}}{\partial \theta_i} = F_i \quad ; i = 1...n, \tag{3.9}$$

where F denotes all external forces and torques operating on the body and linkages of the quadruped robot. The size of the generalized joint coordinates column vector is denoted by n. When Euler-Lagrange equations are applied to a quadruped robot and equations of motion are organized in the manner specified in (3.3), the inertia matrix ($M(\theta)$), external forces (F), Coriolis and centrifugal forces ($C_i(\theta, \dot{\theta})$), and gravitational force ($G_i(\theta)$) are calculated as in [134]:

$$M(\theta) = \sum_{s=1}^{j} J_s^T(\theta)\ M_s(\theta)\ J_s(\theta), \tag{3.10}$$

$$C_i(\theta, \dot{\theta}) = \sum_{j,k=1}^{n} \left(\frac{\partial M_{ij}}{\partial \theta_k} - \frac{1}{2}\frac{\partial M_{kj}}{\partial \theta_i} \right) \dot{\theta}_j\ \dot{\theta}_k; \quad i = 1...n, \tag{3.11}$$

$$G_i(\theta) = \frac{\partial U}{\partial \theta_i} \quad i = 1...n, \tag{3.12}$$

$$F = S^T \tau. \tag{3.13}$$

Here, J_s represents link jacobian matrices and M_s is the matrix of inertia of the robot link.

3.3 Control of the Quadruped Robot

In this chapter, a classical PID control rule based on joint space dynamics was proposed. In order to track generated references torque control method is employed. Torque (τ) in Equation 3.3 is calculated as

$$\tau = K_p\, e + K_i \int_0^t e\, dt + K_d\, \dot{e}, \tag{3.14}$$

where e is angular position error obtained by $e = \theta^{ref} - \theta^{act}$, θ^{ref} is reference angular position and θ^{act} is actual angular position of the robot joint. K_p, K_i and K_d are proportional, integral and derivative gains of the PID controller (Table 3.2). These gains are obtained by trial and error method.

Table 3.2: Control Parameters

	K_p	K_i	K_d
Spine joints	1000	20	700
Leg joints	1000	10	800

Chapter 4

Learning-Based Posture Control of a Quadruped Robot in the Flight Phase of a Long Jump

Reinforcement learning is the process by which a machine learns what to do - how to translate events to actions - in order to maximize a numerical reward signal. The learner is not taught which actions to do; rather, it must determine which activities provide the greatest reward via trial and error [135].

This section describes a novel reference generation strategy based on RL for in-air stabilization of robots without ground contact sites during the flight phase of a long jump. This approach generates reference curves for the quadrupedal robot's waist joints in order to achieve proper orientation in the air. First, the methodology of this approach is presented. Then the proposed posture control approach is demonstrated in a simulation environment. Finally, discussions are presented.

4.1 The Methodology of the Reinforcement Learning Algorithm

The RL-based posture recovery algorithm is divided into two sections: The agent, which includes the policy and learning algorithm and the environment, consisting of the robot controller and robot plant. The agent is the one who makes decisions based on the potential for reward. The term "policy" refers to an agent's method for maximizing reward. The environment is the world with which the agent interacts. The aim of the agent is to choose the ideal course of action for the environment (robot) based on the reward function and observation of the environment's condition. The reward function is a motivational mechanism that informs the agent which actions are correct and which are incorrect. The agent needs inputs in the form of error and body orientation angles such as pitch (α) and roll (β) (Figure 4.1). The reward function is defined as follows

$$R(s_k) = -(e_k)^2. \tag{4.1}$$

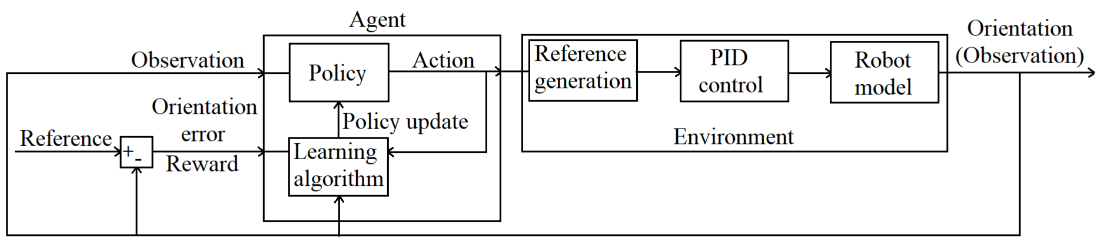

FIGURE 4.1: Overview of posture recovery algorithm.

Here, R represents the reward function, s is the state, k is the state index and e stands for the robot's orientation error relative to the world coordinate frame (Figure 4.2). The objective of the algorithm is to decrease orientation error. Roll and pitch orientation angles are assigned to be zero degrees in landing. A Q-table and a policy function are constructed. A Q-table is a straightforward look-up table in which the maximum projected future rewards for each action are calculated. The table contains information on the orientation of the robot, joint numbers, actions, and rewards. The motion is characterized by two waist joint positions, as illustrated in Figure 4.2. For the table, the exploration of action space and orientation space is discretized. Roll and pitch angle space is defined in increments of ten degrees from zero to forty degrees. The action space is specified as -0.286 to 0.286 degrees (-0.005 to 0.005 radians) in increments of $0,057$ degrees (0.001 radians). Table 4.1 illustrates the Q-table's structure. The number of training runs required to fill the Q-table is 220, since ten orientation angles of the robot body (five distinct pitch angles and five distinct roll angles) are examined with eleven distinct actions for two joints.

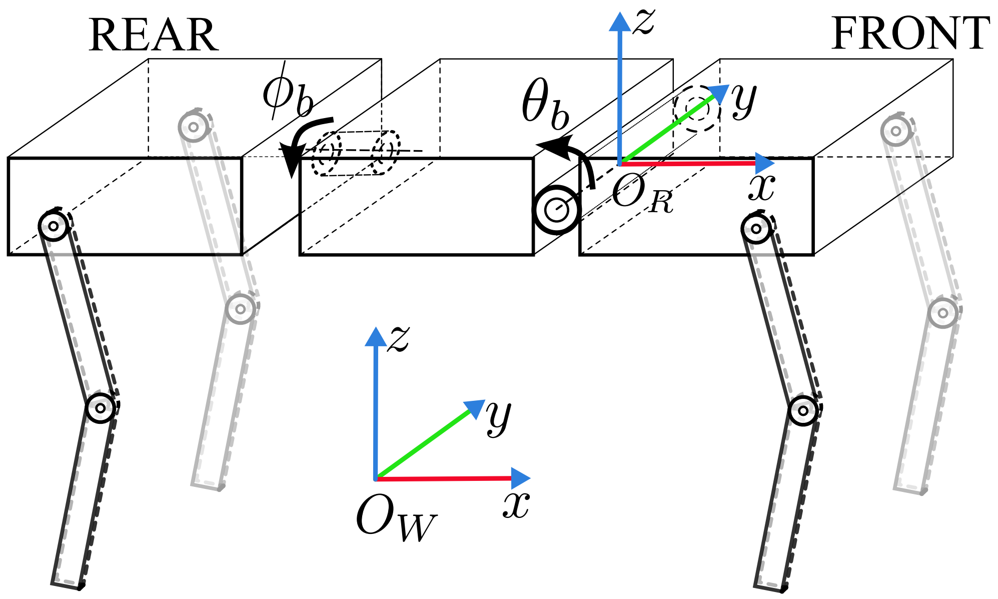

FIGURE 4.2: Robot model for simulation. θ_b is front spine joint angle, ϕ_b is rear spine joint angle and α (roll) and β (pitch) are body orientation angles about the world coordinate axes (red and green axes).

TABLE 4.1: Q-table for posture recovery algorithm.

Column 1	Column 2	Column 3	Column 4
Robot orientation angles $(\alpha,\ \beta)$	Waist Joints $(\theta_b,\ \phi_b)$	Actions $(\Delta\theta,\ \Delta\phi)$	Rewards $(R(s))$

After calculating reward values for the discretized orientation space, the continuous orientation space is handled by a policy function. Due to the roll and pitch angles, two distinct movements are necessary. The roll angle policy function is as follows,

$$\Delta\theta = [(\Delta\theta_1\ (\theta_2 - |\alpha|)) + (\Delta\theta_2\ (|\alpha| - \theta_1))]\ \frac{|\alpha|}{\alpha}\ \frac{1}{\theta_2 - \theta_1}. \tag{4.2}$$

Here, α denotes the current orientation of the robot along the robot's x-axis (roll angle) and $\Delta\theta$ is the best action. θ_1 and θ_2 are the two closest roll values found in the Q-table. $\Delta\theta_1$ is the optimal action for θ_1, while $\Delta\theta_2$ is the optimal action for θ_2. Due to the rotation's symmetry, if the ideal action for a particular roll angle is $\Delta\theta$, the optimal action for the negative of that roll angle is $-\Delta\theta$. As a result, this equation also takes into account negative orientation space. For example, if the robot's starting roll angle is -35 degrees, the closest roll angles to the absolute value of this number at the Q-table are 30 and 40 (θ_1 and θ_2) degrees. Then, for 30 degrees, the most awarded action is $\Delta\theta_1$ for the rear spine joint, and for 40 degrees, the most rewarded action is $\Delta\theta_2$. Similarly, the policy function for pitch angle is as follows,

$$\Delta\phi = [(\Delta\phi_1\ (\phi_2 - |\beta|)) + (\Delta\phi_2\ (|\beta| - \phi_1))]\ \frac{|\beta|}{\beta}\ \frac{1}{\phi_2 - \phi_1}. \tag{4.3}$$

Here, β denotes the current orientation along the robot y-axis (pitch angle) and $\Delta\phi$ is the best action. ϕ_1 and ϕ_2 are the closest existing roll values on Q-table. $\Delta\phi_1$ is the optimal action for ϕ_1 and $\Delta\phi_2$ is the optimal action for ϕ_2. Because simulations are conducted using a realistic robot model, the robot's joint torque and joint position constraints prevent speeds from changing the orientation in the air by more than forty degrees within a realistic flight period. Therefore, if the robot's orientation is more than 40 degrees, the optimal case is to repeat the procedure for 40 degrees. Finally, the angular position references for spine joints are determined using the following formulae.

$$\theta_b(T) = \theta_b(T-1) + \Delta\theta_b, \tag{4.4}$$

$$\phi_b(T) = \phi_b(T-1) + \Delta\phi_b. \tag{4.5}$$

θ_b denotes the angular position of the front spine joint, whereas ϕ_b denotes the back spine joint. T is the time step. While $\Delta\theta_b$ and $\Delta\phi_b$ denote the optimal action values. Along with the action value, the spine joint that will perform that action is also selected to maximize reward from the Q-table.

4.2 Simulation Results

Training simulations are used to construct a Q-table (Table 4.1). The number of training required to fill the Q-table is 220, since 10 distinct body orientation angles are examined, each having eleven distinct movements for two joints. Figure 4.3 illustrates eleven training results. The training set in question is for when the angular position of the robot body along the x-axis is 20 degrees and the actions for distinct $\Delta\phi_b$ values vary between 0.286 and -0.286 degrees (-0.005 and 0.005 radians).

The three steps of the quadruped jumping gait are takeoff, flight, and landing. During the takeoff phase, the velocity in the x and z axes relative to the world coordinate frame is set to 5 m/s. The location of the quadrupedal robot as a consequence of the initial velocities is depicted in Figure 4.4. The flight lasts around one second. In order to model an unstable takeoff, the initial body orientations α and β are adjusted to 15 and -25 degrees, respectively. These numbers have been selected purposefully to demonstrate how the learning process works for orientation angles other than those in the training set and for negative orientation angles. For both α and β, the reference body orientations are set at zero degrees. As indicated earlier, the extrapolated joint coordinates are θ_b and ϕ_b, which correspond

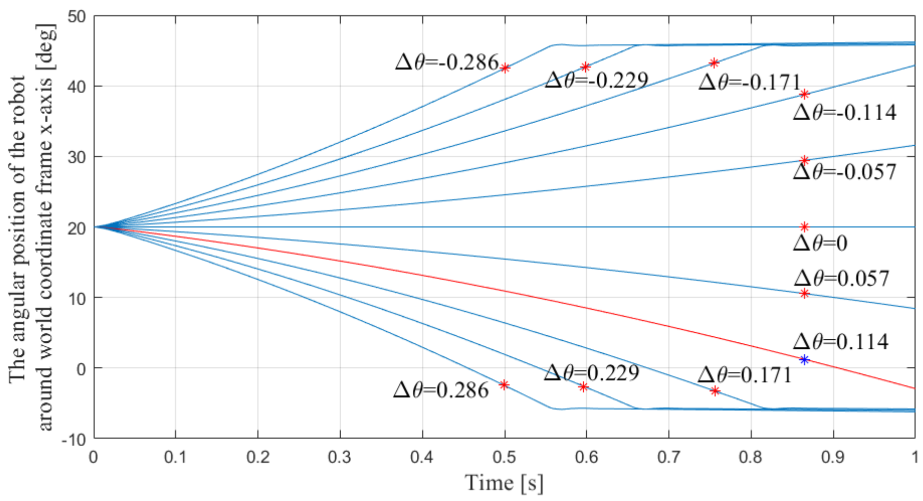

FIGURE 4.3: The angular position of the robot about world coordinate frame x-axis. Training results for 20 degrees. ϕ_b (rear spine joint angle) actions are presented. The red line is optimal action to take.

to the front spine joint angle and rear joint angle, respectively (Figure 4.2). The reference values generated are applied to the spine joints and the references are tracked using a simple PID controller. The computed references and the actual joint coordinates of the spine joints can be seen in Figures 4.5 and 4.6. According to the findings presented in Figure 4.5 and Figure 4.6, the PID controller is successful for tracing reference values. Throughout the flight period, the roll and pitch angles of the quadrupedal robot with respect to the world coordinate frame are presented in Figure 4.7. The intended final value is 0 for all angular positions. Additionally, posture recovery for body orientation is around 94 percent when the device is underactuated. Various initial roll and pitch body orientations are simulated; posture recovery is around 94 percent between -35 and 35 degrees. Outside of these values, orientation correction is approximately 30 degrees. The landing orientation of the robot is demonstrated to improve as it approaches the desired landing point.

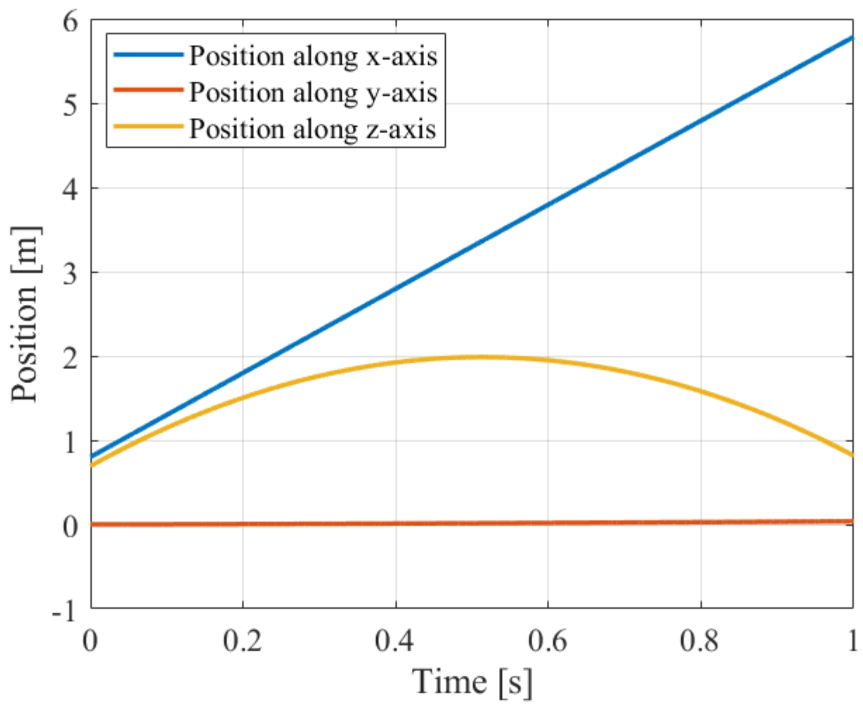

FIGURE 4.4: Linear positions of the quadrupedal robot along the x-axis (blue), y-axis (red), and z-axis (yellow).

4.3 Discussion

This study proposes a RL algorithm for generating reference trajectories for free-flying robots. This reference generation approach is employed on a free-flying quadrupedal robot in a simulation environment. Reference trajectories are generated for the front and rear spine joints for control of landing angular positions of the robot about the x and y axes of the world coordinate frame.

The method does not take any action on the body yaw angle (about z-axis of the world coordinate frame). This omission is acceptable since the yaw angle has no influence on the stability of the landing. Controlling roll and pitch angles is critical in this application because they have a significant effect on the landing stability of quadrupedal robots.

The primary interest in the study is to bring the front trunk block (termed as the body in this work) to a desired orientation. However, this process can move rear legs or body portions into locations not suitable for certain landing scenarios. The

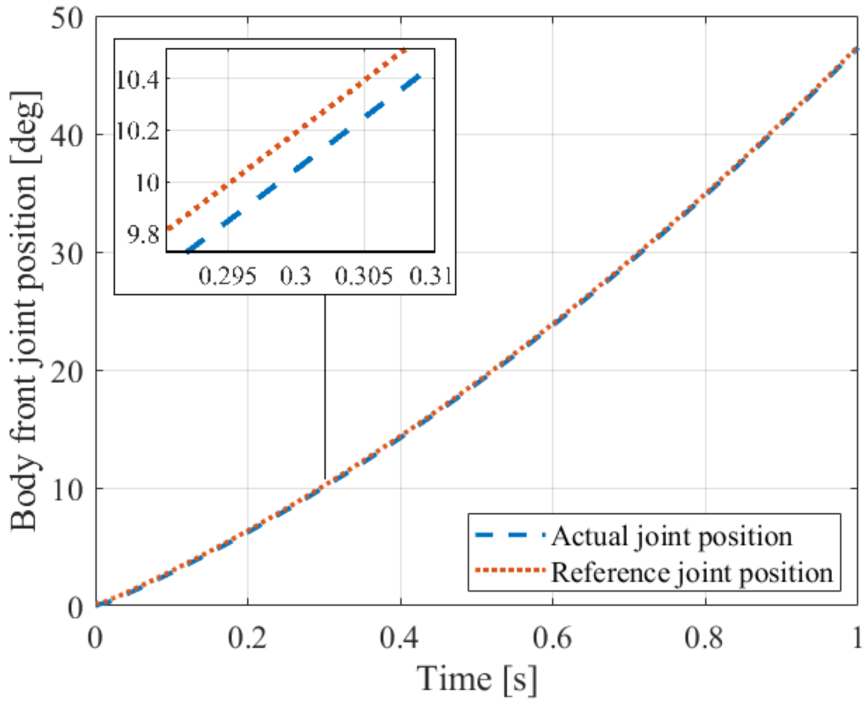

FIGURE 4.5: Front spine joint (pitch) actual position (blue) and reference position (red) with posture recovery algorithm.

landing posture of the middle and rear body parts and rear legs can be constrained depending on particular landing situations. They are, however, just constrained by joint space limits in this study.

In order to track generated references, a PID controller is employed. According to simulation data, the algorithm correctly recovered around 94% of the angular positions throughout the flight period. The learning-based reference generation strategy is applicable to any free-flying robot in order to improve flight and landing stability.

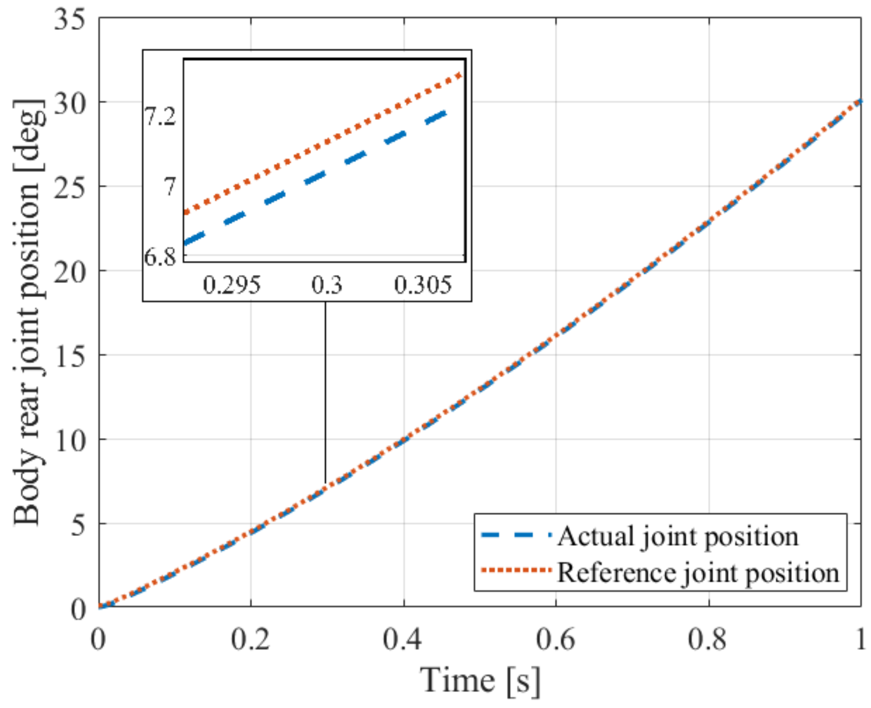

FIGURE 4.6: Rear spine joint (roll) actual position (blue) and reference position (red) with posture recovery algorithm.

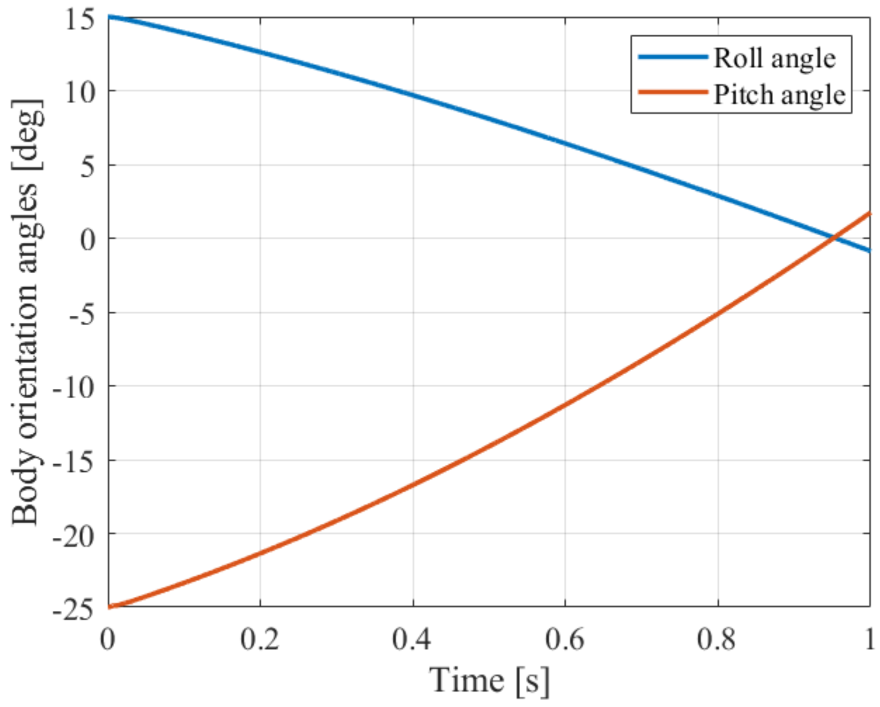

FIGURE 4.7: Angular positions (roll and pitch) of the quadrupedal robot about the world coordinate frame x-axis (blue) and y-axis (red) with posture recovery algorithm.

Chapter 5

Reference Trajectory Generation Using Angular Momentum for Posture Control of Free Flying Robots

This section presents a reference trajectory generation strategy for posture control of free-flying robots based on angular momentum. No physical modifications such as a tail or a gyro carried out on the system. This section is divided into three subsections: the methodology of the posture control algorithm, simulation results and discussion.

5.1 The Methodology of the Posture Control Algorithm

The trajectory generation algorithm uses the desired and actual body angular position and velocity values as inputs. These variables are assumed to be accessible. Since the workspace is three-dimensional, three equations can be formulated.

Three joints are chosen to adjust the overall angular momentum of the robot body in order to reach the desired angular position in the flight stage of a long jump. The expression for discretized angular momentum is

$$L(t + \Delta t) - L(t) = \tau \ \Delta t, \tag{5.1}$$

where $L(t)$ is angular momentum as expressed in the robot coordinate frame (Figure 5.1) and t is time. τ stands for the torque acting on the quadrupedal robot at time t. Δt is the time step in discretization, τ can be calculated as

$$\tau = r^{COM} \ M \ g, \tag{5.2}$$

where r^{COM} denotes the location of the robot COM in the robot coordinate frame. M is the robot's total mass, and g is the gravitational acceleration. The angular momentum at time t may be obtained using

$$L(t) = I_b \times \omega_b(t). \tag{5.3}$$

Here, I_b represents the moment of inertia of the total robot body with respect to the robot coordinate frame. $\omega_b(t)$ is the angular velocity of the robot body. Angular momentum at time $t + \Delta t$ can be expressed as

$$L(t + \Delta t) = I_b \times \omega_{b,d}(t + \Delta t) + I_i \times \omega_i(t + \Delta t). \tag{5.4}$$

The desired angular velocity of the robot body is $\omega_{b,d}$. The angular velocity of a chosen robot joint is ω_i and I_i is the moment of inertia of the moving robot limb due to the chosen robot joint. The moment of inertia of various parts of the robot are computed as

$$I_{cx} = \frac{1}{12} \ m \ (l_y^2 + l_z^2), \tag{5.5}$$

$$I_{cy} = \frac{1}{12}\, m\, (l_x^2 + l_z^2), \tag{5.6}$$

$$I_{cz} = \frac{1}{12}\, m\, (l_x^2 + l_y^2), \tag{5.7}$$

where I_{cx}, I_{cy}, and I_{cz} are the moment of inertia values about the x, y, and z axes, respectively, at the COM of the rectangular prism describing the robot body component (Figure 5.2). m is the mass of the part. l_x, l_y and l_z are length of the part with respect to their coordinate axes.

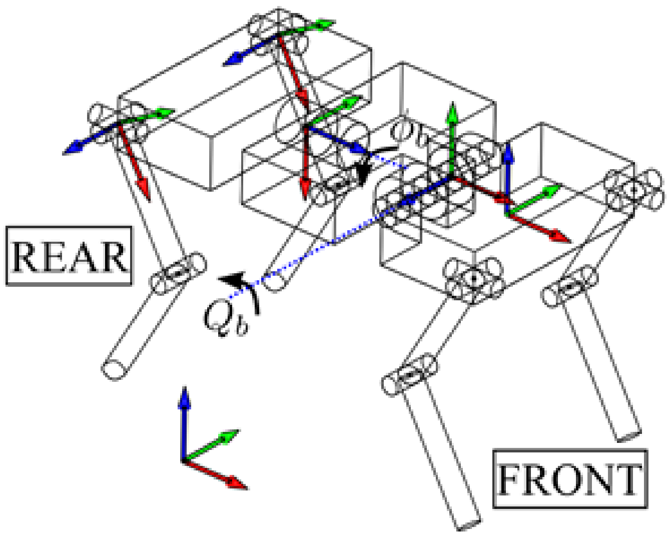

FIGURE 5.1: Quadruped robot model for simulation. θ_b is front spine joint angle (pitch) about corresponding blue axis, ϕ_b is rear spine joint angle (roll) about the corresponding blue axis and θ_{ulrr}, θ_{ullr} are rear hip joint angles about corresponding blue axes. α, β, and γ are body orientation angles about the world coordinate axes (red, green, and blue axes).

The parallel axis theorem is utilized to determine the moment of inertia with respect to robot coordinate axes:

$$I_i = I_{ci}\; m_i\; d_i^{\,2}. \tag{5.8}$$

The moment of inertia of the ith body component is I_i, the mass of the ith body component is m_i and the distance between the axes is denoted by d_i. When determining the moment of inertia for the middle torso component around the y-axis, for example, first I_{cy} is computed around y_{1c}, which is the y-axis through the

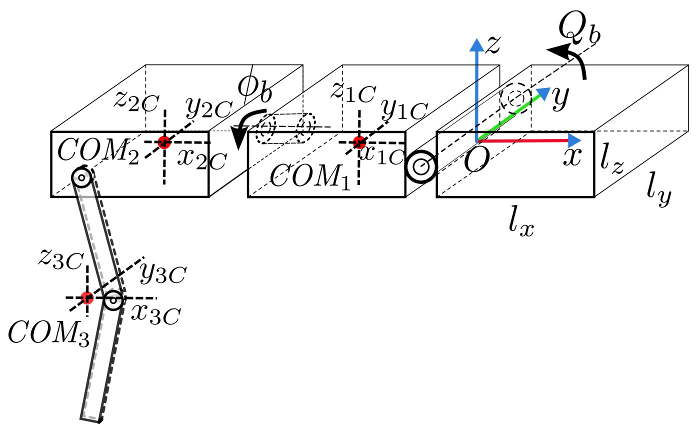

FIGURE 5.2: Quadruped robot model for the moment of inertia computations. COM_1 is the COM of the middle torso component, COM_2 is the COM of the rear torso component and COM_3 is the COM of the rear leg. The robot coordinate frame axes are shown with three different colors (red for x-axis, green for y-axis, and blue for z-axis).

COM of the body part, and then I_i is computed using the parallel axis theorem. d_i is the distance between y_{1c} and the y-axis of the robot coordinate frame, as shown in Figure 5.2.

Since roll, pitch, and yaw angles of the robot body (α, β, and γ), must be controlled, three joint angles are selected as action variables to construct a trajectory. As a result, the three orientation angles are controlled by three independent factors. The two spine joint angles θb, ϕ_b, and the two rear leg joint angles θ_{ullr}, θ_{ulrr} are the joint variables employed. For maximal precession of the total angular momentum around the z-axis of the robot's coordinate frame, rear leg joint angles are constrained to be equal in magnitude and opposite in direction. As a result, they can be considered as if they were one independent variable. Because the principal inertial axes of these joints are in the y, x, and z directions, they are chosen. Selected joint angles are assumed to have impact on just their respective primary inertial axes to simplify computations and reduce computing time. For example, changes in θ_{ullr} are thought to affect solely angular momentum around the z-axis. The following equations are found by computing the moment of inertia values and combining equations (5.1), (5.2), (5.3), and (5.4).

$$\dot{\phi}_b = \frac{\tau_x\ \Delta t + I_{b,x}\ \omega_{b,x}(t) - I_{b,x}\ \omega_{b,d,x}(t+\Delta t)}{I_{i,x}}, \tag{5.9}$$

$$\dot{\theta}_b = \frac{\tau_y\ \Delta t + I_{b,y}\ \omega_{b,y}(t) - I_{b,y}\ \omega_{b,d,y}(t+\Delta t)}{I_{i,y}}, \tag{5.10}$$

$$\dot{\theta}_{ullr} = \frac{\tau_z\ \Delta t + I_{b,z}\ \omega_{b,z}(t) - I_{b,z}\ \omega_{b,d,z}(t+\Delta t)}{I_{i,z} \times 2}. \tag{5.11}$$

The angular velocities of spine joints are denoted by $\dot{\phi}_b$ and $\dot{\theta}_b$. The torque components operating on the quadrupedal robot about the x, y, and z-axes, respectively, as stated in the robot coordinate frame are τ_x, τ_y, and τ_z. The angular velocities of the rear leg joints are represented by $\dot{\theta}_{\text{ullr}}$. The angular velocity components of the robot body in the robot coordinate frame are $\omega_{b,x}$, $\omega_{b,y}$, and $\omega_{b,z}$. The moment of inertia components of the complete robot body with respect to the robot coordinate frame, about the x, y, and z-axes, are denoted by $I_{b,x}$, $I_{b,y}$ and $I_{b,z}$. The desired angular velocities of the robot body about the x, y, and z-axes in the same coordinate frame are $\omega_{b,d,x}$, $\omega_{b,d,y}$, and $\omega_{b,d,z}$. The desired angular velocities and torque components can be computed using the formulas

$$\omega_{b,d,x}(t+\Delta t) = \frac{\alpha_d - \alpha}{\Delta t}, \tag{5.12}$$

$$\omega_{b,d,y}(t+\Delta t) = \frac{\beta_d - \beta}{\Delta t}, \tag{5.13}$$

$$\omega_{b,d,z}(t+\Delta t) = \frac{\gamma_d - \gamma}{\Delta t}, \tag{5.14}$$

and

$$\tau_x = r_x^{COM}\ M\ g, \tag{5.15}$$

$$\tau_y = r_y^{COM}\ M\ g, \tag{5.16}$$

$$\tau_z = r_z^{COM}\ M\ g. \tag{5.17}$$

The desired roll, pitch, and yaw angles are α_d, β_d, and γ_d, respectively.

5.2 Simulation Results

5.2.1 Simulation Results Without External Disturbance

After reference trajectories are computed with the posture control algorithm, obtained trajectories are applied in quadrupedal robot simulation. Simulations are carried out in the MATLAB & Simulink platform. Parameters of the robot model used in the simulation are given in Table 3.1. Locomotion parameters are presented in Table 5.1.

TABLE 5.1: Locomotion parameters for posture control method.

Locomotion Parameters		
Definition	Symbol	Value (Unit)
Take-off velocity about x-axis	v_x	5 (m/s)
Take-off velocity about y-axis	v_y	0 (m/s)
Take-off velocity about z-axis	v_z	5 (m/s)
Roll - Pitch - Yaw	v_z	$\pi/6 - \pi/6 - 0$ (rad)
Flight phase	t_f	1 (s)

In order to generate a jumping scenario, take-off velocity is set to 5 m/s at x and z-directions. The resultant motion has a one-second flight phase. Figure 5.3 shows the linear positions of the robot during a leap. The position control algorithm computes front and rear spine joint trajectories, as well as trajectories for the right and left hip adduction/abduction joints. In order to track the resulting joint reference trajectories, traditional independent joint PID position control is used.

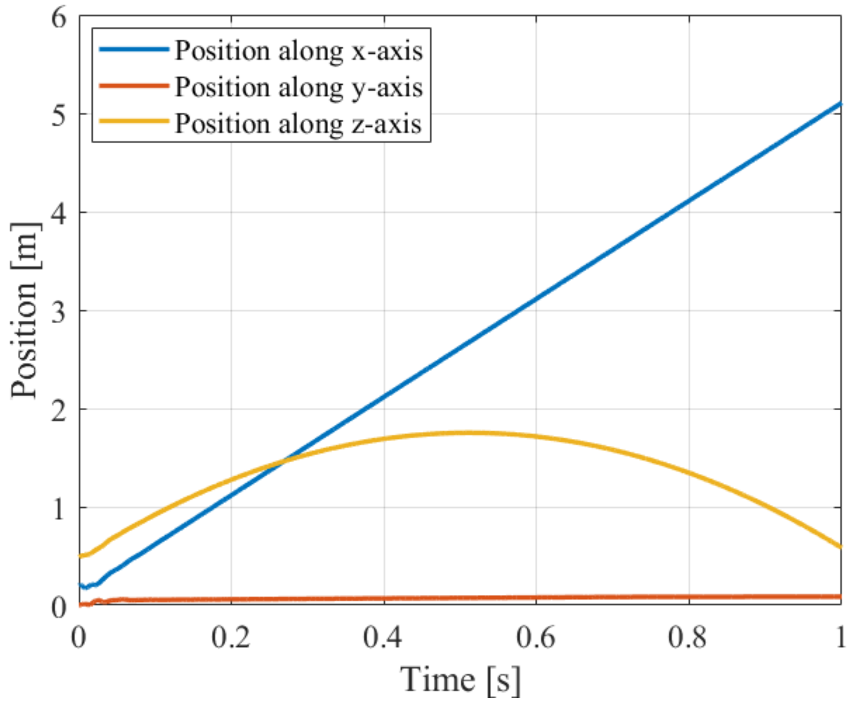

FIGURE 5.3: Linear positions of the quadrupedal robot performing jumping gait, components along the x-axis (blue), y-axis (red), and z-axis (yellow).

The reference and actual positions of the front spine joint can be seen in Figure 5.4. Figure 5.5 presents the actual and reference positions of the rear spine joint. Figures 5.6 and 5.7 show the actual positions and reference trajectories of the right and left hip adduction/adduction joint positions.

Roll, pitch, and yaw angles of the quadrupedal robot with respect to the world coordinate frame can be seen in Figure 5.8. The desired angular position for the robot to attain is selected to be zero radians around the x, y, and z-axes.

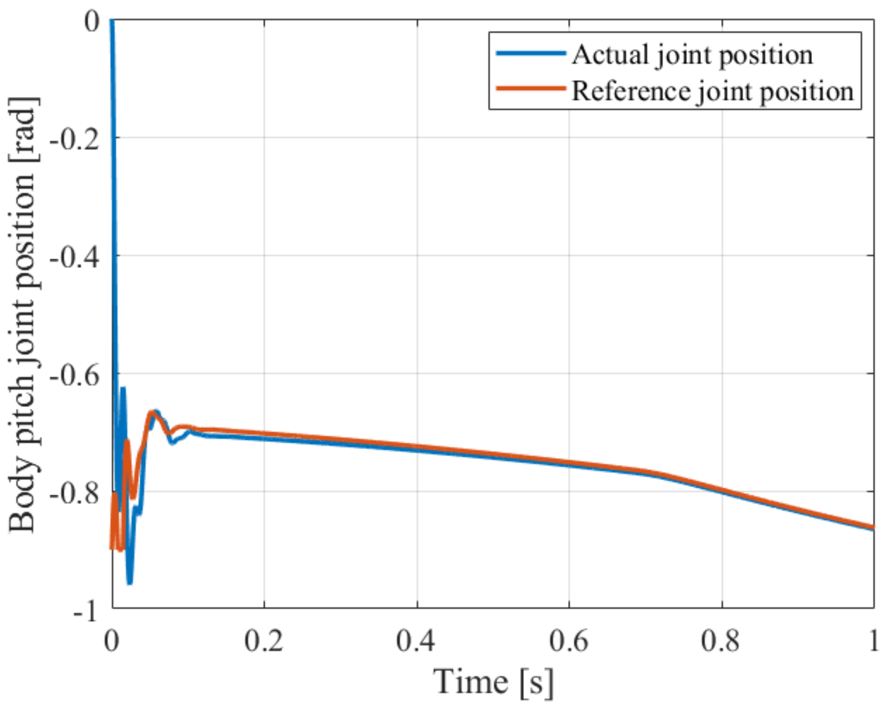

FIGURE 5.4: Front spine joint (body pitch joint) actual position (blue) and reference position (red) with posture control algorithm.

5.2.2 Simulation Results With External Disturbance

This subsection presents a different application of posture recovery algorithm. The method is applied for reference trajectory generation for push recovery of free-flying robots.

Unlike in Section 5.2.1 in (5.1), τ does not depend solely on gravity but also disturbance torque acting on the robot. The duration, magnitude and timing of the disturbance torque due to external impacts are unknowns. In an exact model the torque terms (τ_x, τ_y and τ_z) also include disturbance torque components. In addition, disturbance torques acting on the robot are assumed to change angular velocity components $\omega_{b,x}$, $\omega_{b,y}$, and $\omega_{b,z}$; therefore, the algorithm includes these disturbances to calculate angular velocities of spine joints $\dot{\phi}_b$ and $\dot{\theta}_b$ and hip joints $\dot{\theta}_{ullr}$.

Generated trajectories under disturbance effects are simulated. Locomotion parameters are presented in Table 5.2.

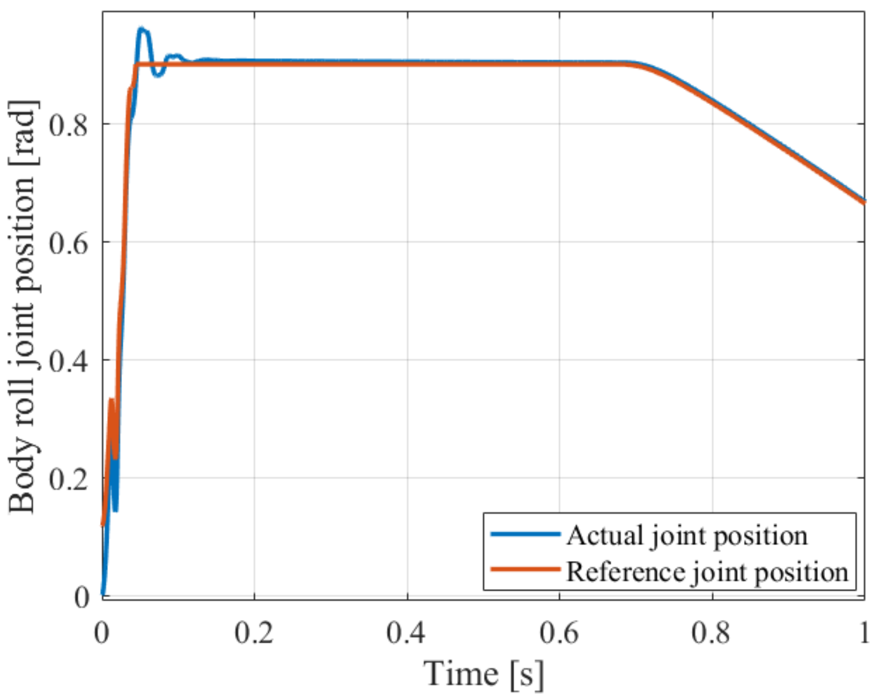

Figure 5.5: Rear spine joint (body roll joint) actual position (blue) and reference position (red) with posture control algorithm.

Take-off velocity is set at five m/s in the x and z axes to produce a jumping scenario—a one-second flying phase results from the motion. Figure 5.9 depicts the quadrupedal robot in a virtual environment. Figure 5.10 shows the quadrupedal robot's linear locations during a leap. At 0.2, 0.5, and 0.8 seconds after take-off, disturbances are created and applied to the robot model. In the robot coordinate frame, various disturbance torques are applied around the x, y, and z-axes, resulting in an equal increase in angular acceleration around these axes (see Figure 5.11). The applied torques around the x, y, and z-axes are 28.7 Nm, 76.6 Nm, and 102 Nm, respectively.

The algorithm computes front and rear joint trajectories, as well as right and left hip adduction/abduction joints trajectories. In order to track the resultant joint reference trajectories, again traditional PID control is employed. In Figures 5.12 to 5.15, calculated joint reference trajectories and actual joint angles are presented. The reference and actual locations of the front spine joint are shown in Fig. 5.12. The PID controller is quite successful in that the errors are smaller than 0.002

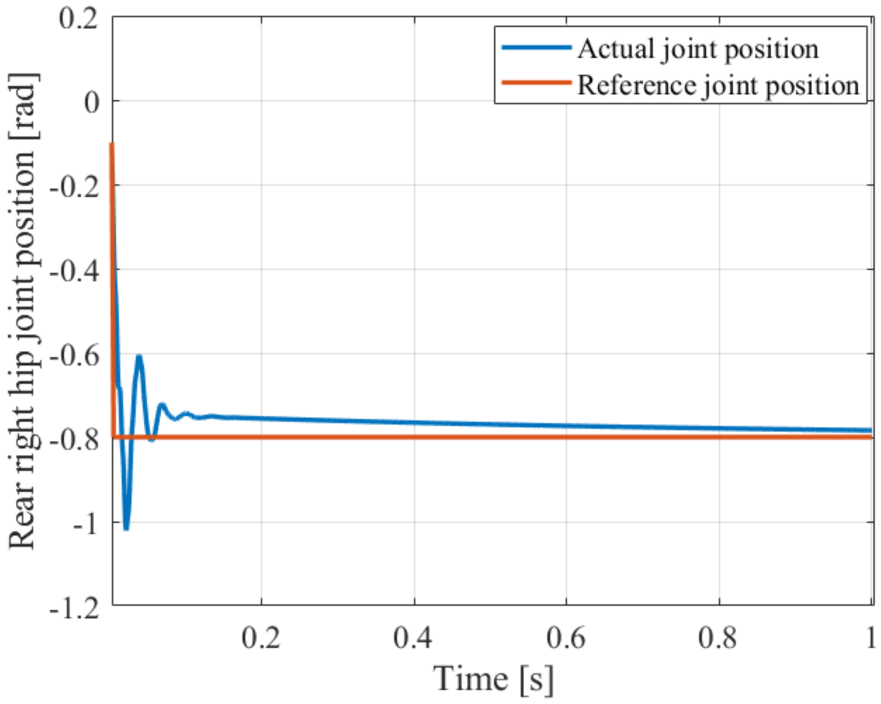

FIGURE 5.6: Rear right hip joint actual position (blue) and reference position (red) with posture control algorithm.

rad. Figure 5.13 presents the actual and reference positions of the rear spine joint. Figures 5.14 and 5.15 present the actual positions and reference trajectories of the right and left hip adduction/adduction joint positions, respectively.

The roll, pitch, and yaw angles of the robot with respect to the world coordinate frame can be seen in Figure 5.16. The intended angular position for the robot to attain after the external disturbance is selected to be zero radians along the x, y, and z-axes. Maximum inaccuracy is roughly 0.009 rad around the x-axis, 0.001 rad along the y-axis, and 0.0005 rad around the z-axis, according to the results.

Identical disturbances are applied on the quadrupedal robot model without a push recovery mechanism in order to demonstrate the effect of the method. The angular velocity around the x-axis with and without this technique can be seen in Figure 5.17. The angular velocity of the robot rises in the absence of the push recovery mechanism, eventually settling at 0.2 rad/s. When the push recovery method is turned on, however, the angular velocity does not exceed 0.2 rad/s, and instead falls to a value near 0 rad/s.

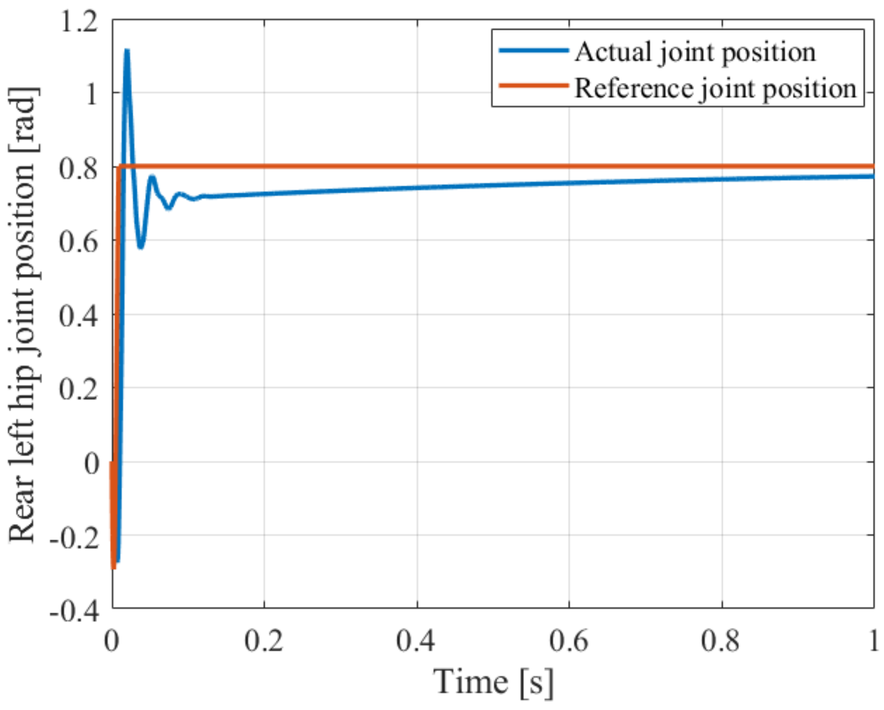

FIGURE 5.7: Rear left hip joint actual position (blue) and reference position (red) with posture control algorithm.

Similarly, without the push recovery mechanism, angular velocity along the y-axis approaches 0.2 rad/s since angular accelerations are identical. It peaks to 0.05 rad/s when applying the method and converges to zero after 0.2 seconds (Figure. 5.18). Finally, angular velocity around the z-axis is presented in Figure 5.19 with and without the push recovery technique. Angular velocity does not increase up to 0.2 rad/s when applying the technique, and settles to around zero rad/s in 0.1 seconds, as it did in prior findings.

5.3 Discussion

This section provides a novel posture control algorithm for legged robot jumping gaits for the flight phase. Angular momentum variables are used in the control process. Parts of the robot body (legs and torso pieces) are modelled with an independent mass.

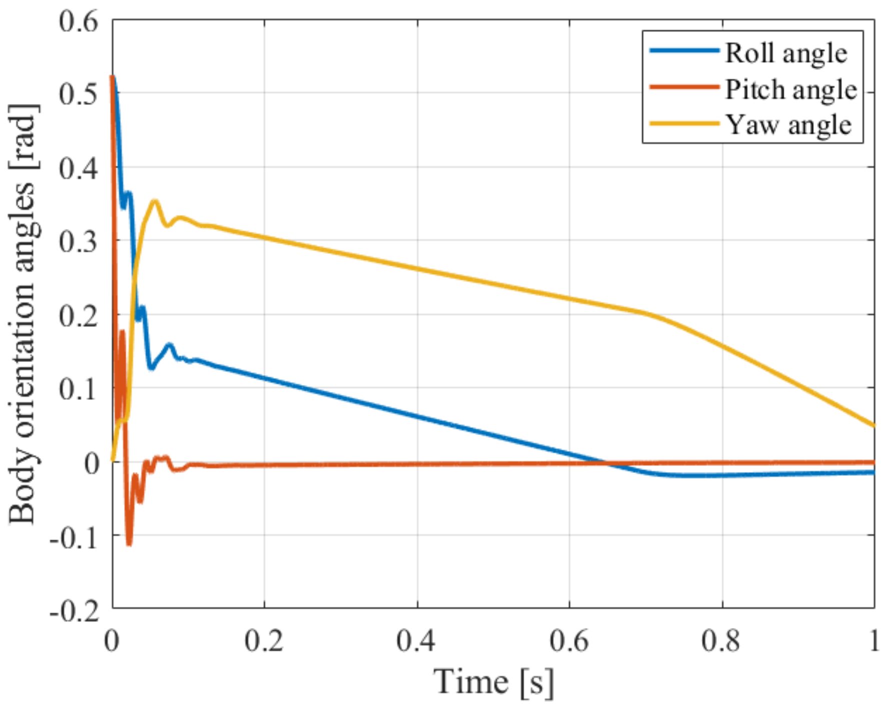

FIGURE 5.8: Angular positions (roll, pitch, and roll) of the quadrupedal robot around the world coordinate frame x-axis (blue), y-axis (red), and z-axis (yellow) with posture control algorithm.

The model is detailed, taking into account individual link masses, and it is computationally efficient enough to be employed in real-time applications.

When the front spine joint moves, two torso components, and two leg components move with it. As a result, the front body joint moves a mass of 50 kg (middle body is 10 kg, rear body is 20 kg and each leg is 10 kg), while the rear spine joint moves a mass of 40 kg. The rear leg joints, on the other hand, move 20 kg for two legs. The performance of controlling angular position around the z-axis is weaker than the x and y-axes because the moment of inertia is a function of mass, and rear leg joints are chosen to regulate angular position around the z-axis.

In the flight phase of a long jump, the suggested posture control method is tested in a simulated environment, including initial body orientation conditions. The technique achieves minor orientation errors, indicating that it is feasible and appropriate for implementation as a posture control method for free-flying robots.

Table 5.2: Locomotion parameters for push recovery method.

Locomotion Parameters		
Definition	Symbol	Value (Unit)
Take-off velocity about x-axis	v_x	5 (m/s)
Take-off velocity about y-axis	v_y	0 (m/s)
Take-off velocity about z-axis	v_z	5 (m/s)
Initial body orientation angles (Roll - Pitch - Yaw)	v_z	$0-0-0$ (rad)
Flight phase	t_f	1 (s)
Disturbance torque around x-axis		28.7 (Nm)
Disturbance torque around y-axis		76.6 (Nm)
Disturbance torque around z-axis		102 (Nm)

Finally, the suggested method is also tested in a simulated environment including external disturbances. The technique achieves is successful in orientation reference tracking, again validating that it is suitable for implementation.

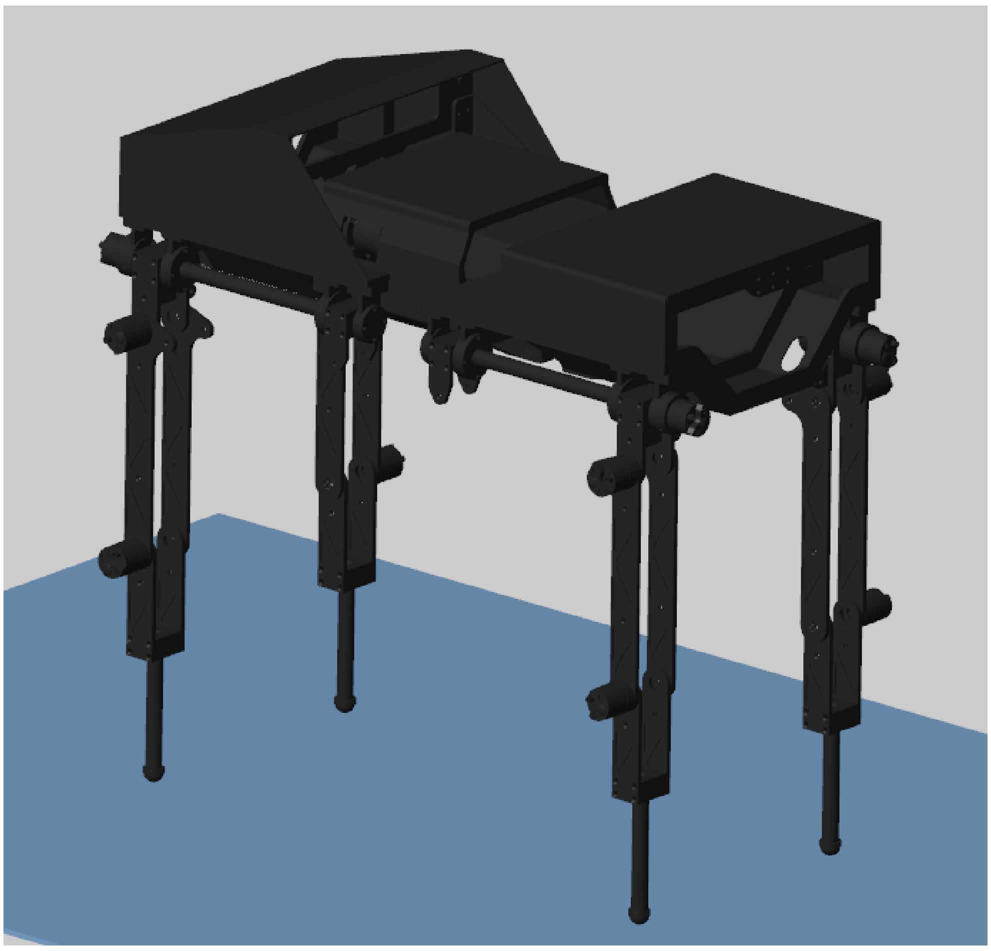

FIGURE 5.9: Quadrupedal robot model in simulation environment.

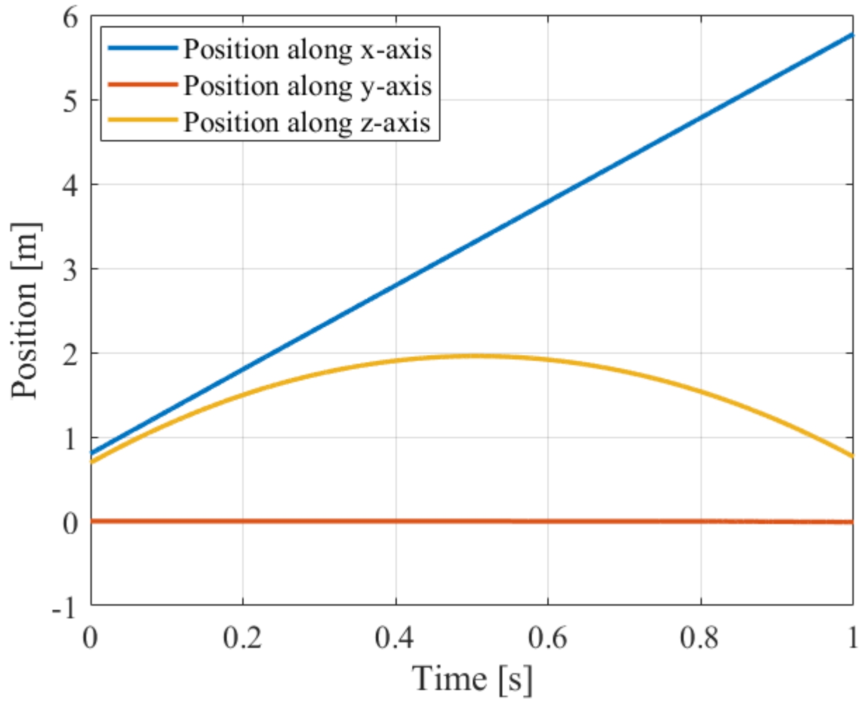

FIGURE 5.10: Linear positions of the quadrupedal robot performing jumping gait along the x-axis (blue), y-axis (red), and z-axis (yellow).

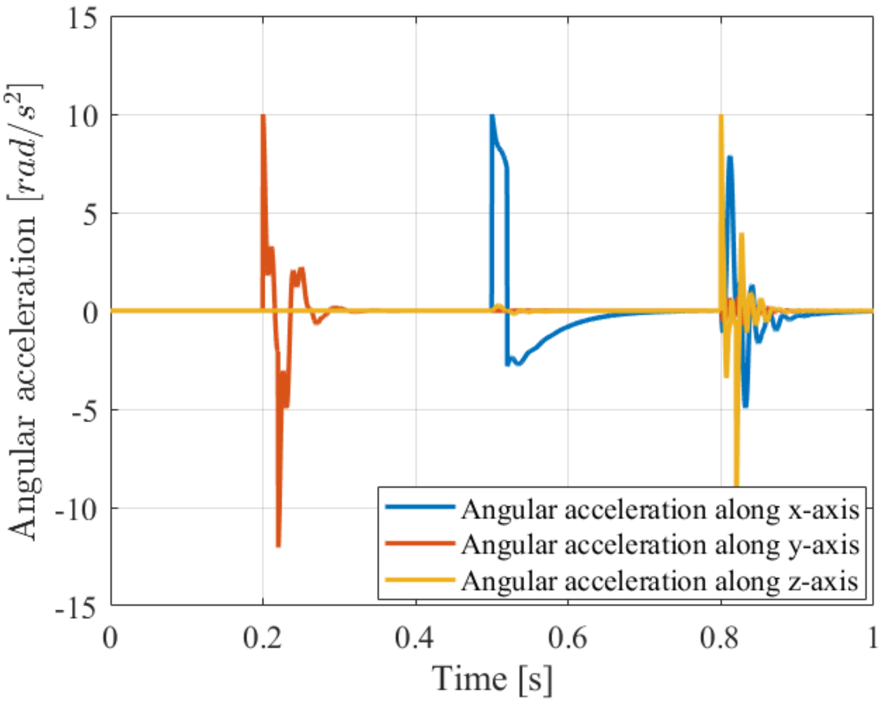

FIGURE 5.11: Angular accelerations of the quadrupedal robot around the x-axis (blue), y-axis (red), and z-axis (yellow) with push recovery mechanism.

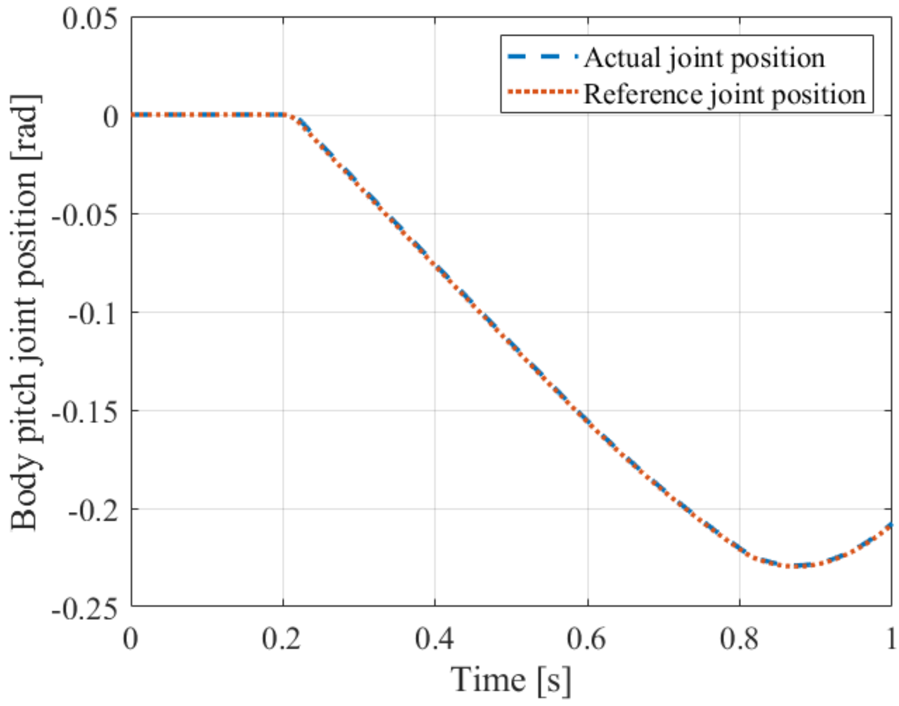

FIGURE 5.12: Front spine joint (body pitch joint) actual position (blue) and reference position (red) with push recovery mechanism.

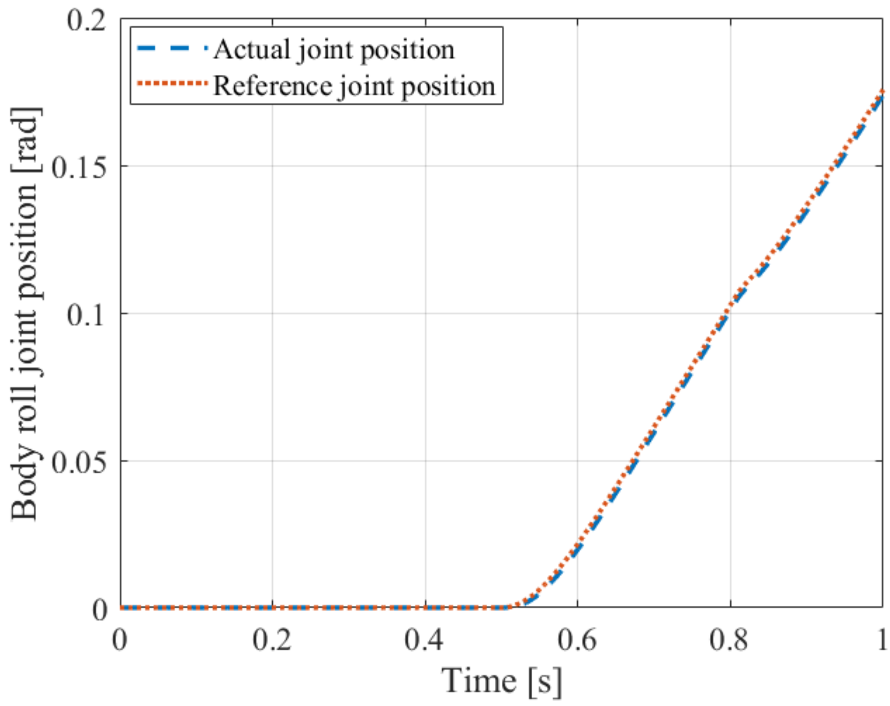

FIGURE 5.13: Rear spine joint (body roll joint) actual position (blue) and reference position (red) with push recovery mechanism.

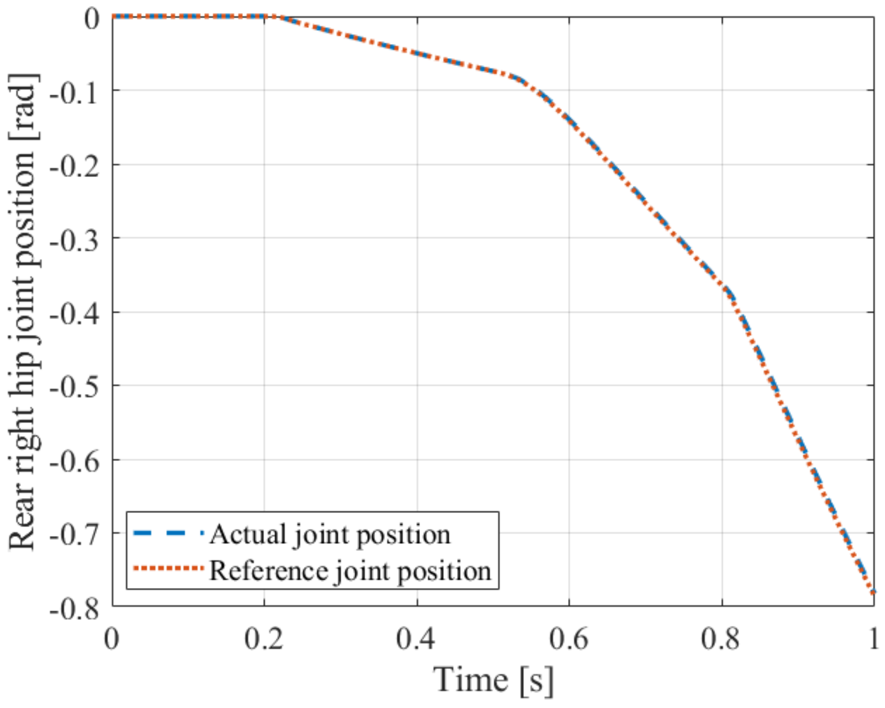

FIGURE 5.14: Rear right hip joint actual position (blue) and reference position (red) with push recovery mechanism.

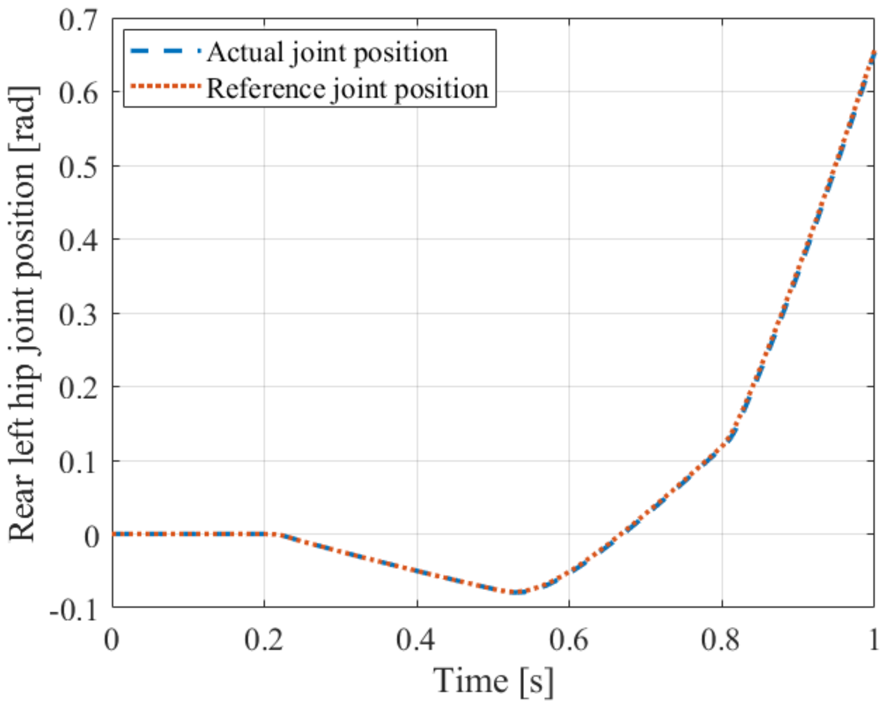

FIGURE 5.15: Rear left hip joint actual position (blue) and reference position (red) with push recovery mechanism.

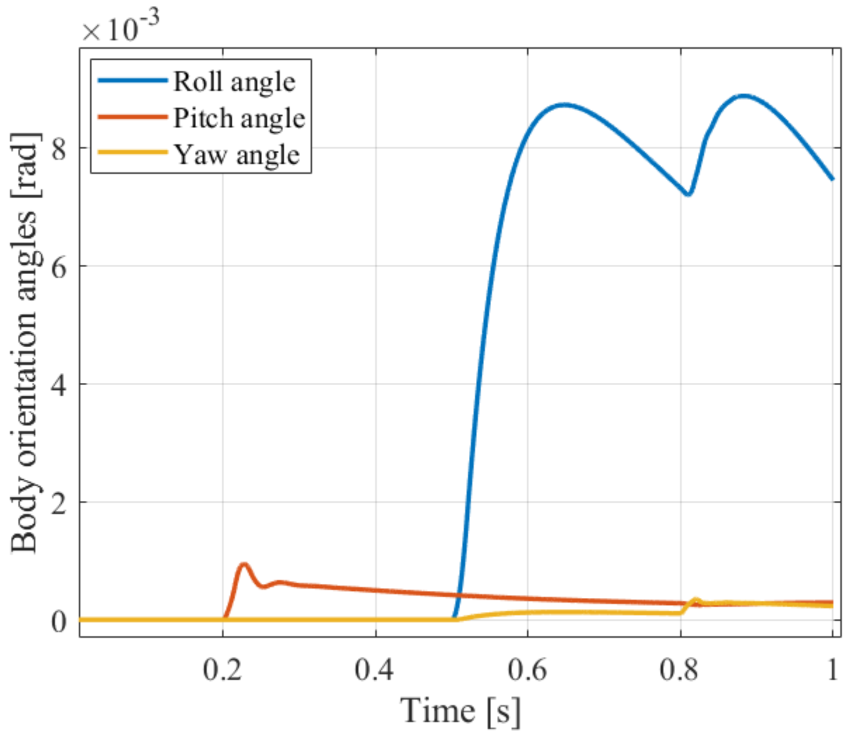

Figure 5.16: Angular positions (roll, pitch, and roll) of the quadrupedal robot around the world coordinate frame x-axis (blue), y-axis (red), and z-axis (yellow) with push recovery mechanism.

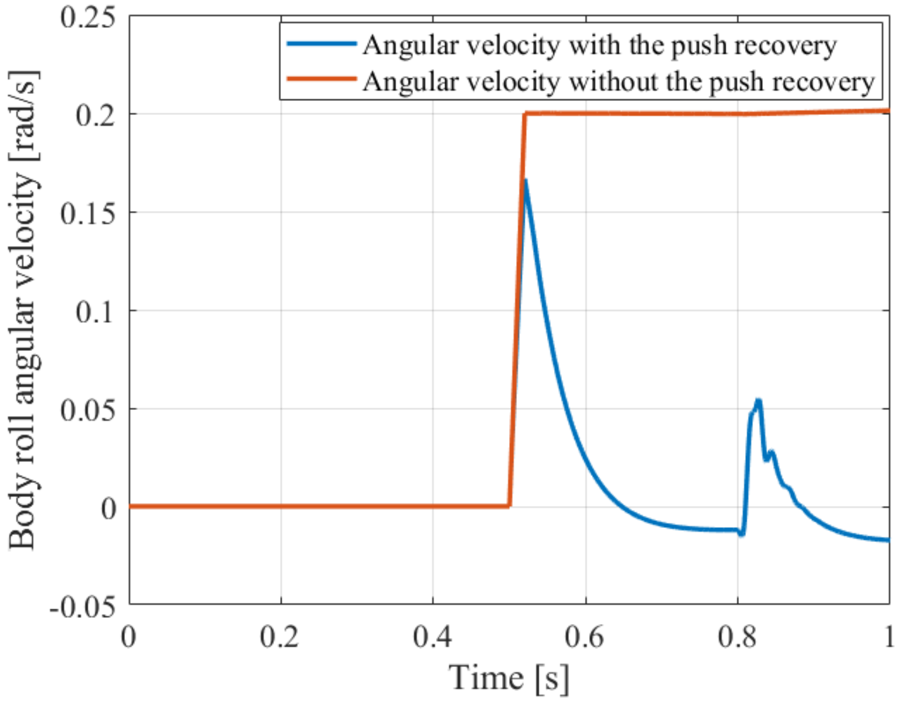

FIGURE 5.17: Angular velocity around the x-axis, the blue line shows results with the push recovery mechanism, the red line is results without push recovery mechanism.

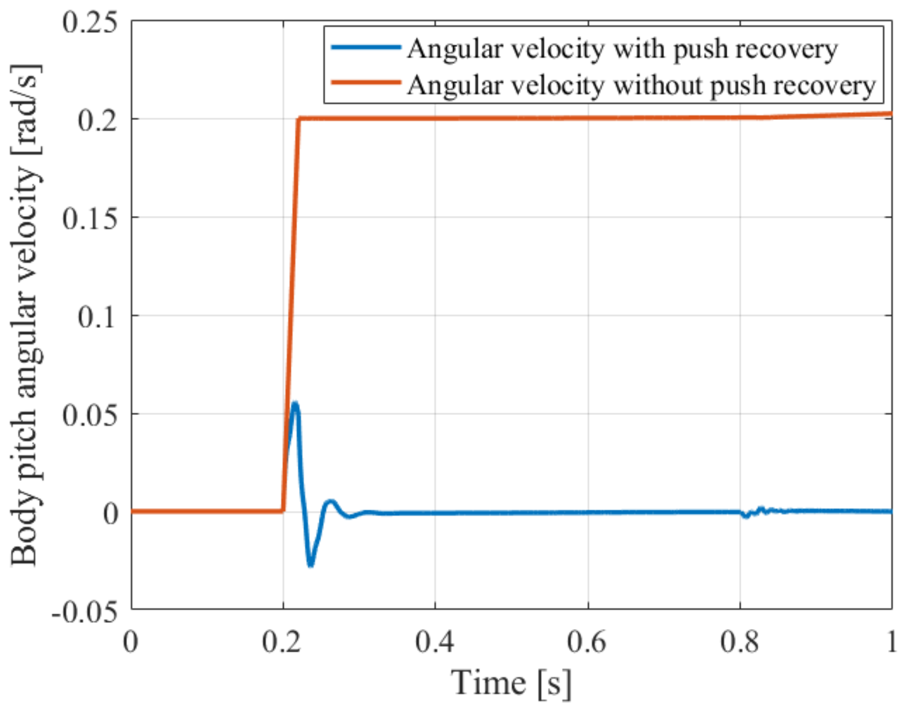

FIGURE 5.18: Angular velocity around the y-axis, the blue line indicates results with the push recovery mechanism, the red line is results without push recovery mechanism.

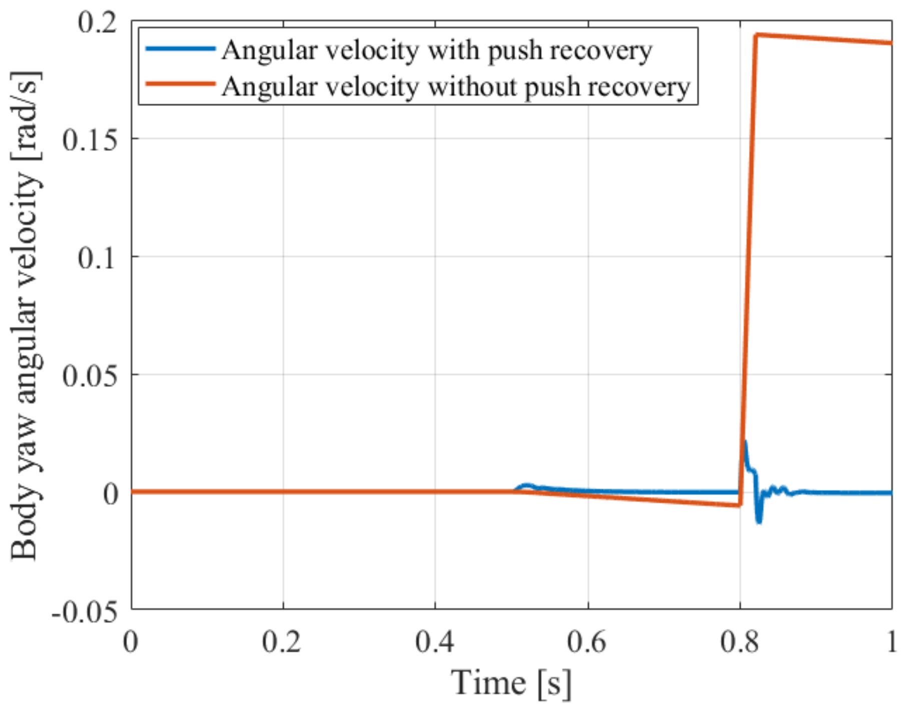

FIGURE 5.19: Angular velocity around the z-axis, the blue line shows results with the push recovery mechanism, the red line is results without push recovery mechanism.

Chapter 6

Reference Trajectory Generation Using Angular Momentum with Real-time Centroidal Dynamics Computation for Posture Control of Free Flying Robots

A novel reference generation approach based on angular momentum for in-air stabilization of robots that lack ground contact sites during certain gait phases is considered in this chapter. This technique provides references for a small set of generalized coordinates and speeds that need another small set of coordinates and speeds (such as the orientation of the robot in flight). The aim of the approach is to reach a specified reference value swiftly and in a stable way at the end of flight. Compared to chapter 5, this algorithm computes the inertias of each robot link in real time at each time step of the simulation. The continuous time rate of change equation for angular momentum is discretized and linearized. A minimum set of linear equations for reference trajectory construction is obtained. This technique is more applicable than techniques involving the addition of a gyroscope or a tail to the robot. As with the methods in the previous chapter, the proposed algorithm and the resulting program make use of a precise model of the robot in a computationally efficient way that is suitable for real-time. Because the angular momentum equations are linearized, the reference generation approach utilizes an arbitrarily detailed model of the robot with no influence on computing performance. A fast and reliable approach is proposed to desired reference orientation parameters for landing employing the suggested method during the jumping phase of a quadrupedal robot.

6.1 The Methodology of the Posture Control algorithm

6.1.1 General Framework

In classical Lagrangian dynamics, a mechanical system can represented as a point in a $2N+1$-dimensional phase space composed of N generalized coordinates θ_i, N corresponding generalized speeds $\dot{\theta}_i$ ($i = 1...N$), and one time coordinate t. For simple external forces, the Lagrangian equations of motion are first order in time derivatives [136]. The corresponding discrete-time version of the equations of motion requires just two lowest-order time slices, one at (say) time t and another at time $t + \Delta t$, where Δt is the time step size. Thus, the discretized equations of motion require a total of $4N$ generalized coordinates and speeds ($2N$ at time t and another $2N$ at time $t + \Delta t$), necessitating the solution of $4N$ independent equations. For a mechanical system, the number of equations might be massive, even analytically unsolvable. In reality, however, many of these generalized coordinates and speeds are either known explicitly (through sensor data, for example) or are restricted by construction or by external forces. For instance, the quadrupedal robot flight phase model utilized in the following sections has $N = 17$. However, sensors monitor the generalized coordinates and speeds at time t, and the COM motion may be isolated from the rest of the problem, making the problem considerably more tractable. There are just a few (potentially extremely complicated) equations of motion remaining to solve.

This chapter proposes a technique for solving the remaining equations of motion. The solution would provide the unknown values of a specific subset of generalized coordinates and speeds at time $t + \Delta t$, pushing all of the generalized coordinates and speeds closer to their reference values at the end of flight. The strategy is based on the assumption that all generalized coordinates and speeds at time t (corresponding to $2N$ of the $4N$ variables) are known from sensor data. The method attempts to solve for a small number $n < N$ of generalized coordinates (or, equivalently, generalized speeds) at time $t + \Delta t$, if (i) the remaining $2N - n$ generalized coordinates and speeds are given by a known trajectory function and (ii) all generalized coordinates and speeds are determined self-consistently, i.e., the generalized speed $\dot{\theta}_i$ i is related to the change in generalized coordinates via $\dot{\theta}_i(t + \Delta t) = \frac{(\theta_i(t+\Delta t) - \theta_i(t))}{\Delta t}$. The remaining n equations are then linearized by Taylor expanding the unknown coordinates or speeds at time $t + \Delta t$ around their values at time t, retaining only terms of first order. Thus, the problem is simplified to a linear set of n-rank equations. Hence given the desired trajectory of the remaining coordinates, the trajectory of a limited number of coordinates is constructed. The application of this strategy is constrained by the piecewise applicability of Taylor expansion to trajectory functions. As a result, Δt must be less than the Taylor expansion's radius of convergence. This constraint would necessitate that the time slices in the simulation are small enough, possibly leading to a greater number of simulation cycles. Since the number of slices to compute in a given time is increased, this condition would require fast computing for real-time applications.

The collection of generalized coordinates and speeds to solve for, and hence the value of n, are determined by the task at hand. In the example provided in this work, the goal is to ensure that the quadrupedal robot's total orientation in three-dimensional space at the conclusion of the flying phase, approaches a specific reference orientation. Thus, the corresponding generalized coordinates are the three ($n = 3$) orientation angles between the robot body coordinates and the world coordinates. (Specifically, the body roll, pitch, and yaw angles are employed. See Figure 5.1) The corresponding equation of motion obtained via Lagrangian mechanics utilizing these generalized coordinates is simply the equation of motion for the system's total angular momentum, defined as $\frac{d\vec{L}}{dt} = \vec{\tau}$, where $\vec{L}$ denotes the total angular momentum of the robot and $\vec{\tau}$ stands for the the external torque acting on the robot [136]. Due to the fact that this is a three-dimensional vector equation, we have $n = 3$ equations to solve for the three unknowns. It should be noted that the conventional parallel axis method for computing the moment of inertia tensor I_{ij} does not yield a simple solution in this case, simply because our robot model contains numerous joints that can rotate in any direction and the axes around which the rotation occurs are not parallel [137]. As a result, this approach is not "simpler" to compute than the conventional integration method explained in (6.12) and (6.13).

In order to design a system trajectory in which the final body orientation approaches a reference set of angles, the following equation is employed as a starting point:

$$\begin{aligned} &\frac{d\vec{L}(t)}{dt} = \vec{\tau}(t), \\ &\vec{L}(t) \to \vec{L}(\theta_j(t),\ \dot{\theta}_j(t) \mid j = 1...N). \end{aligned} \tag{6.1}$$

Here $\theta_j(t)$ and $\dot{\theta}_j(t)$ are generalized coordinate and speed vectors. $\vec{L}$ and $\vec{\tau}$ are the total angular momentum and torque vectors, respectively. N is the total number of generalized coordinates. Following that, (6.1) is discretized in time:

$$\vec{L}(t + \Delta t) - \vec{L}(t) = \vec{\tau}\ \Delta t, \tag{6.2}$$

In (6.2), t is the time at which discretization takes place. After linearizing Equation 6.2, the resultant equation has a maximum rank of three. This enables the identification of $n = 3$ unknown parameters. It is expected that at time $t + \Delta t$,

$N - n = N - 3$ coordinates and speeds may be extrapolated from their values at time t:

$$\begin{aligned} \theta_j(t+\Delta t) &= f_j(\{\theta_i(t),\ \dot{\theta}_i(t)\ ;\quad i = 1\ldots N\}), \\ \dot{\theta}_j(t+\Delta t) &= g_j(\{\theta_i(t),\ \dot{\theta}_i(t)\ ;\quad i = 1\ldots N\}). \end{aligned} \tag{6.3}$$

Here the label j denotes extrapolated coordinates and speeds that range from 1 to $N-3$. f_j and g_j are known trajectory functions corresponding to the desired phase space trajectory. f_j and g_j must be self-consistent for the functions g_j to reflect the time derivative of the generalized coordinates specified by f_j. At this point, $2N$ out of $4N$ variables are known from sensor data, and $2(N-n)$ more variables may be extrapolated straightaway via Equation 6.3. $2n = 6$ more equations are required to get a unique solution. A potential simplification is to assume that the extrapolation functions are dependent on the coordinate or speed:

$$\begin{aligned} \theta_j(t+\Delta t) &= f_j(\{\theta_i(t),\ \dot{\theta}_i(t);\quad i = j\}), \\ \dot{\theta}_j(t+\Delta t) &= g_j(\{\theta_i(t),\ \dot{\theta}_i(t);\quad i = j\}), \end{aligned} \tag{6.4}$$

where the labels j of the extrapolated coordinates and speeds range from 1 to $N-3$ (without sacrificing generality). The following study chooses a linear and self-consistent form for f_j and g_j:

$$\begin{aligned} \theta_j(t+\Delta t) &= \theta_j(t) + \frac{\theta_j^{ref} - \theta_j^{sens}}{N_T - n_t}, \\ \dot{\theta}_j(t+\Delta t) &= \frac{\theta_j^{ref} - \theta_j^{sens}}{(N_T - n_t)\ \Delta t}. \end{aligned} \tag{6.5}$$

Here, θ_j^{ref} is the reference for generalized coordinates, θ_j^{sens} represents sensor data for generalized coordinates at time t, N_T is the total number of steps, and $n_t = 0 \ldots N_T - 1$ stands for the current time step. All $\theta_j(t)$, $\dot{\theta}_j(t)$ for $j = 1 \ldots N$ and $\theta_j(t+\Delta t)$, $\dot{\theta}_j(t+\Delta t)$ for $j = 1 \ldots N-3$ are known at this point. Following that, (6.2) is discretized as

$$\vec{L}(\{\theta_k(t+\Delta t),\ \dot{\theta}_k(t+\Delta t)\}) - \vec{L}(\{\theta_k(t),\ \dot{\theta}_k(t)\}) = \vec{\tau}\ \Delta t, \tag{6.6}$$

where $k \in \{N-2, N-1, N\}$. For simplicity, just the reliance on θ_k and $\dot{\theta}_k$ are retained here. Self-consistency requires that these generalized coordinates and speeds have a first-order relationship:

$$\begin{aligned} \theta_k(t+\Delta t) &= \theta_k(t) + d\theta_k, \\ \dot{\theta}_k(t+\Delta t) &= \frac{\theta_k(t+\Delta t) - \theta_k(t)}{\Delta t} = \frac{d\theta_k}{\Delta t}. \end{aligned} \tag{6.7}$$

In (6.7), k has one of the following values: $N-2$, $N-1$, or N. $d\theta_k$ is a small increment in the coordinate θ_k from time t to time $t+\Delta t$. While the second line of (6.7) seems to include six new equations, the first line merely specifies the unknown $d\theta_k$'s. As a result, three equations must still be solved to get the values for these three unknowns. The vector equation (6.6) includes these three equations.

Additional simplification is achieved by linearizing the angular momentum at time $t+\Delta t$. The first term in (6.6) is expressed in $d\theta_k$ up to first order as:

$$\begin{aligned} &\vec{L}(\{\theta_k(t+\Delta t),\ \dot{\theta}_k(t+\Delta t)\}) \cong \vec{L}(\{\theta_k(t) + d\theta_k,\ \frac{d\theta_k}{\Delta t}\}) \\ &\cong \vec{L}(\{\theta_k(t), 0\}) + \sum_{k=N-2}^{N} d\theta_k \left(\frac{\partial \vec{L}(\{\theta_k(t) + d\theta_k,\ \frac{d\theta_k}{\Delta t}\})}{\partial d\theta_k} \right)_{d\theta_k = 0}. \end{aligned} \tag{6.8}$$

It is important to note that, with the exception of $d\theta_k$'s in the last line of (6.8), every quantity has a known value. Once the configuration, generalized coordinates, and generalized speeds of a robot are known, it is expected that the angular momentum may be determined easily.

In order to complete the equation of motion, the term on the right hand side of (6.6), denoted by $\vec{\tau}\Delta t$, must be determined as well. Internal torques cancel out due to Newton's third law, and the net torque applied to a robot in the air is due to gravity. (Note that the $\vec{L}$ in $\frac{d\vec{L}(t)}{dt}$ denotes the total angular momentum in world coordinates). Therefore

$$\vec{\tau}(t) = \vec{r}^{COM}(t) \times \vec{F_g}, \tag{6.9}$$

where $\vec{r}^{COM}(t)$ denotes the COM coordinates at time t. $\vec{F_g} = m_{total}\,\vec{g}$ with the total mass m_{total} of the system. $\vec{g}$ denotes the gravitational acceleration. $\vec{r}^{COM}(t)$ may be determined from the sensor data for a robot with a particular configuration $\{\theta(t), \dot{\theta}(t)\}$. As a result, $\vec{\tau}(t)\Delta t$ can also computed at this stage.

Finally, by combining (6.6), (6.7) and (6.9) the component of the equation of motion related to angular momentum may be written as follows:

$$\sum_{k=N-2}^{N} \left.\frac{\partial \vec{L}_i}{\partial d\theta_k}\right|_{d\theta_k=0} d\theta_k = \vec{\tau}_i(t)\;\Delta t + L_i(t) - L_i(\theta_k, 0). \tag{6.10}$$

This is a linear three-dimensional vector equation in which $A_{i,k} \times d\theta_k = B_i$ with the three unknown $d\theta_k$'s. Thus, one may define (6.10) in matrix form with regard to the x, y, and z components of the angular momentum vector as $A_{i,k} \times d\theta_k = B_i$, where $i = x, y, z$. Explicitly,

$$\begin{bmatrix} A_{N-2,x} & A_{N-1,x} & A_{N,x} \\ A_{N-2,y} & A_{N-1,y} & A_{N,y} \\ A_{N-2,z} & A_{N-1,z} & A_{N,z} \end{bmatrix} \begin{bmatrix} d\theta_{N-2} \\ d\theta_{N-1} \\ d\theta_N \end{bmatrix} = \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix}. \tag{6.11}$$

Solving for $\{d\theta_{N-2}, d\theta_{N-1}, d\theta_N\}$ completes the creation of a self-consistent trajectory for time $t + \Delta t$. It should be emphasized that all computations up to this point are performed using basic arithmetic functions, and (6.11) requires the solution of a 3×3 linear system. As a result, it is believed that the computation of $d\theta_k$'s, and hence the robot's trajectory from t to $t + \Delta t$, will be rapid enough to be employed in a real-time application.

6.1.2 Application on a quadrupedal robot

In this section, the linearized angular momentum approach of trajectory generation described above is applied to a quadrupedal robot. A linear approach is utilized to determine the reference value of a generalized coordinate. The purpose of trajectory generation is to achieve the desired overall body orientation of a quadruped robot in the air. In order to simplify the computation of angular momentum, all of the robot's linkages are assumed to have uniform stick shapes, except for the rear body, which is assumed to be a uniform plane. Figure 5.1 illustrates the concept of a quadruped robot used to compute angular momentum. Parameters of the robot model used in the simulation and reference generation algorithm are given in Table 3.1.

Table 6.1 contains nomenclature for robot leg coordinates. The legs of the robot have 12 DOF while the spine have two DOF. θ_b and ϕ_b are the generalized coordinates for spinal joints. Along with generalized leg coordinates, the robot has three general orientation coordinates denoted by the letters α, β, and γ, which correspond to roll, pitch, and yaw angles, respectively (Figure 5.1). Also, the robot has three position coordinates along x, y and z axis of the world coordinate frame. Therefore, the robot has a total of 20 generalized coordinates. However, the initial position coordinates and initial velocity values are known, therefore the position and velocity for the center of robot coordinate axis at any time can be trivially calculated. The remaining number of generalized coordinates to be considered is $N = 17$. Hence, this model has $4N = 68$ unknowns, which correspond to generalized coordinates at time t, speeds at time t, coordinates at time $t + \Delta t$, and speeds at time $t + \Delta t$. Generalized coordinates and speeds are known at time t by sensor data; hence, $2N = 34$ unknowns remain.

Extrapolation is performed on the spine joint coordinates θ_b, ϕ_b, and the leg coordinates θ_{ulrr}, θ_{ullr} (Figure 5.1). However, since the equation of motion contains three components, a maximum of three unknowns can be extrapolated. θ_{ulrr} and θ_{ullr} are constrained to have identical magnitude and opposite sign in order to apply maximum precession of the total angular momentum around the z-axis of the robot's coordinate frame. The remaining unknowns are assumed to be approaching to the reference value of the coordinate (see (6.5)) via a known function (linearly, in our case). Assuming all robot parts have uniform density, the angular momentum of one-dimensional robot parts is computed as

TABLE 6.1: Leg Coordinate Names

	Front		Rear	
Right	Upper	θ_{ulrf}	Upper	θ_{ulrr}
		ϕ_{ulrf}		ϕ_{ulrr}
	Lower	θ_{llrf}	Lower	θ_{llrr}
Left	Upper	θ_{ullf}	Upper	θ_{ullr}
		ϕ_{ullf}		ϕ_{ullr}
	Lower	θ_{lllf}	Lower	θ_{lllr}

$$\vec{L}_1(t) = \sum_l m_l \int_{s=0}^{1} \Big(\vec{r}_l(s,t) \times \dot{\vec{r}}_l(s,t) \Big)\, ds, \tag{6.12}$$

where m_l denotes the mass of the robot component with the index l. $\vec{r}_l(s,t)$ tracks the spatial coordinates of the robot component identified by the same index. s is a positive integer between 0 and 1. $\vec{r}$ is computed using joint positions and leg dimensions. For a two-dimensional body component, the equation for angular momentum becomes

$$\vec{L}_2(t) = m \int_{s_1=0}^{1} \int_{s_2=-1/2}^{1/2} \Big(\vec{r}(s_1,s_2,t) \times \dot{\vec{r}}(s_1,s_2,t) \Big)\, ds_1\, ds_2, \tag{6.13}$$

where s_1 and s_2 are normalized line segments along the two dimensions of the component.

The total angular momentum is defined as $\vec{L} = \vec{L}_1 + \vec{L}_2$. In order to find the RHS of (6.6) as determined by (6.9), $\vec{r}^{COM}(t)$ must be obtained. It can be computed in the following expression:

$$\vec{r}^{COM}(t) = \frac{\sum_l m_l\, \vec{r}_l\, \left(s = \frac{1}{2}, t\right)}{\sum_l m_l}. \tag{6.14}$$

6.1.3 Generalized Rotation Matrices

Rotation matrices are widely employed for angular momentum calculations. When the parameters of a rotation matrix change by small amounts, one can simplify the functional form of this change by Taylor-expanding the rotation matrix around the original value of the changing parameter and keeping only up to the first-order terms. This linearizes the dependence on the change in the parameter. Generalized rotation matrices (GRM) introduced in this section and used in the trajectory generation code is a tool to simplify these calculations. Moreover, all the parameters and time dependence used to compute the angular momenta are on the rotation matrices: One multiplies the rotation matrix (or the time-derivative of the rotation matrix) by the constant and known home-position of the various components to calculate the position and velocity vectors of the robot components, from which one can calculate the angular momentum.

In this work, the rotation matrices used can depend on many parameters (the angular degrees of freedom, $\{\theta_i\,; i = 1...N\}$). Only a small number (three in this case, θ_{N-2}, θ_{N-1} and θ_N) will be linearized, as explained in the previous sections. When θ_i change by a small amount $\theta_i \to \theta_i + d\theta_i$ $(i = N-2, N-1, N)$, the rotation matrix also changes as below.

$$\begin{aligned}
&R(\theta_{N-2} + d\theta_{N-2}, \theta_{N-1} + d\theta_{N-1}.\theta_N + d\theta_N) \\
&\quad = R(\theta_{N-2}, \theta_{N-1}, \theta_N) + \frac{\partial R(\theta_{N-2}, \theta_{N-1}, \theta_N)}{\partial \theta_{N-2}} d\theta_{N-2} \\
&\qquad + \frac{\partial R(\theta_{N-2}, \theta_{N-1}, \theta_N)}{\partial \theta_{N-1}} d\theta_{N-1} + \frac{\partial R(\theta_{N-2}, \theta_{N-1}, \theta_N)}{\partial \theta_N} d\theta_N \\
&\qquad + \mathscr{O}(2).
\end{aligned} \tag{6.15}$$

Here, the dependence of the rotation matrix on $\theta_i (i = 1...N-3)$ is suppressed for convenience. The form of the linearized rotation matrix in (6.15) is condensed to a GRM. A GRM has 4 components, each of which is a 3×3 matrix. The first component is the zeroth order term in (6.15) and the next three components are the three first-order terms. The GRM corresponding to the rotation matrix in (6.15) can be written as:

$$\begin{aligned} GRM(R) &= \Big[R(\theta_{N-2}, \theta_{N-1}, \theta_N),\ \frac{\partial R(\theta_{N-2}, \theta_{N-1}, \theta_N)}{\partial \theta_{N-2}}, \\ &\qquad \frac{\partial R(\theta_{N-2}, \theta_{N-1}, \theta_N)}{\partial \theta_{N-1}},\ \frac{\partial R(\theta_{N-2}, \theta_{N-1}, \theta_N)}{\partial \theta_N}\Big] \\ &\equiv \left[R^{(0)},\, R^{(1)},\, R^{(2)},\, R^{(3)}\right]. \end{aligned} \tag{6.16}$$

We note for subsequent use that not only the rotation matrices, but also the linearized form of their time derivatives can be written as a GRM.

The addition of two GRMs corresponding to rotation matrices R_1 and R_2 yields another GRM as follows:

$$\begin{aligned} GRM(R_1 + R_2) &= GRM(R_1) + GRM(R_2) \\ &= \left[R_1^{(0)},\, R_1^{(1)},\, R_1^{(2)},\, R^{(3)_1}\right] + \left[R_2^{(0)},\, R_2^{(1)},\, R_2^{(2)},\, R_2^{(3)}\right] \\ &= \left[R_1^{(0)} + R_2^{(0)},\, R_1^{(1)} + R_2^{(1)},\, R_1^{(2)} + R_2^{(2)},\, R_1^{(3)} + R_2^{(3)}\right]. \end{aligned} \tag{6.17}$$

This result can be verified using (6.15) and (6.16).

The GRM of the product of two rotation matrices $R_1.R_2$ is given by

$$\begin{aligned} GRM(R_1.R_2) &= \left[R_1^{(0)},\, R_1^{(1)},\, R_1^{(2)},\, R^{(3)_1}\right] . \left[R_2^{(0)},\, R_2^{(1)},\, R_2^{(2)},\, R_2^{(3)}\right] \\ &= \Big[R_1^{(0)}.R_2^{(0)},\, R_1^{(0)}.R_2^{(1)} + R_1^{(1)}.R_2^{(0)}, \\ &\qquad R_1^{(0)}.R_2^{(2)} + R_1^{(2)}.R_2^{(0)},\, R_1^{(0)}.R_2^{(3)} + R_1^{(3)}.R_2^{(0)}\Big]. \end{aligned} \tag{6.18}$$

In order to obtain this result, one starts from (6.15) and multiplies two matrices as usual, keeping track of the order of multiplication and keeping only the terms up to first order in $d\theta_i$ $(i = N-2, N-1, N)$.

We also introduce the concept of a generalized vector (GV), which is used to keep track of the result of the application of a 3×3 rotation matrix on a 3×1 vector. The resulting form is an ordered 4-tuple of 3×1 vectors. (In the following as well as in the computational work, a GV may sometimes be condensed into a single 3×4 matrix, formed by the concatenation of the components of the GV.) The generalized vector corresponding to the product of a generalized rotation matrix $GRM(R)$ by a vector $\vec{c} = [c_x, c_y, c_z]^T$ is defined as

$$\begin{aligned} GV(R\,.\,\vec{c}) &\equiv GRM(R)\,.\,\vec{c} \\ &= \left[R^{(0)},\, R^{(1)},\, R^{(2)},\, R^{(3)}\right]\,.\,\vec{c} \\ &\equiv \left[R^{(0)}.\,\vec{c},\, R^{(1)}.\,\vec{c},\, R^{(2)}.\,\vec{c},\, R^{(3)}.\,\vec{c}\right] \quad . \end{aligned} \tag{6.19}$$

The multiplications in the last line of (6.19) are ordinary multiplications between a 3×3 matrix and a 3×1 vector. This definition follows naturally from (6.15). The usefulness of the generalized vector concept is limited to keeping track of the multiplication between a matrix and a vector. While the GV notion in itself does not provide any simplifications and shortcuts, it is handy where GRMs are involved.

The addition of two GVs to yield a 1×4 vector and cross product of two GVs to yield another GV are given below.

$$\begin{aligned} &GV_1 = \vec{c}_1^{(0)} + \vec{c}_1^{(1)} d\theta_{N-2} + \vec{c}_1^{(2)} d\theta_{N-1} + \vec{c}_1^{(3)} d\theta_N \\ &GV_2 = \vec{c}_2^{(0)} + \vec{c}_2^{(1)} d\theta_{N-2} + \vec{c}_2^{(2)} d\theta_{N-1} + \vec{c}_2^{(3)} d\theta_N \\ &GV_1 + GV_2 = \vec{c}_1^{(0)} + \vec{c}_2^{(0)} + (\vec{c}_1^{(1)} + \vec{c}_2^{(1)}) d\theta_{N-2} + (\vec{c}_1^{(2)} + \vec{c}_2^{(2)}) d\theta_{N-1} + (\vec{c}_1^{(3)} + \vec{c}_2^{(3)}) d\theta_N \\ &GV_1 \times GV_2 = \vec{c}_1^{(0)} \times \vec{c}_2^{(0)} + (\vec{c}_1^{(0)} \times + \vec{c}_2^{(1)} + \vec{c}_2^{(0)} \times + \vec{c}_1^{(1)}) d\theta_{N-2} \\ &\qquad + (\vec{c}_1^{(0)} \times + \vec{c}_2^{(2)} + \vec{c}_2^{(0)} \times + \vec{c}_1^{(2)}) d\theta_{N-1} \\ &\qquad + (\vec{c}_1^{(0)} \times + \vec{c}_2^{(3)} + \vec{c}_2^{(0)} \times + \vec{c}_1^{(3)}) d\theta_N . \end{aligned} \tag{6.20}$$

Here, GV_1 and GV_2 are two arbitrary GVs and $\vec{c}_m^{(n)}$ is mth column vector of nth GV. The results in (6.20) can again be verified by using (6.19) and keeping terms up to first order in $d\theta_i$.

6.1.4 Tree Structure

In computer science, a tree is a highly generic and efficient data structure that resembles a natural tree. It is composed of an ordered collection of connected nodes in a connected graph, where each node has a maximum of one parent node and zero or more ordered children nodes. If a node has zero children, it is called a leaf. There must always be a single 'top level' node referred to as the root. Then, given a node, any node on the subsequent 'down' level that is related to it through a branch is child of the node. In contrast, the node on the level above a non-root node is linked to the provided node (through an edge) is its parent [138]. Every non-root node has a single unique parent, and a root node has no parent.

The tree structure is employed in this algorithm to simplify calculations. This method involving the tree structure is applicable to legged robots since all links are physically attached to a previous link like a tree (see Figure 6.1). Each nodes in the tree structure correspond to robot links. The root node is the origin in the robot coordinate system, which is attached to the front body link in our system. The parent node-child node hierarchy is mapped to how the robot components are attached to each other. The leaf nodes (or the end nodes) are the lower legs in the quadrupedal robot considered. Just as in a tree data structure, each non-root node has a unique parent node but can have multiple child nodes. In the quadrupedal robot case, this means each robot part is attached to a single other robot part towards the origin, but it can have multiple parts attached to it when going away from the origin (see Figure 6.1).

The tree structure keeps the calculations relatively straight forward, mapping the calculated quantities to the robot structure. The tree structure shortens computations since all data from parent node is calculated and stored. In this way there is no need to calculate them again to obtain the data of the child node. For example, the overall rotation matrix related to a lower leg part is the product of the child node of the overall rotation matrix of the upper leg part attached (the parent node) times the rotation matrix describing just the lower leg part in its coordinate system. Similarly, the derivatives of the rotation matrices can also be calculated and stored in the same manner: Consider the time derivative of a rotation matrix R, given by $\frac{dR}{dt}$. In order to calculate this derivative we need to know partial derivatives of the rotation matrix with respect to joint angles such as $\frac{\partial R}{\partial \theta_1}$, $\frac{\partial R}{\partial \theta_2}$, $\frac{\partial R}{\partial \theta_3}$ to $\frac{\partial R}{\partial \theta_N}$ since joint angles are time dependent. Here, N is the total number of robot joint angles and body orientation angles, which for our case is equal to the number of DOF. Therefore, applying chain rule:

$$\frac{d}{dt}R(\theta_1(t), \theta_2(t)...\theta_N(t)) = \frac{\partial R}{\partial t} + \frac{\partial R}{\partial \theta_1}\dot{\theta}_1 + \frac{\partial R}{\partial \theta_2}\dot{\theta}_2 + \frac{\partial R}{\partial \theta_3}\dot{\theta}_3 + ... + \frac{\partial R}{\partial \theta_N}\dot{\theta}_N. \quad (6.21)$$

Here, we note all $\dot{\theta}(t)$s are known from sensor data. $\dot{\theta}_{N-2,N-1,N}(t+\Delta t)$ are approximated as $\frac{d\theta_{N-2,N-1,N}}{\Delta t}$ for the joint angles that are chosen to generate trajectory and other joint angles $\dot{\theta}_{1,\ldots,N-3}(t+\Delta t)$ have reference trajectories assigned to them and they are calculated as $\frac{\theta^{ref}-\theta^{sens}}{T-t}$. θ^{ref} is the reference angle, θ^{sens} is sensor data, T is total time steps and t is current time step. When tree structure is employed to calculate derivatives of rotation matrices these computations become simpler.

In the following paragraphs, we exemplify the usage of tree structure by calculating the rotation matrices and their time derivatives of a lower front leg piece. Derivative of a front lower leg rotation matrix (R_{lllf}) is calculated by multiplying rotation matrices for robot body orientation angles ($R(\alpha)$, $R(\beta)$ and $R(\gamma)$) two upper leg rotation matrices ($R(\theta_{ullf})$, $R(\phi_{ullf})$) and a lower leg rotation matrix $R(\theta_{lllf})$.

$$R_{llf} = R(\alpha)\, R(\beta)\, R(\gamma)\, R(\theta_{ulf})\, R(\phi_{ulf})\, R(\theta_{llf}). \quad (6.22)$$

Terms up to $R(\theta_{llf})$ will be called R_{upper} in the text to follow. When chain rule is applied,

$$\frac{dR_{llf}}{dt} = \frac{dR_{\text{upper}}}{dt}\, R(\theta_{llf}) + R_{\text{upper}}\, \frac{dR(\theta_{llf})}{dt}, \quad (6.23)$$

where $\frac{dR_{\text{upper}}}{dt}$ and R_{upper} are known at the time of this computation since the results for the upper leg node were obtained before and stored for the node for the upper leg piece. $R(\theta_{llf})$ is known since its argument is acquired from sensor data. Therefore the only derivative that needs to be computed is $\frac{dR(\theta_{llf})}{dt}$. When constructing node data such as rotation matrices or the derivative of rotation matrices, parent node information is used to simplify and shorten calculations (Figure 6.2).

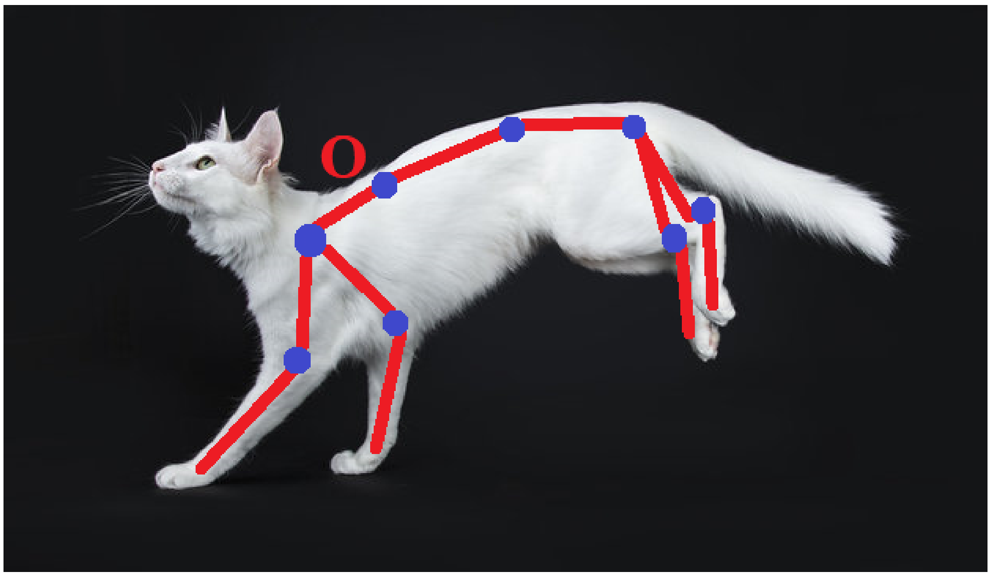

FIGURE 6.1: Tree structure example for legged robots, O denotes the origin of the tree

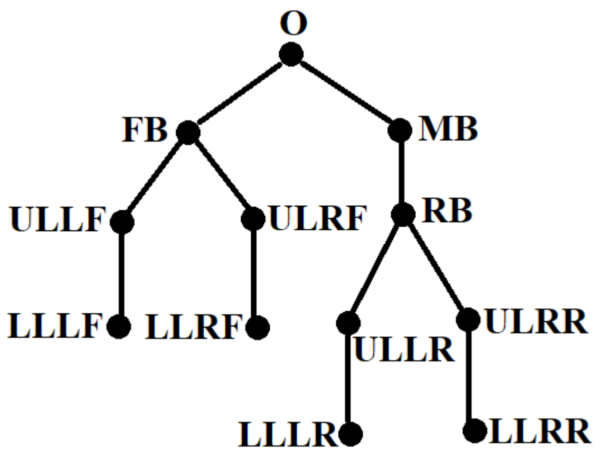

FIGURE 6.2: Tree structure example for the articulated quadruped robot and node names.

The tree structure used in the algorithm has 12 nodes (Figure 6.2). First node is origin (O), this is defined at origin of the robot coordinate axis, then front body (FB) and middle body (MB) nodes are children of the origin. Left front upper leg (ULLF) and right front upper leg (ULRF) are children of the front body. Left front lower leg (LLLF) is child of left front upper leg and right front lower leg (LLRF) is child of right front upper leg and so on. This tree structure is employed for every tree that is created.

The reference trajectory generation computation ultimately requires the calculation of angular momentum at time t and $t + \Delta t$. While all of this data could be stored in a single tree, it is convenient to store these two sets of data stored in separate trees. One general property of a tree structure utilized here is the ability to clone the tree or to create a second but empty tree isomorphic to the first one. Furthermore, again for convenience, we create six homomorphic trees for each time slice, namely the quadruped tree, the angular momentum tree, the positions tree, the derivative of positions tree, the GRM tree and the derivative of the GRM tree. All trees are built to calculate these variables for all $6 \times 2 = 12$ nodes. For time t, quadruped tree collects link lenghts, masses, joint angles, reference values, home position, rotation axes for joints and origin data. GRM tree takes quadruped tree as input and calculates all nodes GRM matrices. Derivative GRM tree takes input as quadruped tree and GRM tree to calculate derivative of GRM martices for each node. Positions tree takes GRM tree and quadruped tree as input and calculates positions of each node with respect to world coordinate frame. Derivative positions tree takes input as derivative GRM tree, positions tree and quadruped tree and calculates velocities of each node. Finally angular momentum tree takes input as position tree, derivative positions tree and quadruped tree and calculates angular momentum of each node. For time $t + \Delta t$, quadruped tree takes input as quadruped tree for time t. GRM tree takes quadruped tree for time t and quadruped tree for time $t + \Delta t$ as input and calculates all nodes GRM matrices for time $t + \Delta t$. Derivative GRM tree takes input as quadruped trees for time t and time $t + \Delta t$ and GRM tree for time $t + \Delta t$ to calculate derivative of GRM matrices for each node at time $t + \Delta t$. Positions tree takes GRM tree at time $t + \Delta t$ and quadruped tree at time $t + \Delta t$ as input and calculates positions of each node with respect to world coordinate frame at time $t + \Delta t$. Derivative positions tree takes input as derivative GRM tree at time $t + \Delta t$, positions at time $t + \Delta t$ tree and quadruped tree at time $t + \Delta t$ and calculates velocities of each node at time $t + \Delta t$. Finally angular momentum tree takes input as position tree at time $t+\Delta t$, derivative positions tree at time $t+\Delta t$ and quadruped tree at time $t + \Delta t$ and calculates angular momentum of each node at time $t + \Delta t$.

Home position of a node means starting position of a link and it calculates as;

$$p_h(node) = R_{\text{upper}}(t) \begin{bmatrix} x_h \\ y_h \\ z_h \end{bmatrix}, \tag{6.24}$$

where p_h is home position, R_{upper} is upper nodes rotation matrix and x_h, y_h and z_h are home position or origin at time t. When x_h, y_h and z_h are zero this means that the link is positioned at the ending position of previous link. Angular momentum of a link is calculated as

$$\vec{L}(t) = m_l \int_{s=0}^{1} \Big(\vec{r}_l(s,t) \times \dot{\vec{r}}_l(s,t)\Big)\, ds, \tag{6.25}$$

where integration boundaries are link starting and ending positions which are calculated by positions tree. m_l denotes the mass of the link with the index l. $\vec{r}_l(s,t)$ tracks the spatial coordinates of the robot component identified by the index l, computed by GRM tree and home position. $\dot{\vec{r}}_l(s,t)$ is calculated by derivative GRM tree and home position. $\vec{r}_l(s,t)$ and $\dot{\vec{r}}_l(s,t)$ are in GV notation as $\vec{r}_l(s,t) = GRM(l)\, p_h(l)$ and $\dot{\vec{r}}_l(s,t) = dGRM(l)\, p_h(l)$. Here, $GRM(l)$ is GRM matrix of lth node and $p_h(l)$ is home position of lth node, $dGRM(l)$ is the derivative of GRM matrix for node l. Finally, at time t and time $t + \Delta t$ there is a tree function to sum all the node values so total angular momentum is calculated. The angular momentum data at nodes is kept in GV format. Therefore,

$$\vec{L}^{total}(t) = \sum_j \vec{L}_j = [c_0(t); c_1(t); c_2(t); c_3(t)] = [c_0(t); A(t)], \tag{6.26}$$

where, j is an index that runs over nodes of the angular momentum tree, $\vec{L}_j$ is the angular momentum of jth node, $\vec{L}^{total}(t)$ is total angular momentum at time t, c_i's are column vectors of the matrix and A is a matrix created by concatenating last three column vectors.

$$\begin{aligned}\vec{L}^{total}(t+\Delta t) &= \sum_j \vec{L}_j(t+\Delta t)\\ &= [c_0(t+\Delta t); c_1(t+\Delta t); c_2(t+\Delta t); c_3(t+\Delta t)]\\ &= [c_0(t+\Delta t); A(t+\Delta t)], \qquad (6.27)\end{aligned}$$

where, j is an index that runs over nodes of the angular momentum tree at time $t+\Delta t$ and $\vec{L}_j(t+\Delta t)$ is the angular momentum of jth node at the time slice, $\vec{L}^{total}(t+\Delta t)$ is total angular momentum at time $t+\Delta t$, c_i's are column vectors of the matrix and $A(t+\Delta t)$ is a matrix created by concatenating last three column vectors. Then (6.6) becomes a linear equation with 3 unknowns as $d\theta_{N-2}$, $d\theta_{N-1}$ and $d\theta_N$.

$$A_{(3\times3)}\begin{bmatrix} d\theta_{N-2}\\ d\theta_{N-1}\\ d\theta_N\end{bmatrix} = \vec{b}, \qquad (6.28)$$

where A is a 3×3 matrix, which is calculated as $A(t+\Delta t) - A(t)$ and $\vec{b}$ is a vector computed as $\tau\Delta t - (c_0(t+\Delta t) - c_o(t))$. Therefore the desired changes in angles $(d\theta_i, i = N-2, N-1, N)$ needed for the reference trajectory generation can be computed by multiplying inverse of matrix A with $\vec{b}$.

6.2 Simulation Results

Three stages comprise the quadruped jumping gait: take-off, flight, and landing. For the take-off phase, the velocity in the x and z directions relative to the world coordinate frame is set to 5 m/s. The subsequent shift in the robot's location is seen in Figure 6.3. Due to the increased beginning velocity, the flight duration is reduced to around 1 seconds. In order to simulate an unstable takeoff, the initial body angles α, β, and γ are set to 20, 20, and 0 degrees, respectively. As previously stated, the extrapolated joint coordinates are θ_b, ϕ_b, and θ_{ulrr}, θ_{ullr}, which correspond to the front spine joint angle, rear joint angle, and rear legs' adduction/abduction joint angles, respectively. On the chosen joints, generated reference values are applied. A basic PID controller is utilized to track the references. In Figures 6.4, 6.5, 6.6 and 6.7, generated references and actual joint coordinates for these angles can be seen. The rear leg angles in Figure 6.6 and 6.7 demonstrate that the required reference is greater than the joint angle limit. The joint angles are saturated at 50 degrees in our simulation program as a measure in order to avoid link to link collisions, which constitute problems in real-world applications. The results demonstrate that the PID controller is capable of tracking reference values accurately.

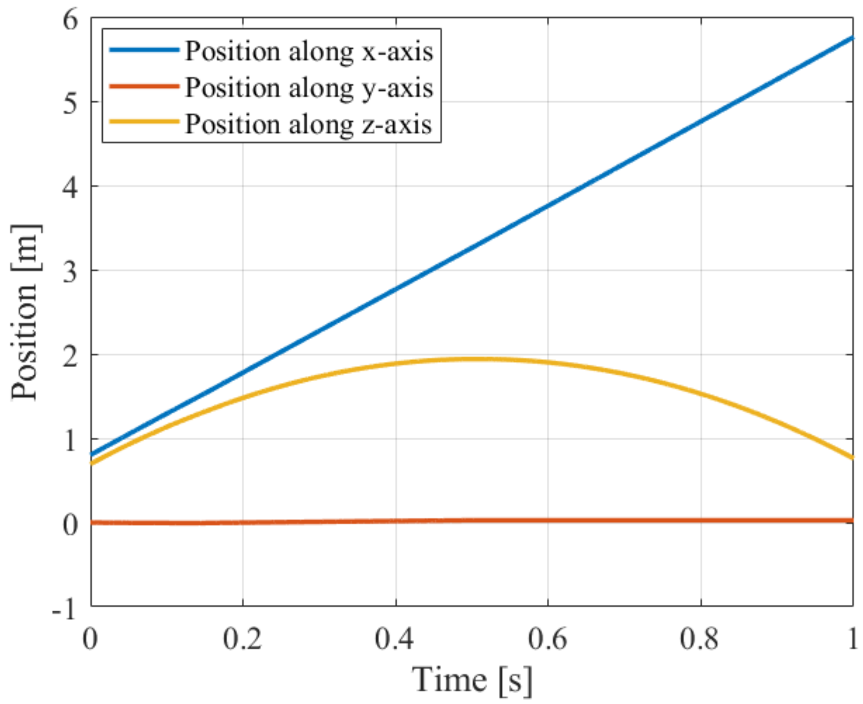

FIGURE 6.3: Quadruped robot position in x (red line), y (blue line) and z (yellow line) axes with respect to world coordinate frame.

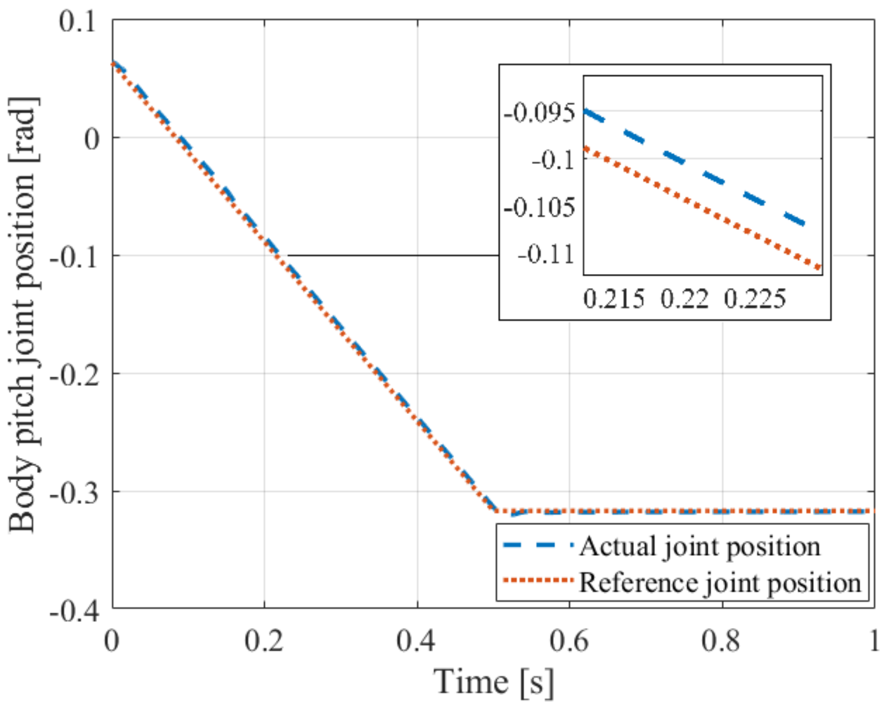

FIGURE 6.4: Front spine joint angular positions (blue line is actual angular position, red line is reference angular position).

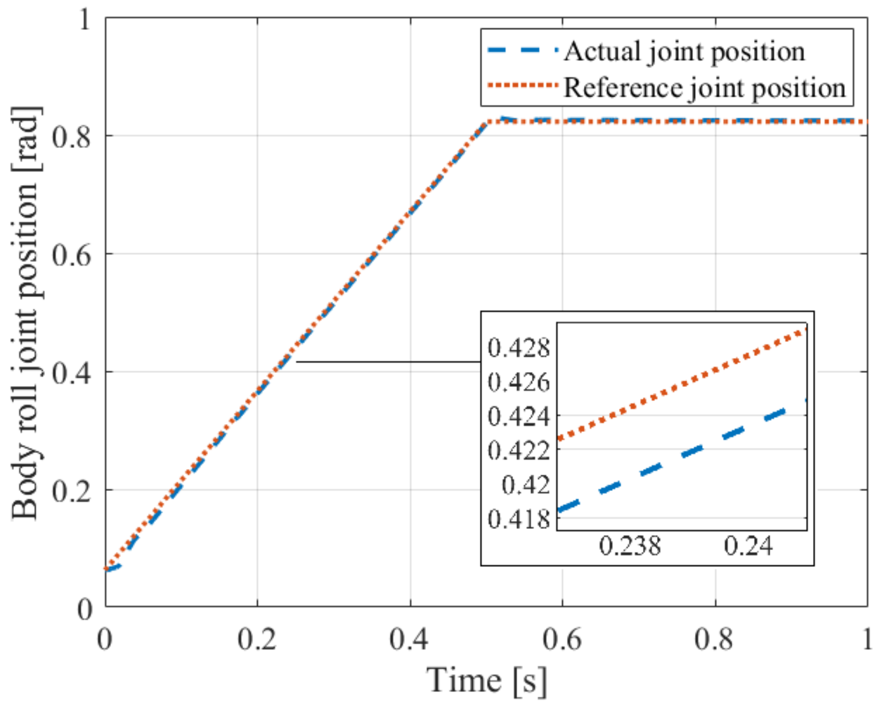

FIGURE 6.5: Rear spine joint angular positions (blue line is actual angular position, red line is reference angular position).

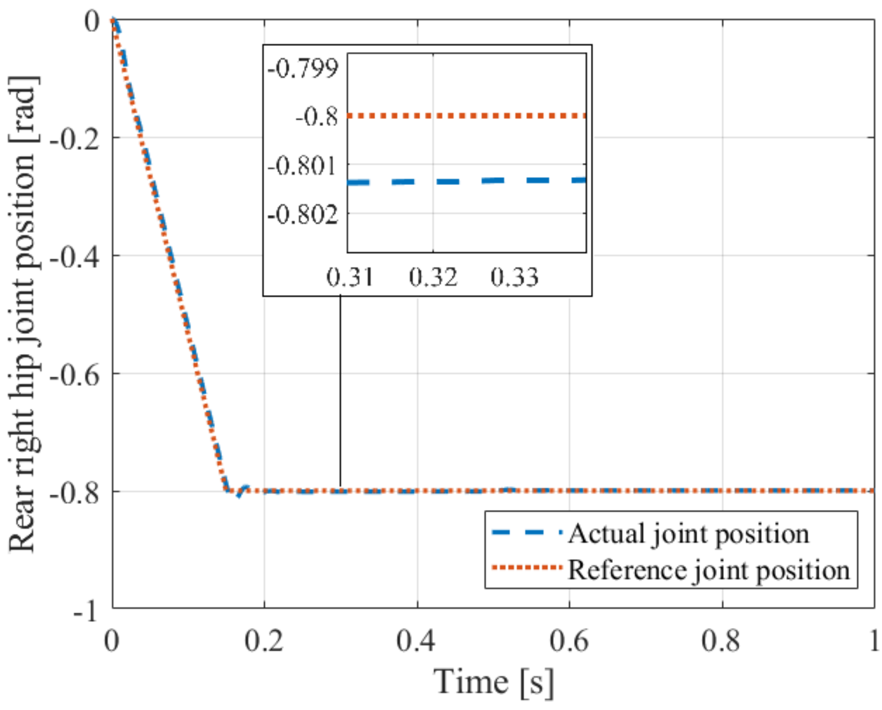

FIGURE 6.6: Right rear leg's adduction/abduction joint angular position (blue line is right leg's actual angular position, red line is right leg's reference angular position)

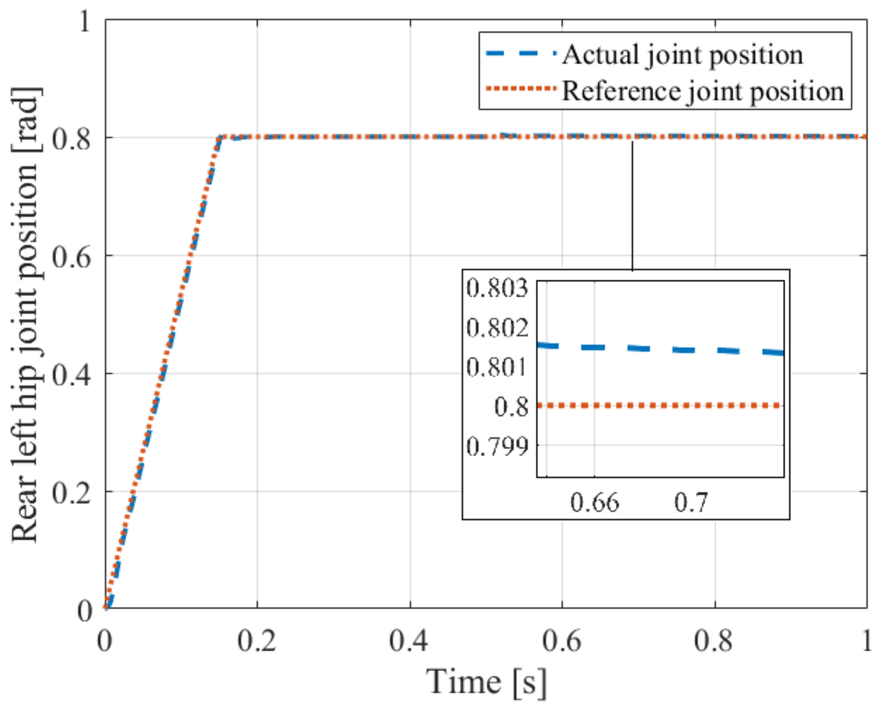

FIGURE 6.7: Left rear leg's adduction/abduction joint angular position (blue line is left leg's actual angular position, red line is left leg's reference angular position)

The roll, pitch, and yaw angles of the robot as expressed in the world coordinate frame during flight can be seen in Figure 6.8. For all angular positions, the intended final value is 0. However, since joint angles are limited, yaw angle control performance is less than roll and pitch angles. Additionally, the error value for yaw angle is approximately 7 degrees. Additionally, angle recovery for roll and pitch angles is roughly 93% in 0.5 seconds when the system is underactuated. Because the ideal landing position is nearly achieved, the robot's landing stability is improved.

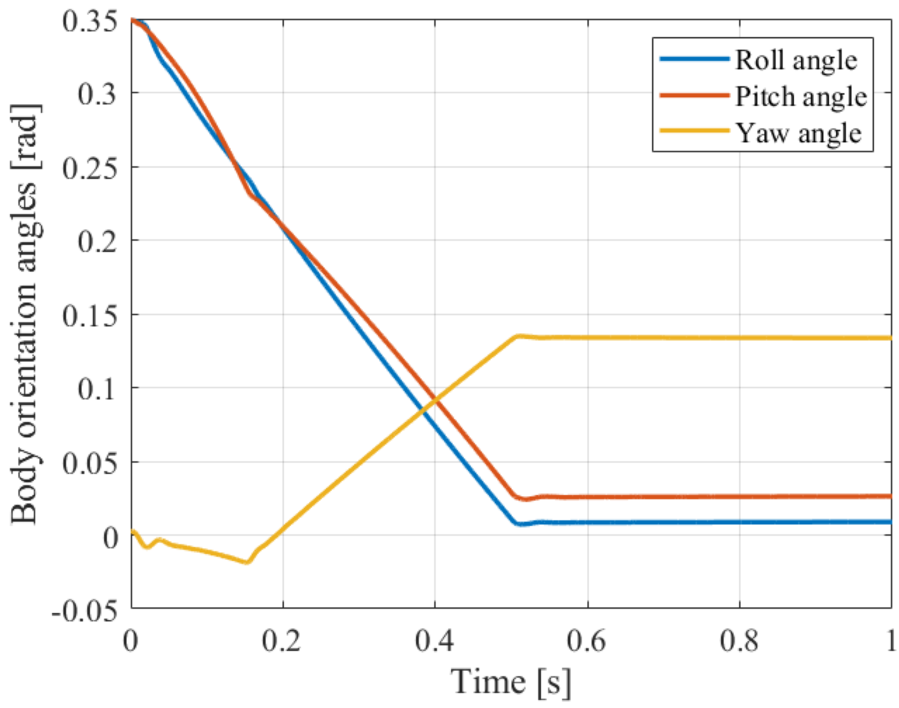

FIGURE 6.8: Quadruped robot roll (blue line), pitch (red line), and yaw (yellow line) angles with respect to world coordinate frame.

Since, this approach is more detailed than previous methods, simulations are carried out with higher the take-off velocity in the x and z directions relative to the world coordinate frame. The velocities are set to 10 m/s. As a result, the flight time increased to 2 seconds. Figure 6.9 presents resulting positions of the robot. Figures 6.10, 6.11, 6.12 and 6.13 show generated references and actual joint coordinates for θ_b, ϕ_b, and θ_{ulrr}, θ_{ullr}. Finally, The robot roll, pitch, and yaw angles can be seen in Figure 6.14. Figure 6.14 presents that this method is applicable to jumps with higher take-off velocities.

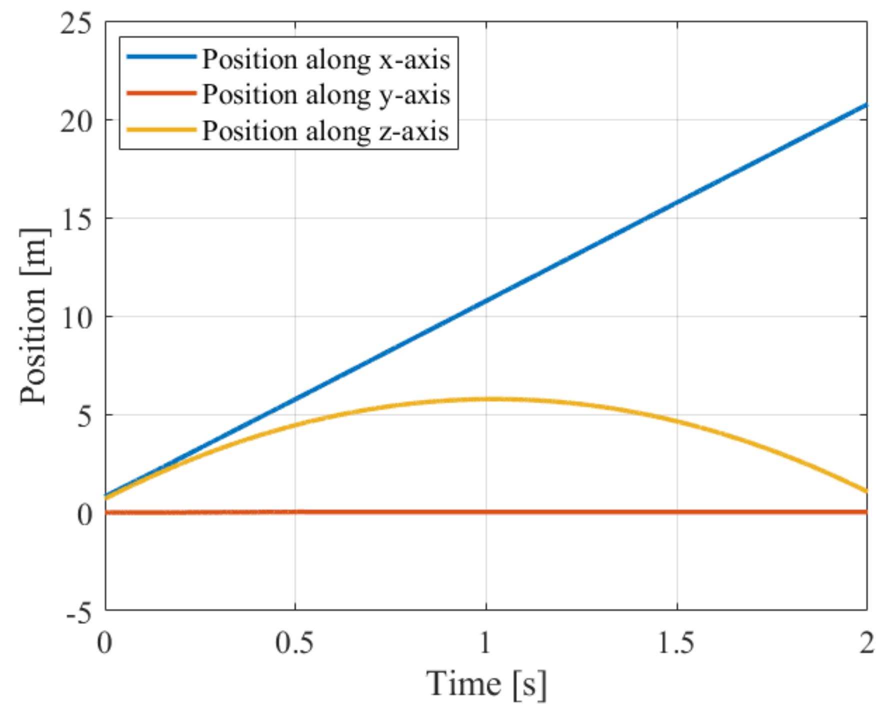

FIGURE 6.9: Quadruped robot position in x (red line), y (blue line) and z (yellow line) axes with respect to world coordinate frame.

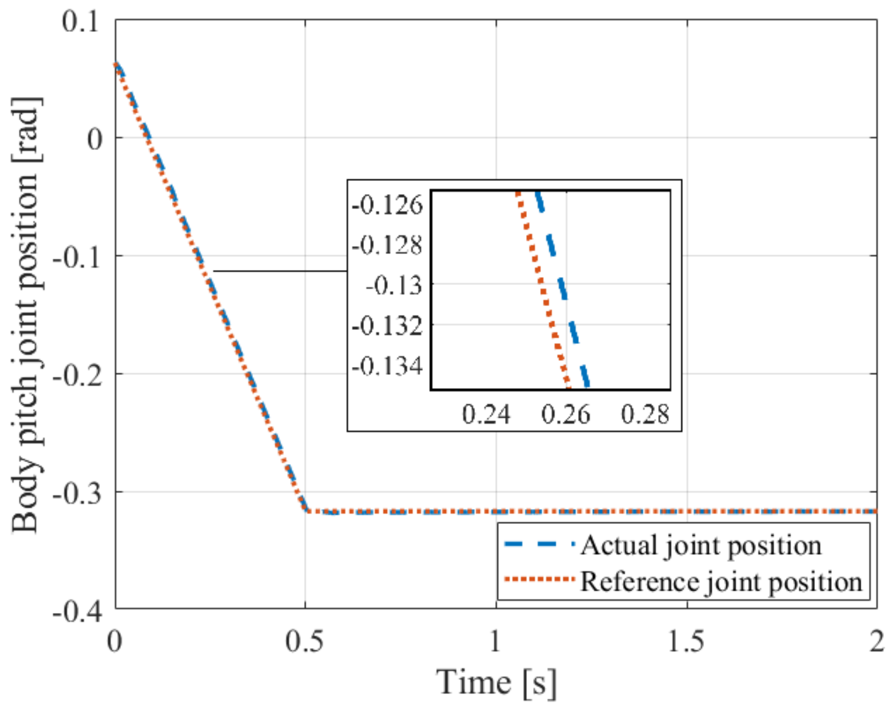

FIGURE 6.10: Front spine joint angular positions (blue line is actual angular position, red line is reference angular position).

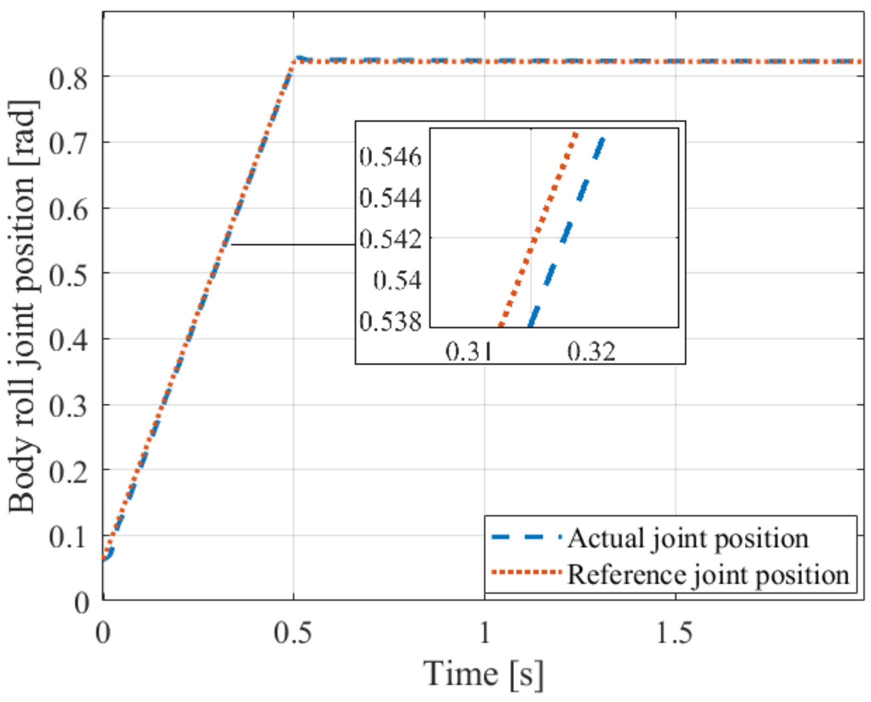

FIGURE 6.11: Rear spine joint angular positions (blue line is actual angular position, red line is reference angular position).

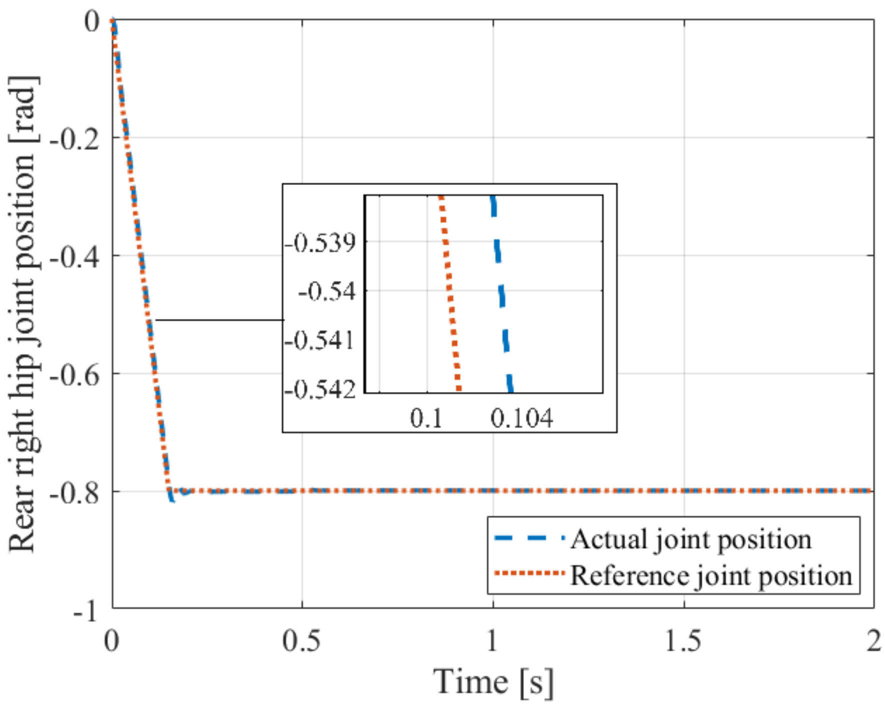

FIGURE 6.12: Right rear leg's adduction/abduction joint angular position (blue line is right leg's actual angular position, red line is right leg's reference angular position)

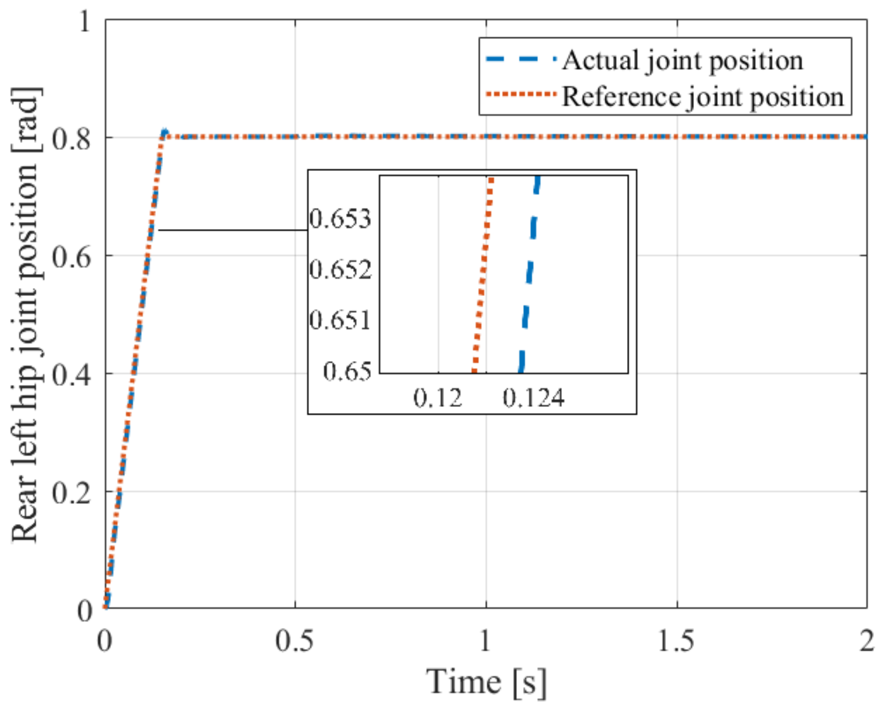

FIGURE 6.13: Left rear leg's adduction/abduction joint angular position (blue line is left leg's actual angular position, red line is left leg's reference angular position)

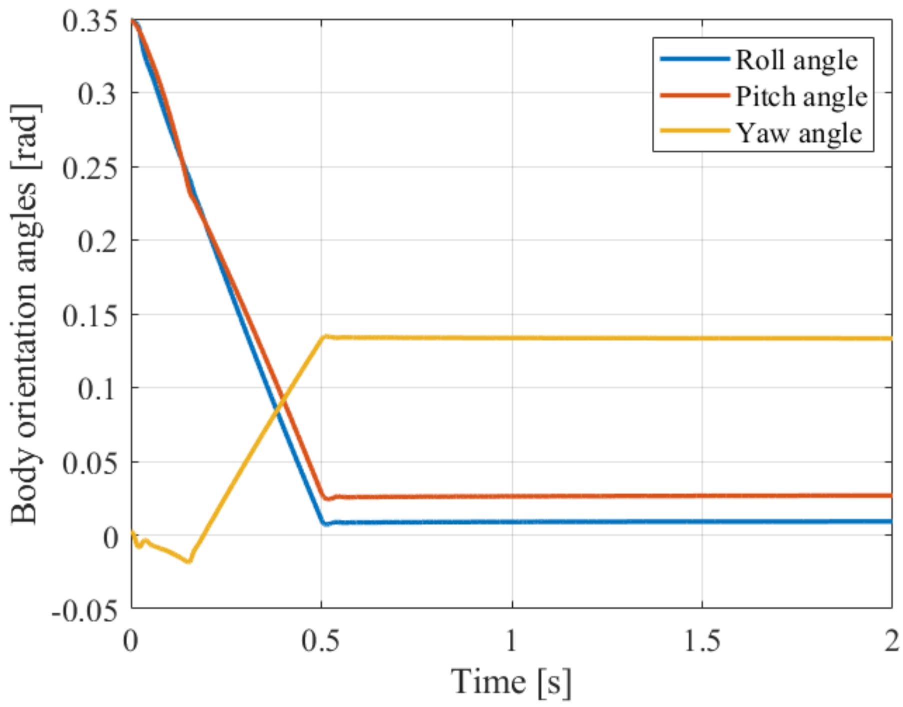

FIGURE 6.14: Quadruped robot roll (blue line), pitch (red line), and yaw (yellow line) angles with respect to world coordinate frame.

6.3 Discussion

This chapter proposes a strategy for generating reference trajectory for selected joints of free-flying robots that is based on angular momentum. Similar to previous chapters, additional tail or gyroscope does not required for this approach. Three joint angles are selected as a reference for controlling the robot's landing angular positions. These are two quadruped spine angles and a rear leg angle. However, this approach is different from Chapter 5. In Chapter 6, moment of inertia values of the links are computed beforehand and they are assumed to be constant during the flight time. In this chapter, moment of inertia is obtained for every simulation cycle. This approach eliminates some of the assumptions made in the previous chapter and decreases modeling errors.

These angular positions are computed by using linearization, GRM matrices and tree structures. Linearization is made in order to shorten computation time and made method applicable to real-time applications. GRM matrices are employed to track unknowns and to create final linear equations. Tree structure is utilized since it is very similar to legged robot structure. In addition to that this structure is applicable on any machines that has hierarchical connections between links. Tree structure cannot be employed machines that has loops since in a tree structure every child can have only one parent.

Initial yaw angle for robot body is selected as 0 degrees. Due to the design of the selected quadrupedal robot, where the rear legs have substantially less mass than the body, it is more difficult to influence the quadruped's yaw angle. The principal axis near the x direction has the smallest eigenvalue and the one near the z direction has the larger eigenvalue. Hence, the ability of manipulating angular position around z-axis is more difficult than x- and y-axes. This is an acceptable constraint, since the yaw angle has no direct effect on landing stability. Because roll and pitch angles have a considerable effect on the landing stability of quadrupedal robots, regulating these angles has a greater significance in this application area.

The program retrieved nearly 93% of the body's angular positions throughout flying time, according to simulation data. Owing to the underactuated nature of the system due to the absence of ground contact points, the reference generation approach may considerably increase the flight and landing stability of legged robots and any free-flying robot.

Additionally, similar to previous chapters, this method makes use of joints that are already present in robot mechanics. Because no additional components are required to provide flight stability, this method provides a potential for weight and cost reduction for legged jumping robots.

Chapter 7

Conclusion and Future Work

This dissertation describes novel posture and push recovery algorithms for the long-jump flying phase of an articulated-body quadrupedal robot. The methodologies are applied to the robot in a simulation environment. Our methodology is compared with the literature in this chapter, our simulation results are evaluated, and future studies are proposed.

This thesis presents a toolbox of techniques for orientation manipulation for flight phase. This is where it contrasts the literature on legged locomotion considering ground contact as the primary orientation and balance control tool. Most posture control research has been on legged robot gaits such as walking, running, bounding, and trotting. The unifying characteristic of these gaits is that they all generate contact with the ground. As a result, ground contact forces are often planned to ensure the legged robot's motion stability and posture. Another often-used technique is step-planning to prevent falling. However, this technique is not applicable when ground contact is not present. Finally, extra tail or gyro-type components are added to robot bodies in the literature to create postural stability using an angular momentum technique. While this is a bio-inspired technology, the inclusion of components increases the cost and weight of the robots. We offer methods for orientation control applicable to robots that are not in touch with the ground without extra components. These algorithms use joints that are already present in robot mechanics. Because no additional components are required to provide flight stability, the cost and weight of flying robots are reduced.

According to simulation data, the created reference generation system, based on RL, correctly restored around 94 percent of the angular body positions around roll and pitch angles throughout the flight period. The approach provides joint position reference trajectories for the quadrupedal robot's waist joints in order to achieve the off-ground robot's desired orientation.

The simplistic posture and push recovery method is based on angular momentum variables. We derive reference trajectories for a quadrupedal robot waist and rear hip joints during the flight phase of a long jump. The approach provides reference trajectories for the quadrupedal robot's waist and rear hip joints in order to achieve the required orientation of the off-ground robot. In the simulation environment, computed trajectories are applied to the robot. When used for posture control scenarios, this method recovers 94 percent roll angle, almost 100 percent pitch angle, and has a yaw angle inaccuracy of 3 degrees. When applied to the push recovery scenario, it is seen that in the absence of the orientation control system, the robot's angular velocity rises and then settles at 0.2 rad/s after a disruption. In comparison, when the orientation control method is active, the angular velocity does not exceed 0.2 rad/s but instead declines to near 0 rad/s.

The posture recovery method that computes centroidal dynamics real-time is based on angular momentum and is used to compute reference trajectories for a quadrupedal robot with waist and rear hip joints during the flight phase of a long jump. In this method, moment of inertia values of the robot links are obtained real-time in each simulation cycle. The approach provides reference trajectories for the quadrupedal robot's waist and rear hip joints in order to achieve the required orientation of the off-ground robot. In the simulation environment, computed trajectories are applied to the robot. According to simulation data, the algorithm recovered roughly 97 percent of the body's roll angle and 93 percent of its pitch angle during the flight, and the yaw angle inaccuracy is 8 degrees. Due to the system's underactuated state with the lack of ground contact points, the reference generation strategy can significantly improve the flight and landing stability of any free-flying robot.

In future studies, the learning algorithm may be improved by using neural networks; in other words, deep reinforcement learning can be used to regulate posture. Additionally, the entirety body joints may be used for posture control to reduce the amount of time required to balance the robot and boost the effectiveness of these techniques. Because the developed algorithms are computationally efficient for real-time applications, they may be deployed on a physical quadrupedal robot.

Appendix A

List of Publications

- B. Bahceci, O.K. Adak, K. Erbatur, "Push Recovery of a Quadrupedal Robot in the Flight Phase of a Long Jump", International Journal of Mechanical Engineering and Robotics Research, vol. 11(7), pp. 486-493, 2022.
- A.S. Pehlivan, D. Kraljevic, I. Triplat, B. Bahçeci, "Critical Speed Calculation of a Refurbishment of 11MW Hydro Power Plant Unit", Turkish Journal of Electrical Engineering & Computer Sciences, vol. 30(3), pp. 644-658, 2022.
- A.S. Pehlivan, B. Bahçeci, K. Erbatur, "Performance Analysis of a Pitch Angle Controller for 2MW Wind Turbine under Abrupt Wind Speed Conditions", *2021 International Conference on Electrical, Computer and Energy Technologies* (ICECET), Cape Town, South Africa, 2021, pp. 1-5.
- Ö.K. Adak, B. Bahçeci, K. Erbatur, "Operasyonel Uzayda Hibrit Kuvvet-Hareket Kontrolüne Sahip Tek Bacaklı Robot ", *Otomatik Kontrol Türk Milli Komitesi Ulusal Kongresi* (TOK 2021), Van, Turkey, 2021 (in Turkish).
- O.K. Adak, B. Bahceci, K. Erbatur, "Hybrid Force-Motion Control for One-Legged Robot in Operational Space", *IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (AIM), Delft, the Netherlands, 2021.

- A.S. Pehlivan, B. Bahçeci, K. Erbatur, "2MW Rüzgar Türbini için Kanat Açısı Kontrolörleri P, PI, PID ve Optimize Edilmiş PID'nin Performans Karşılaştırması", *Otomatik Kontrol Türk Milli Komitesi Ulusal Kongresi* (TOK 2021), Van, Turkey, 2021 (in Turkish).

- T. Akbaş, Ş.E. Eskimez, S. Özel, K.C. Fidan, B. Bahçeci, Ö.K. Adak, M.M. Gülhan, K. Erkekli, K. Erbatur, "Dört bacaklı robotlar için önizleme kontrolü ve sıfır moment noktası esaslı yürüyüş yörüngesi üretimi", *Türkiye Robotbilim Konferansı* (TORK 2016), Istanbul, Turkey, 2016 (in Turkish).

- Ö.K. Adak, M. Gülhan, B. Bahçeci, K. Erkekli, K. Erbatur, "Dört bacaklı robotlarda merkezi örüntü üreteci ve genetik algoritmalar ile referans sentezi", *Türkiye Robotbilim Konferansı* (TORK 2016), Istanbul, Turkey, 2016 (in Turkish).

- O.K. Adak, B. Bahceci, K. Erbatur, "Modeling of a Quadruped Robot with Spine Joints and Full-Dynamics Simulation Environment Construction", International Journal of Intelligent Robotics and Applications (**Under Review**)

- O.K. Adak, B. Bahceci, K. Erbatur, "Bound Gait Motion Control for a Quadruped Robot in the Operational Space Based on Contact Force Planning", Journal Paper (**In preparation**)

- O.K. Adak, B. Bahceci, K. Erbatur, "Bound Gait Control of a Quadruped Robot with an Active Spine", Journal Paper (**In preparation**)

- B. Bahceci, O.K. Adak, K. Erbatur, "Learning-Based Posture Control of a Quadrupedal Robot in the Flight Phase of a Long Jump", Journal Paper (**In preparation**)

- A.S. Pehlivan, B. Bahçeci, K. Erbatur, "Genetically Optimized Pitch Angle Controller of a Wind Tur-bine with Fuzzy Logic Design Approach", Journal Paper (**In preparation**)

Bibliography

[1] Y. Ishii, H. Yamamoto, S. Kim, and Y. Ikemoto, "Development of the experimental system that can acquire the gait data online in a quadruped robot," in *2018 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, (Nagoya, Japan), pp. 1–4, Dec. 2018.

[2] C. Kertesz and M. Turunen, "Body State Recognition for a Quadruped Mobile Robot," in *2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES)*, (Las Palmas de Gran Canaria), pp. 323–328, June 2018.

[3] S.-J. Yi, B.-T. Zhang, D. Hong, and D. D. Lee, "Online learning of a full body push recovery controller for omnidirectional walking," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*, (Bled, Slovenia), pp. 1–6, Oct. 2011.

[4] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Madrid), pp. 1–9, Oct. 2018.

[5] B. Ugurlu, I. Havoutis, C. Semini, and D. G. Caldwell, "Dynamic trot-walking with the hydraulic quadruped robot HyQ: Analytical trajectory generation and active compliance control," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Tokyo), pp. 6044–6051, Nov. 2013.

[6] B. Stephens, "Humanoid push recovery," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*, (Pittsburgh, PA, USA), pp. 589–595, Nov. 2007.

[7] B. J. Stephens and C. G. Atkeson, "Push Recovery by stepping for humanoid robots with force controlled joints," in *2010 10th IEEE-RAS International Conference on Humanoid Robots*, (Nashville, TN, USA), pp. 52–59, Dec. 2010.

[8] L.-F. Wu and T.-H. S. Li, "Fuzzy Dynamic Gait Pattern Generation for Real-Time Push Recovery Control of a Teen-Sized Humanoid Robot," *IEEE Access*, vol. 8, pp. 36441–36453, 2020.

[9] S. Dafarra, F. Romano, and F. Nori, "Torque-controlled stepping-strategy push recovery: Design and implementation on the iCub humanoid robot," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, (Cancun, Mexico), pp. 152–157, Nov. 2016.

[10] P. Ghassemi, M. T. Masouleh, and A. Kalhor, "Push recovery for NAO humanoid robot," in *2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, (Tehran, Iran), pp. 035–040, Oct. 2014.

[11] B. Ugurlu, K. Kotaka, and T. Narikiyo, "Actively-compliant locomotion control on rough terrain: Cyclic jumping and trotting experiments on a stiff-by-nature quadruped," in *2013 IEEE International Conference on Robotics and Automation*, (Karlsruhe, Germany), pp. 3313–3320, May 2013.

[12] N. Dini and V. J. Majd, "An MPC-based two-dimensional push recovery of a quadruped robot in trotting gait using its reduced virtual model," *Mechanism and Machine Theory*, vol. 146, p. 103737, Apr. 2020.

[13] R. C. Luo and C. C. Chen, "Quasi-Natural Humanoid Robot Walking Trajectory Generator Based on Five-Mass With Angular Momentum Model," *IEEE Transactions on Industrial Electronics*, vol. 65, pp. 3355–3364, Apr. 2018.

[14] P. Shamna, N. Priya, and K. S. Ahamed, "Walking stability control of biped robot based on three mass with angular momentum model using predictive PID control," in *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, (Coimbatore), pp. 584–588, Apr. 2017.

[15] B. Ugurlu and A. Kawamura, "Bipedal Trajectory Generation Based on Combining Inertial Forces and Intrinsic Angular Momentum Rate Changes: Eulerian ZMP Resolution," *IEEE Transactions on Robotics*, vol. 28, pp. 1406–1415, Dec. 2012.

[16] K. Guan, K. Yamamoto, and Y. Nakamura, "Push Recovery by Angular Momentum Control during 3D Bipedal Walking based on Virtual-mass-ellipsoid Inverted Pendulum Model," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, (Toronto, Canada), pp. 120–125, Oct. 2019.

[17] M. Farrell and H. Herr, "Angular momentum primitives for human turning: Control implications for biped robots," in *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, (Daejeon), pp. 163–167, Dec. 2008.

[18] A. Goswami and V. Kallem, "Rate of change of angular momentum and balance maintenance of biped robots," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, (New Orleans, LA, USA), pp. 3785–3790 Vol.4, 2004.

[19] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Resolved momentum control: humanoid motion planning based on the linear and angular momentum," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, vol. 2, (Las Vegas, NV, USA), pp. 1644–1650, 2003.

[20] M. Schwienbacher, T. Buschmann, S. Lohmeier, V. Favot, and H. Ulbrich, "Self-collision avoidance and angular momentum compensation for a biped

humanoid robot," in *2011 IEEE International Conference on Robotics and Automation*, (Shanghai, China), pp. 581–586, May 2011.

[21] X. Li, Z. Jiang, H. Li, Y. Mo, M. Zou, and Q. Huang, "Dynamic stability control for a bio-robot with primates-inspired active tail," in *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*, (Beijing), pp. 2035–2040, Aug. 2015.

[22] Y. Morita and K. Ohnishi, "Attitude control of hopping robot using angular momentum," in *IEEE International Conference on Industrial Technology, 2003*, (Maribor, Slovenia), pp. 173–178, 2003.

[23] J. M. Jordan, *Robots.* The MIT Press essential knowledge series, Cambridge, MA: MIT Press, 2016.

[24] R. I. of America, *RIA Worldwide Robotics Survey and Directory.* Robot Institute of America, 1982.

[25] W. G. Walter, *The living brain.* No. 153 in The Norton library, New York: Norton, 5. [print.] ed., 1963.

[26] R. A. Brooks, "New Approaches to Robotics," *Science*, vol. 253, pp. 1227–1232, Sept. 1991.

[27] M. H. Raibert, *Legged robots that balance.* The MIT Press series in artificial intelligence, Cambridge, Mass: MIT Press, 1986.

[28] R. Arkin, "The impact of cybernetics on the design of a mobile robot system: a case study," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, pp. 1245–1257, Dec. 1990.

[29] T. Owen, "Biologically Inspired Robots: Snake-Like Locomotors and Manipulators," *Robotica*, vol. 12, pp. 282–282, May 1994.

[30] R. D. Beer, R. D. Quinn, H. J. Chiel, and R. E. Ritzmann, "Biologically inspired approaches to robotics: what can we learn from insects?," *Communications of the ACM*, vol. 40, pp. 30–38, Mar. 1997.

[31] M. H. Raibert and H. B. Brown, "Experiments in Balance With a 2D One-Legged Hopping Machine," *Journal of Dynamic Systems, Measurement, and Control*, vol. 106, pp. 75–81, Mar. 1984.

[32] M. Ahmadi and M. Buehler, "Stable control of a simulated one-legged running robot with hip and leg compliance," *IEEE Transactions on Robotics and Automation*, vol. 13, pp. 96–104, Feb. 1997.

[33] P. Gregorio, M. Ahmadi, and M. Buehler, "Design, control, and energetics of an electrically actuated legged robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 27, pp. 626–634, Aug. 1997.

[34] R. Niiyama, A. Nagakubo, and Y. Kuniyoshi, "Mowgli: A Bipedal Jumping and Landing Robot with an Artificial Musculoskeletal System," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, (Rome, Italy), pp. 2546–2551, Apr. 2007. ISSN: 1050-4729.

[35] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "BigDog, the Rough-Terrain Quadruped Robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10822–10825, 2008.

[36] U. Saranli, M. Buehler, and D. E. Koditschek, "RHex: A Simple and Highly Mobile Hexapod Robot," *The International Journal of Robotics Research*, vol. 20, pp. 616–631, July 2001.

[37] J. Clark, J. Cham, S. Bailey, E. Froehlich, P. Nahata, R. Full, and M. Cutkosky, "Biomimetic design and fabrication of a hexapedal running robot," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, vol. 4, (Seoul, South Korea), pp. 3643–3649, 2001.

[38] S. Seok, A. Wang, M. Y. Chuah, D. J. Hyun, J. Lee, D. M. Otten, J. H. Lang, and S. Kim, "Design Principles for Energy-Efficient Legged Locomotion and Implementation on the MIT Cheetah Robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, pp. 1117–1129, June 2015.

[39] K. Machairas and E. Papadopoulos, "On quadruped attitude dynamics and control using reaction wheels and tails," in *2015 European Control Conference (ECC)*, (Linz, Austria), pp. 753–758, July 2015.

[40] F. Doshi, E. Brunskill, A. Shkolnik, T. Kollar, K. Rohanimanesh, R. Tedrake, and N. Roy, "Collision Detection in Legged Locomotion using Supervised Learning," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (San Diego, CA), pp. 317–322, Oct. 2007.

[41] A. Marco, D. Baumann, M. Khadiv, P. Hennig, L. Righetti, and S. Trimpe, "Robot Learning With Crash Constraints," *IEEE Robotics and Automation Letters*, vol. 6, pp. 1439–1446, Apr. 2021.

[42] A. Schperberg, K. Chen, S. Tsuei, M. Jewett, J. Hooks, S. Soatto, A. Mehta, and D. Hong, "Risk-Averse MPC via Visual-Inertial Input and Recurrent Networks for Online Collision Avoidance," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Las Vegas, NV, USA), pp. 5730–5737, Oct. 2020.

[43] J. Degrave, R. V. Cauwenbergh, F. Wyffels, T. Waegeman, and B. Schrauwen, "Terrain Classification for a Quadruped Robot," in *2013 12th International Conference on Machine Learning and Applications*, (Miami, FL, USA), pp. 185–190, Dec. 2013.

[44] C. Kertesz, "Exploring surface detection for a quadruped robot in households," in *2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, (Espinho, Portugal), pp. 152–157, May 2014.

[45] T. Sun, Z. Dai, and P. Manoonpong, "Distributed-force-feedback-based reflex with online learning for adaptive quadruped motor control," *Neural Networks*, vol. 142, pp. 410–427, Oct. 2021.

[46] Jeong-Jung Kim and Ju-Jang Lee, "Adaptation of quadruped gaits using surface classification and gait optimization," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Tokyo), pp. 716–721, Nov. 2013.

[47] X. Li, J. Li, and Y. Guo, "Foothold selection for quadruped robot based on learning from expert," in *2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*, (Hefei), pp. 223–228, Aug. 2017.

[48] J. Kolter, P. Abbeel, and A. Ng, "Hierarchical Apprenticeship Learning with Application to Quadruped Locomotion," in *Advances in Neural Information Processing Systems*, vol. 20, 2008.

[49] Q. Yao, J. Wang, D. Wang, S. Yang, H. Zhang, Y. Wang, and Z. Wu, "Hierarchical Terrain-Aware Control for Quadrupedal Locomotion by Combining Deep Reinforcement Learning and Optimal Control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Prague, Czech Republic), pp. 4546–4551, Sept. 2021.

[50] H. Zhang, J. Wang, Z. Wu, Y. Wang, and D. Wang, "Terrain-Aware Risk-Assessment-Network-Aided Deep Reinforcement Learning for Quadrupedal Locomotion in Tough Terrain," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Prague, Czech Republic), pp. 4538–4545, Sept. 2021.

[51] T. Blum, W. Jones, and K. Yoshida, "PPMC Training Algorithm: A Deep Learning Based Path Planner and Motion Controller," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, (Fukuoka, Japan), pp. 193–198, Feb. 2020.

[52] A. Paolone de Medeiros, J. Lopes dos Santos, and C. L. Nascimento Junior, "Steering a quadruped robot: Simulation and experimental results," in *2014 IEEE International Systems Conference Proceedings*, (Ottawa, ON, Canada), pp. 460–464, Mar. 2014.

[53] N. Ratliff, J. A. Bagnell, and S. S. Srinivasa, "Imitation learning for locomotion and manipulation," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*, (Pittsburgh, PA, USA), pp. 392–397, Nov. 2007.

[54] S. Chemova and M. Veloso, "An evolutionary approach to gait learning for four-legged robots," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, (Sendai, Japan), pp. 2562–2567, 2004.

[55] D. Gu and H. Hu, "Integration of Coordination Architecture and Behavior Fuzzy Learning in Quadruped Walking Robots," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 37, pp. 670–681, July 2007.

[56] D. DeFazio and S. Zhang, "Learning Quadruped Locomotion Policies with Reward Machines," *arXiv:2107.10969 [cs]*, July 2021. arXiv: 2107.10969.

[57] Jian-Nan Lin and Shin-Min Song, "Modeling gait transitions of quadrupeds and their generalization with CMAC neural networks," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 32, pp. 177–189, Aug. 2002.

[58] D. Marhefka, D. Orin, J. Schmiedeler, and K. Waldron, "Intelligent control of quadruped gallops," *IEEE/ASME Transactions on Mechatronics*, vol. 8, pp. 446–456, Dec. 2003.

[59] Y. Shao, Y. Jin, X. Liu, W. He, H. Wang, and W. Yang, "Learning Free Gait Transition for Quadruped Robots Via Phase-Guided Controller," *IEEE Robotics and Automation Letters*, vol. 7, pp. 1230–1237, Apr. 2022.

[60] 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, pp. 236–258, Feb. 2011.

[61] M. Saggar, T. D'Silva, N. Kohl, and P. Stone, "Autonomous Learning of Stable Quadruped Locomotion," in *RoboCup 2006: Robot Soccer World Cup*, vol. 4434, pp. 98–109, Berlin, Heidelberg: Springer, 2007.

[62] T. Sato, K. Watanabe, and H. Igarashi, "Acquisition of adaptive walking behaviors using machine learning with Central Pattern Generator," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, (Barcelona, Spain), pp. 1–8, July 2010.

[63] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots," *arXiv:1804.10332 [cs]*, May 2018. arXiv: 1804.10332.

[64] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Transactions on Graphics*, vol. 37, pp. 1–11, Aug. 2018.

[65] A. Agrawal, A. Jadhav, N. Pareekutty, S. Mogili, and S. V. Shah, "Terrain Adaptive Posture Correction in Quadruped for Locomotion On Unstructured Terrain," in *Proceedings of the Advances in Robotics on - AIR '17*, (New Delhi, India), pp. 1–6, 2017.

[66] S. Gay, J. Santos-Victor, and A. Ijspeert, "Learning robot gait stability using neural networks as sensory feedback function for Central Pattern Generators," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Tokyo), pp. 194–201, Nov. 2013.

[67] Z. Li, Q. Ge, W. Ye, and P. Yuan, "Dynamic Balance Optimization and Control of Quadruped Robot Systems With Flexible Joints," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, pp. 1338–1351, Oct. 2016.

[68] Y.-z. Chen, W.-Q. Hou, J. Wang, J.-W. Wang, and H.-x. Ma, "A strategy for push recovery in quadruped robot based on reinforcement learning," in *2015 34th Chinese Control Conference (CCC)*, (Hangzhou, China), pp. 3145–3151, July 2015.

[69] D. Ferigo, R. Camoriano, P. M. Viceconte, D. Calandriello, S. Traversaro, L. Rosasco, and D. Pucci, “On the Emergence of Whole-body Strategies from Humanoid Robot Push-recovery Learning,” *arXiv:2104.14534 [cs, stat]*, Apr. 2021. arXiv: 2104.14534.

[70] D. Luo, X. Han, Y. Ding, Y. Ma, Z. Liu, and X. Wu, “Learning push recovery for a bipedal humanoid robot with Dynamical Movement Primitives,” in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, (Seoul, South Korea), pp. 1013–1019, Nov. 2015.

[71] V. B. Semwal, S. A. Katiyar, R. Chakraborty, and G. Nandi, “Biologically-inspired push recovery capable bipedal locomotion modeling through hybrid automata,” *Robotics and Autonomous Systems*, vol. 70, pp. 181–190, Aug. 2015.

[72] V. B. Semwal, K. Mondal, and G. C. Nandi, “Robust and accurate feature selection for humanoid push recovery and classification: deep learning approach,” *Neural Computing and Applications*, vol. 28, pp. 565–574, Mar. 2017.

[73] H. Kim, D. Seo, and D. Kim, “Push Recovery Control for Humanoid Robot Using Reinforcement Learning,” in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, (Naples, Italy), pp. 488–492, Feb. 2019.

[74] S.-J. Yi, B.-T. Zhang, D. Hong, and D. D. Lee, “Practical bipedal walking control on uneven terrain using surface learning and push recovery,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (San Francisco, CA), pp. 3963–3968, Sept. 2011.

[75] S.-J. Yi, B.-T. Zhang, D. Hong, and D. D. Lee, “Online learning of low dimensional strategies for high-level push recovery in bipedal humanoid robots,” in *2013 IEEE International Conference on Robotics and Automation*, (Karlsruhe, Germany), pp. 1649–1655, May 2013.

[76] T. Akbas, S. E. Eskimez, S. Ozel, O. K. Adak, K. C. Fidan, and K. Erbatur, “Zero Moment Point based pace reference generation for quadruped

robots via preview control," in *2012 12th IEEE International Workshop on Advanced Motion Control (AMC)*, (Sarajevo, Bosnia and Herzegovina), pp. 1–7, Mar. 2012.

[77] F. Asadi, M. Khorram, and S. A. A. Moosavian, "CPG-based gait transition of a quadruped robot," in *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, (Tehran, Iran), pp. 210–215, Oct. 2015.

[78] M. Raibert, M. Chepponis, and H. Brown, "Running on four legs as though they were one," *IEEE Journal on Robotics and Automation*, vol. 2, no. 2, pp. 70–82, 1986.

[79] Shaoping Bai, K. Low, and T. Zielinska, "Quadruped free gait generation combined with body trajectory planning," in *Proceedings of the First Workshop on Robot Motion and Control. RoMoCo'99*, (Kiekrz, Poland), pp. 165–170, 1999.

[80] J. Chen, X. Yongjian, C. Ming, and Z. Xiaodong, "Post-capture Angular Momentum Management of Space Robot with Controllable Damping Joints," in *2019 IEEE 2nd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, (Shenyang, China), pp. 638–642, Nov. 2019.

[81] M. Khorram and S. A. A. Moosavian, "Balance recovery of a quadruped robot," in *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, (Tehran, Iran), pp. 259–264, Oct. 2015.

[82] M. Khorram and S. A. A. Moosavian, "Push recovery of a quadruped robot on challenging terrains," *Robotica*, vol. 35, pp. 1670–1689, Aug. 2017.

[83] J. Zhao, S. Schutz, and K. Berns, "Biologically motivated push recovery strategies for a 3D bipedal robot walking in complex environments," in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, (Shenzhen, China), pp. 1258–1263, Dec. 2013.

[84] A. Parashar, A. Parashar, S. Goyal, and B. Sahjalan, “Push Recovery for Humanoid Robot in Dynamic Environment and Classifying the Data Using K-Mean,” in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies - ICTCS ’16*, (Udaipur, India), pp. 1–6, 2016.

[85] Akash, S. Chandra, Abha, and G. Nandi, “Modeling a bipedal humanoid robot using inverted pendulum towards push recovery,” in *2012 International Conference on Communication, Information & Computing Technology (ICCICT)*, (Mumbai, India), pp. 1–6, Oct. 2012.

[86] T. Kamioka, H. Kaneko, M. Kuroda, C. Tanaka, S. Shirokura, M. Takeda, and T. Yoshiike, “Dynamic gait transition between walking, running and hopping for push recovery,” in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, (Birmingham), pp. 1–8, Nov. 2017.

[87] M. Shafiee, G. Romualdi, S. Dafarra, F. J. A. Chavez, and D. Pucci, “Online DCM Trajectory Generation for Push Recovery of Torque-Controlled Humanoid Robots,” in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, (Toronto, ON, Canada), pp. 671–678, Oct. 2019.

[88] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba, “Online walking pattern generation for push recovery and minimum delay to commanded change of direction and speed,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Vilamoura-Algarve, Portugal), pp. 3411–3416, Oct. 2012.

[89] A. Yasin, Q. Huang, Q. Xu, and W. Zhang, “Biped robot push detection and recovery,” in *2012 IEEE International Conference on Information and Automation*, (Shenyang, China), pp. 993–998, June 2012.

[90] C. Ott, M. A. Roa, and G. Hirzinger, “Posture and balance control for biped robots based on contact force optimization,” in *2011 11th IEEE-RAS*

International Conference on Humanoid Robots, (Bled, Slovenia), pp. 26–33, Oct. 2011.

[91] J. K. Hodgins and M. H. Raibert, "Biped Gymnastics," *The International Journal of Robotics Research*, vol. 9, pp. 115–128, Apr. 1990.

[92] V. B. Semwal, P. Chakraborty, and G. Nandi, "Less computationally intensive fuzzy logic (type-1)-based controller for humanoid push recovery," *Robotics and Autonomous Systems*, vol. 63, pp. 122–135, Jan. 2015.

[93] M. Buehler, R. Battaglia, A. Cocosco, G. Hawker, J. Sarkis, and K. Yamazaki, "SCOUT: a simple quadruped that walks, climbs, and runs," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, vol. 2, (Leuven, Belgium), pp. 1707–1712, 1998.

[94] J.-W. Chung, I.-H. Lee, B.-K. Cho, and J.-H. Oh, "Posture Stabilization Strategy for a Trotting Point-foot Quadruped Robot," *Journal of Intelligent & Robotic Systems*, vol. 72, pp. 325–341, Dec. 2013.

[95] T. Mita and T. Ikeda, "Proposal of a variable constraint control for SMS with application to a running and jumping quadruped," in *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, (Tokyo, Japan), pp. 140–145, 1999.

[96] H. Wong and D. Orin, "Dynamic control of a quadruped standing jump," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, (Atlanta, GA, USA), pp. 346–351, 1993.

[97] H. C. Wong and D. E. Orin, "Control of a quadruped standing jump over irregular terrain obstacles," *Autonomous Robots*, vol. 1, no. 2, pp. 111–129, 1995.

[98] Y. H. Lee, Y. H. Lee, H. Lee, H. Kang, J. H. Lee, J. M. Park, Y. B. Kim, H. Moon, J. C. Koo, and H. R. Choi, "Whole-Body Control and Angular Momentum Regulation using Torque Sensors for Quadrupedal Robots," *Journal of Intelligent & Robotic Systems*, vol. 102, p. 66, July 2021.

[99] W. Shang, Z. Wu, Q. Liu, L. Duan, and C. Wang, "Foot Placement Estimator for Quadruped Push Recovery," in *2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, (Suzhou, China), pp. 1530–1534, July 2019.

[100] Y. Kojio, Y. Ishiguro, K.-N.-K. Nguyen, F. Sugai, Y. Kakiuchi, K. Okada, and M. Inaba, "Unified Balance Control for Biped Robots Including Modification of Footsteps with Angular Momentum and Falling Detection Based on Capturability," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Macau, China), pp. 497–504, Nov. 2019.

[101] R. C. Luo, Jun Sheng, Chin-Cheng Chen, Peng-Hsi Chang, and Che-I Lin, "Biped robot push and recovery using flywheel model based walking perturbation counteraction," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, (Atlanta, GA), pp. 50–55, Oct. 2013.

[102] M. Shafiee-Ashtiani, A. Yousefi-Koma, M. Shariat-Panahi, and M. Khadiv, "Push recovery of a humanoid robot based on model predictive control and capture point," in *2016 4th International Conference on Robotics and Mechatronics (ICROM)*, (Tehran, Iran), pp. 433–438, Oct. 2016.

[103] R. C. Luo, W. C. Hung, and R. Chatila, "Mimicking human push-recovery strategy based on five-mass with angular momentum model," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, (Florence, Italy), pp. 716–721, Oct. 2016.

[104] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture Point: A Step toward Humanoid Push Recovery," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*, (Genova, Italy), pp. 200–207, Dec. 2006.

[105] J. Rebula, F. Canas, J. Pratt, and A. Goswami, "Learning Capture Points for humanoid push recovery," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*, (Pittsburgh, PA, USA), pp. 65–72, Nov. 2007.

[106] Yu Wei, Bao Gang, and Wang Zuwen, "Balance recovery for humanoid robot in the presence of unknown external push," in *2009 International Conference*

on Mechatronics and Automation, (Changchun, China), pp. 1928–1933, Aug. 2009.

[107] E. C. Whitman, B. J. Stephens, and C. G. Atkeson, "Torso rotation for push recovery using a simple change of variables," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, (Osaka, Japan), pp. 50–56, Nov. 2012.

[108] S.-H. Lee and A. Goswami, "Reaction Mass Pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, (Rome, Italy), pp. 4667–4672, Apr. 2007. ISSN: 1050-4729.

[109] S. M. Kasaei, N. Lau, A. Pereira, and E. Shahri, "A reliable model-based walking engine with push recovery capability," in *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, (Coimbra, Portugal), pp. 122–127, Apr. 2017.

[110] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*, (Madrid), pp. 295–302, Nov. 2014.

[111] K.-h. Ahn and Y. Oh, "Walking Control of a Humanoid Robot via Explicit and Stable CoM Manipulation with the Angular Momentum Resolution," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Beijing, China), pp. 2478–2483, Oct. 2006.

[112] C.-H. Chang, H.-P. Huang, H.-K. Hsu, and C.-A. Cheng, "Humanoid robot push-recovery strategy based on CMP criterion and angular momentum regulation," in *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, (Busan, South Korea), pp. 761–766, July 2015.

[113] S. Kajita, K. Yokoi, M. Saigo, and K. Tanie, "Balancing a humanoid robot using backdrive concerned torque control and direct angular momentum feedback," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, vol. 4, (Seoul, South Korea), pp. 3376–3382, 2001.

[114] T. Otani, K. Hashimoto, S. Miyamae, H. Ueta, M. Sakaguchi, Y. Kawakami, H. Lim, and A. Takanishi, "Angular momentum compensation in yaw direction using upper body based on human running," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, (Singapore, Singapore), pp. 4768–4775, May 2017.

[115] M. Popovic, A. Englehart, and H. Herr, "Angular momentum primitives for human walking: biomechanics and control," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, (Sendai, Japan), pp. 1685–1691, 2004.

[116] A. H. Adiwahono, C.-M. Chew, W. Huang, and Y. Zheng, "Push recovery controller for bipedal robot walking," in *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, (Singapore), pp. 162–167, July 2009.

[117] A. H. Adiwahono, Chee-Meng Chew, Weiwei Huang, and Van Huan Dau, "Humanoid robot push recovery through walking phase modification," in *2010 IEEE Conference on Robotics, Automation and Mechatronics*, (Singapore), pp. 569–574, June 2010.

[118] R. J. Griffin, A. Leonessa, and A. Asbeck, "Disturbance compensation and step optimization for push recovery," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Daejeon, South Korea), pp. 5385–5390, Oct. 2016.

[119] A. Hosseinmemar, J. Baltes, J. Anderson, M. C. Lau, C. F. Lun, and Z. Wang, "Closed-loop push recovery for inexpensive humanoid robots," *Applied Intelligence*, vol. 49, pp. 3801–3814, Nov. 2019.

[120] S.-H. Hyon, R. Osu, and Y. Otaka, "Integration of multi-level postural balancing on humanoid robots," in *2009 IEEE International Conference on Robotics and Automation*, (Kobe), pp. 1549–1556, May 2009.

[121] Y. Kojio, Y. Omori, K. Kojima, F. Sugai, Y. Kakiuchi, K. Okada, and M. Inaba, "Footstep Modification Including Step Time and Angular Momentum

Under Disturbances on Sparse Footholds," *IEEE Robotics and Automation Letters*, vol. 5, pp. 4907–4914, July 2020.

[122] J. Lack, "Integrating the effects of angular momentum and changing center of mass height in bipedal locomotion planning," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, (Seoul, South Korea), pp. 651–656, Nov. 2015.

[123] S.-H. Lee and A. Goswami, "A momentum-based balance controller for humanoid robots on non-level and non-stationary ground," *Autonomous Robots*, vol. 33, pp. 399–414, Nov. 2012.

[124] R. C. Luo and C.-W. Huang, "A push-recovery method for walking biped robot based on 3-D flywheel model," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, (Yokohama), pp. 002685–002690, Nov. 2015.

[125] R. Schuller, G. Mesesan, J. Englsberger, J. Lee, and C. Ott, "Online Centroidal Angular Momentum Reference Generation and Motion Optimization for Humanoid Push Recovery," *IEEE Robotics and Automation Letters*, vol. 6, pp. 5689–5696, July 2021.

[126] Seung-kook Yun and A. Goswami, "Momentum-based reactive stepping controller on level and non-level ground for humanoid robot push recovery," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (San Francisco, CA), pp. 3943–3950, Sept. 2011.

[127] M. Nohmi, D. Nenchev, and M. Uchiyama, "Momentum control of a tethered space robot through tether tension control," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, vol. 1, (Leuven, Belgium), pp. 920–925, 1998.

[128] X. Tang and Li Chen, "Nonholonomic Motion Planning of Space Robot System with Dual-Arms Using Genetic Algorithm," in *2007 IEEE International Symposium on Industrial Electronics*, (Vigo, Spain), pp. 2156–2160, June 2007.

[129] T. Ikeda, Taek-Kun Nam, T. Mita, and B. Anderson, "Variable constraint control of underactuated free flying robots-mechanical design and convergence," in *Proceedings of the 38th IEEE Conference on Decision and Control*, vol. 3, (Phoenix, AZ, USA), pp. 2539–2544, 1999.

[130] N. Gupta, M. Kothari, and Abhishek, "Flight dynamics and nonlinear control design for variable-pitch quadrotors," in *2016 American Control Conference (ACC)*, (Boston, MA, USA), pp. 3150–3155, July 2016.

[131] T. Mita, Taek-Kun Nam, and Sang-Ho Hyon, "Analytical time optimal control solution for a 2-link free flying acrobots," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, vol. 3, (Seoul, South Korea), pp. 2741–2746, 2001.

[132] M. J. Powell and A. D. Ames, "Mechanics-based control of underactuated 3D robotic walking: Dynamic gait generation under torque constraints," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Daejeon, South Korea), pp. 555–560, Oct. 2016.

[133] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, (Stockholm), pp. 3555–3561, May 2016.

[134] O. K. Adak, *Whole-body bound gait control of a quadruped robot equipped with an active spine joint.* PhD thesis, Sabanci University, 2021.

[135] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction.* Adaptive computation and machine learning series, Cambridge, Massachusetts: The MIT Press, second edition ed., 2018.

[136] H. Goldstein, C. P. Poole, and J. L. Safko, *Classical mechanics.* San Francisco Munich: Addison Wesley, 3. ed. ed., 2008.

[137] A. R. Abdulghany, "Generalization of parallel axis theorem for rotational inertia," *American Journal of Physics*, vol. 85, pp. 791–795, Oct. 2017.

[138] C. A. Shaffer and C. A. Shaffer, *Data structures & algorithm analysis in C++*. Dover books on mathematics, Mineola, NY: Dover Publications, 3rd ed., dover ed ed., 2011. OCLC: ocn721884632.