ANALYTIC BIPEDAL WALKING WITH FUSED ANGLES AND CORRECTIVE ACTIONS IN THE TILT PHASE SPACE

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

PHILIPP ALLGEUER

aus

Klosterneuburg, Österreich

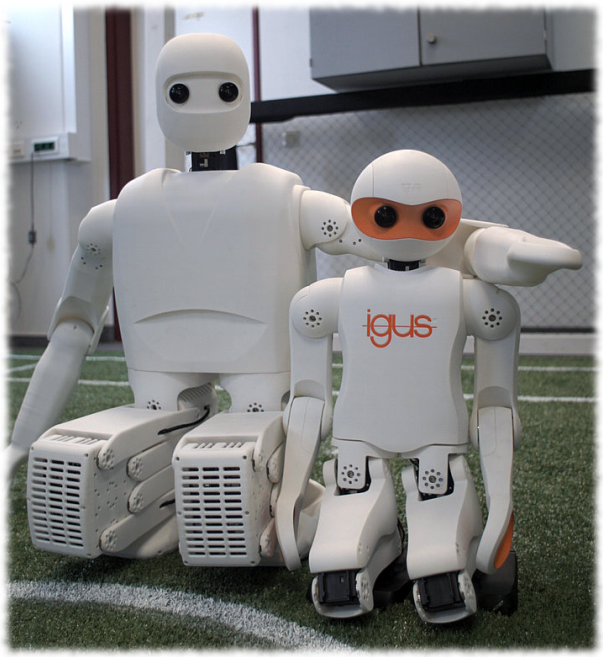
Bonn, März 2020

Angefertigt mit Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Rheinischen Friedrich-Wilhelms-Universität Bonn

1. GUTACHTER: Prof. Dr. Sven Behnke
2. GUTACHTER: Prof. Dr. Maren Bennewitz

TAG DER PROMOTION: 4. November 2020
ERSCHEINUNGSJAHR: 2020

To the one who wraps me, and to my mother Karin,
without whose tireless efforts spanning two decades
I would not even be writing this.



ABSTRACT

As humanoid robots start to move from research labs to workplace environments and homes, the topic of how they should best and most reliably locomote in the face of unknown disturbances will be a topic of increasing importance. This thesis presents algorithms and methods for the feedback-stabilised walking of bipedal humanoid robotic platforms, along with the underlying theoretical and sensorimotor frameworks required to achieve it. Bipedal walking is inherently complex and difficult to control due to the high level of nonlinearity and significant number of degrees of freedom of the concerned robots, the limited observability and controllability of the corresponding states, and—especially for low-cost robots—the combination of imperfect actuation with less-than-ideal sensing. The methods presented in this thesis deal with these issues in a multitude of ways, ranging from the development of an actuator control and feed-forward compensation scheme, to the implementation of numerous sensor calibration and processing schemes, to the inclusion of inherent filtering in almost all of the gait stabilisation feedback pipelines.

Two gaits are developed and investigated in this work, the direct fused angle feedback gait, and the tilt phase controller. Both gaits follow the design philosophy of internally leveraging a semi-stable open-loop gait generator, and extending it through stabilising feedback via the means of so-called corrective actions. The idea of using corrective actions is to modify the generation of the open-loop joint waveforms in such a way that the balance of the robot is influenced and thereby ameliorated. Examples of such corrective actions include modifications of the arm swing and leg swing trajectories, the application of dynamic positional and rotational offsets to the hips and feet, and adjustments of the commanded step size and timing.

Underpinning both feedback gaits and their corresponding gait generators (the CPG and KGG) are significant advances in the field of 3D rotation theory. These advances in particular include the development of three novel rotation representations, the tilt angles, fused angles, and tilt phase space representations. All three of these representations are founded on a new innovative way of splitting 3D rotations into their respective yaw and tilt components.

All of the algorithms presented in this thesis were implemented as part of the Humanoid Open Platform ROS Software release, and tested on a multitude of real and simulated robots, including in particular the igus Humanoid Open Platform and NimbRo-OP2. The notable walking stability that was achieved critically contributed to Team NimbRo's yearly wins at the international RoboCup competition from 2016 onwards.

ZUSAMMENFASSUNG

Humanoide Roboter verlassen zunehmend die Forschungslabore und beginnen das alltägliche Leben zu durchdringen, sowohl in Arbeitsumgebungen als auch in Privathaushalten. Angesichts dieser Entwicklung gewinnt die Frage, wie Roboter sich unter dem Einfluss unvorhersehbarer Störungen zuverlässig fortbewegen können, eine immer größere Bedeutung. Die vorliegende Arbeit stellt Methoden, Algorithmen, sowie die dazu erforderlichen theoretischen und sensomotorischen Grundlagen des stabilen Gangs auf zwei Beinen vor. Die Bipedie ist von Natur aus komplex und schwierig zu steuern, da zweibeinige Roboter ein hohes Maß an Nichtlinearität und eine hohe Anzahl an Freiheitsgraden aufweisen. Weitere Schwierigkeiten ergeben sich durch die eingeschränkte Beobachtbarkeit und Kontrollierbarkeit des Zustands, sowie die insbesondere bei kostengünstigen Robotern auftretende Kombination aus unpräziser Motorik und ungenauer Sensorik. Die vorgestellten Algorithmen lösen diese Problematik durch die Kombination verschiedener Techniken, insbesondere durch ein neuartiges Aktuator-Regelungsschema, zahlreiche Sensorkalibrierungs- und Prozessschemata, sowie die Anwendung von Filtern in den Rückkopplungspfaden.

In dieser Arbeit werden zwei Gangarten entwickelt und untersucht, der „direct fused angle feedback gait“ und der „tilt phase controller“. Beide Gangarten benutzen einen semistabilen Open-Loop-Ganggenerator, der zur Stabilisierung des Roboters mit Korrekturmaßnahmen erweitert wird. Diese Korrekturmaßnahmen modifizieren die erzeugten Gelenktrajektorien, und beeinflussen dadurch das Gleichgewicht des Roboters. Beispiele für diese Korrekturmaßnahmen stellen die gezielte Abänderung der Bewegungsbahnen der Arme und Beine, sowie die Adaptierung der Schrittgrößen und Zeitabfolgen dar.

Beide Gangarten, sowie die dazugehörigen Ganggeneratoren (CPG und KGG), bauen auf umfangreichen Neuentwicklungen auf dem Gebiet der 3D-Rotationstheorie auf, die ebenfalls in dieser Arbeit vorgestellt werden. Diese umfassen insbesondere die Entwicklung von drei neuartigen Rotationsdarstellungen, die „tilt angles“, „fused angles“ und „tilt phase space“ Darstellungen. Alle drei basieren auf einem neuen, innovativen Ansatz zur Aufteilung von Rotationen im dreidimensionalen Raum in Gier- und Neigungskomponenten.

Alle in dieser Arbeit vorgestellten Algorithmen wurden in der Humanoid Open Platform ROS Software implementiert und auf mehreren humanoiden Robotern praktisch erprobt, unter anderem auch auf der igus Humanoid Open Platform. Die hohe Stabilität des damit erzielten Gangs trug maßgeblich zu den zahlreichen Siegen des Teams NimbRo bei den jährlichen RoboCup Wettbewerben ab 2016 bei.

ACKNOWLEDGEMENTS

I would first of all like to thank Lynne Winkler for her unquestionable support in all aspects of my life, be it robots, RoboCup, climbing, Ninja Warrior competitions, or anything else. I would also like to thank Marcell Missura for inducting me into the world of bipedal walking, Hafez Farazi for the hard work we shared preparing for those many RoboCups, and Grzegorz Ficht for building the NimbRo-OP2 and NimbRo-OP2X robots, with the help of André Brandenburger, who also kept the robots in good shape for whenever I needed them. I would also like to thank Michael Schreiber for his general mechanical and mechatronic support, Max Schwarz for helping me whenever I had a particularly fiendish Linux problem or bug, and Prof. Dr. Sven Behnke for the opportunity to pursue my research interests and compete at the RoboCup competitions.

This is an abridged version of the full PhD thesis,
for the purposes of making it suitable for submission.

A more detailed version that incorporates extra
explanations and significant extra material can be found at:
https://arxiv.org/pdf/2011.10339

Refer to: Allgeuer (2020)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AHRS	Attitude and Heading Reference System	114
CAD	Computer-Aided Design	31
CCW	Counterclockwise	34
CMP	Centroidal Moment Pivot [Point]	14
CoM	Centre of Mass [Point]	4
CoP	Centre of Pressure [Point]	13
CPG	Central Pattern Generator	4
CPU	Central Processing Unit [Hardware]	132
CW	Clockwise	127
DC	Direct Current	21
DCM	Divergent Component of Motion [Point]	14
DDP	Differential Dynamic Programming	11
DoF	Degree of Freedom	26
eCMP	Enhanced Centroidal Moment Pivot [Point]	15
EKF	Extended Kalman Filter [Filter]	114
EMF	Electromotive Force	22
EW	Exponentially Weighted [Integrator]	149
GCV	Gait Command Velocity	136
GPS	Global Positioning System [Sensor]	115
IIR	Infinite Impulse Response [Filter]	197
ILC	Iterative Learning Control	25
IMU	Inertial Measurement Unit [Sensor]	5
KGG	Keypoint Gait Generator	4
LIDAR	Light Detection and Ranging [Sensor]	115
LIPM	Linear Inverted Pendulum Model	10
MEMS	Micro-Electro-Mechanical Systems [Sensor]	113
MIQCP	Mixed-Integer Quadratically Constrained Program	12
MPC	Model Predictive Control	11
PC	Personal Computer [Hardware]	25
PD	Proportional-Derivative [Controller]	17
PI	Proportional-Integral [Controller]	14
PID	Proportional-Integral-Derivative [Controller]	21
PWM	Pulse Width Modulation	22
QP	Quadratic Program	12
RBDL	Rigid Body Dynamics Library [Software]	26
RNEA	Recursive Newton Euler Algorithm	26
ROS	Robot Operating System [Middleware]	6
SISO	Single-input single-output	114
SPL	Standard Platform League	17
SQD	Spatially Quantised Dynamics	10
URDF	Unified Robot Description Format [File format]	26

1 INTRODUCTION

The task of bipedal locomotion exposes many facets of the concept of balance—most notably the many and varied methods by which balance can be preserved. While humans, even in early childhood, seem to effortlessly know how to stabilise their gait and best react to pushes from all directions while walking, the situation is very different for humanoid robotic platforms. For robots it must first be established by which approaches they can be made to keep their balance, before algorithms can be developed that allow them to reliably execute such strategies. It is a desire in the field of humanoid robotics for legged robots to eventually be able to move as fluidly and dynamically as their animal and human counterparts. Although this is something that we are beginning to see in the state of the art of quadruped robots, for biped robots this goal is still decently removed.

This thesis explores the generation of robust feedback-stabilised bipedal walking gaits, with the aim of using as diverse a set of strategies as possible for keeping balance—not just the standard adjustments of step size and timing.[1] It is explored how gaits and their underlying sensorimotor architecture can be constructed so that they work in particular for low-cost robots, where dealing with sensor noise and lacking actuator precision is paramount, and where the kinds of calculations and predictions one can make about the future states of the robot are limited. With possible additional restrictions on the available onboard computational resources, methods are investigated that rely predominantly on analytic calculation, and not on the solution of large-scale numerical optimisation problems. This allows the resulting methods to be as portable and efficient as possible in generating largely model-free walking motions for a wide range of bipedal robots.

The task of bipedal locomotion poses many difficulties for a robot, including having to deal with incomplete information, sensor noise, imperfect actuation, joint backlash, structural non-rigidities, uneven surfaces, external disturbances, and latencies in the sensorimotor loop. The control aspect of bipedal walking is also made difficult by the high dimensionality and significant nonlinearity of the system, as well as the floating-base nature of the trunk, relatively low controllability of the full dynamical system, and the only indirect observability of the positions and orientations of the trunk and limbs of the robot. Viewed at a fundamental level, all changes of state of the robot can only be effectuated via the foot-to-ground contact forces, but exactly

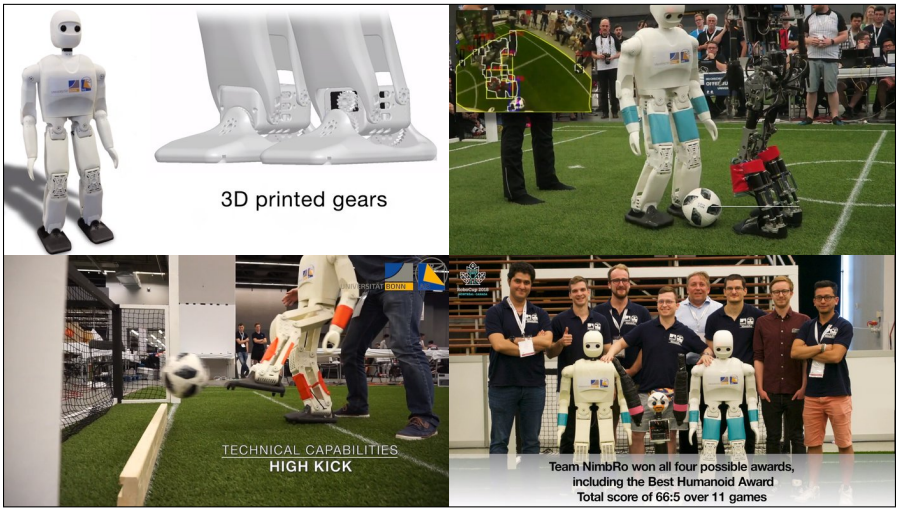
1 With sometimes the additional application of ankle torque strategies.

Video 1.1: Highlights of the performance of team NimbRo in the TeenSize soccer competition at RoboCup 2016 in Leipzig, Germany. The robot soccer team won the competition, and consisted of Dynaped and four igus Humanoid Open Platform robots.
https://youtu.be/G9llFqAwI-8
RoboCup 2016: Humanoid TeenSize Soccer Winner NimbRo

these forces are neither controllable nor measurable[2] nor particularly predictable, so the resulting task is highly challenging.

The algorithms that are developed in this thesis to address these challenges—and specifically the requirements that these algorithms then have to fulfil—are motivated by the environment of the RoboCup competition. RoboCup, or specifically the RoboCup Humanoid League, is a yearly international competition where teams from all over the world come together to compete against each other in robot soccer. The robots have to be fully autonomous, and fall into one of three size classes: KidSize, TeenSize and AdultSize. Videos 1.1 and 1.2 show summaries of two of NimbRo's many winning entries to the TeenSize and AdultSize competitions over the recent years.[3] As can be seen in the videos, the soccer fields that the robots play on consist of an uneven artificial grass surface that can be unpredictable and destabilising at times. Frequent tangles and contacts between the robots also cause the field to become a locomotion environment that is rife with a wide variety of unforeseeable disturbances.[4] The main influence of the RoboCup competition on the methods developed in this thesis however, is from the perspective of robustness under continuous autonomy. RoboCup lends the viewpoint that a method is only useful if it does the right thing every time, in every situation, without the possibility of external intervention—and not if it only does the right thing 95% of the time.

2 No reliable ankle force-torque sensors are assumed to be available in this thesis.
3 See also: https://youtu.be/RG205OwGdSg and https://youtu.be/6ldHWWHfeBc
4 See: https://youtu.be/IuqGolmLi2M



Video 1.2: Highlights of the performance of team NimbRo in the AdultSize soccer competition at RoboCup 2018 in Montréal, Canada. The robot soccer team won the competition, and consisted of the NimbRo-OP2, NimbRo-OP2X, and a highly modified Copedo robot for jumping.
https://youtu.be/tPktQyFrMuw
RoboCup 2018 Humanoid AdultSize Soccer Winner NimbRo

There are two main paradigms for the implementation of bipedal robotic gaits. A common approach in the state of the art is to use a dynamics model of some kind to capture and predict the physical response of the robot, and calculate or optimise a trajectory to satisfy the required motion and balance criteria. This motion trajectory is then executed on the robot, often with a controller to reject deviations and enforce tracking, and/or under regular recomputation to adapt for differences in the real response of the robot. For low-cost and imprecise robots however, where good quality tracking and execution of a trajectory is not given, using such optimised trajectories generated directly from simplified or even whole-body dynamics models is often fraught with difficulty. Significant nonlinearities, such as joint backlash, sensor and actuator delays, irregular properties of the contact surface, and unmeasurable external disturbances, are difficult to incorporate into models. This greatly limits the predictive power of such models, and subsequently the applicability of such methods to such robots.

Small, cheap and simple robots can often easily be made to walk despite these nonlinearities using hand-crafted gaits. These gaits are usually not very flexible, and not particularly resistant to disturbances, but nevertheless find a way to make use of the natural dynamics of the robot to produce a functional semi-stable gait. This is the foundation of the second paradigm for the implementation of bipedal gaits—letting the robot find its own natural rhythm with an open-loop gait generator, and extending this generator with feedback controllers that seek to return the robot to that rhythm when there are noticeable

deviations from it. The gaits developed in this thesis, namely the direct fused angle feedback gait and the tilt phase controller, both follow this paradigm, and extend their respective open-loop gaits, the Central Pattern Generator (CPG) and the Keypoint Gait Generator (KGG), with so-called corrective actions. These are targeted dynamic modifications of the generated open-loop waveforms that are designed to influence the balance of the robot in a particular way. Examples of corrective actions include tilt adjustments of the arms, orientation adjustments of the feet, and translational shifts of the robot's Centre of Mass (CoM). It should be noted that step size and timing adjustments are also considered to be corrective actions, but step size feedback, for example, is only used as a last resort for keeping balance, as amongst other things it leads to a direct non-compliance with any existing footstep plans.

Underpinning the two new gaits are many developments in the areas of sensor processing, state estimation, actuation and rotation theory. These developments are all presented in detail in this thesis, and in particular include methods for actuator control, sensor calibration, sensor processing and attitude estimation. In terms of rotation theory, significant advancements to the state of the art are made through the introduction of a new and better concept of 'yaw', namely fused yaw. Three new rotation representations that leverage this concept of yaw—the tilt angles, fused angles and tilt phase space representations—are also introduced and used recurrently throughout the remainder of the thesis.

All of the developments and contributions made throughout this thesis have been released fully open source online (Team NimbRo, 2018). They have been tested not only in simulation, but also predominantly on real hardware. This is important, as real hardware conditions are almost always more strenuous than simulated ones, and arguably, performance on a real robot is all that matters. Testing has been carried out on a wide variety of real robots, including the igus Humanoid Open Platform and the NimbRo-OP2.

1.1 KEY CONTRIBUTIONS

The following key contributions are made by this thesis:

- The novel concept of fused yaw is introduced as a way of partitioning 3D rotations into their respective yaw and tilt components in a geometrically and mathematically meaningful way.
- Based on the concept of fused yaw, three new rotation representations are introduced and investigated in detail: the tilt angles, fused angles and tilt phase space representations. The latter two in particular offer definitions of 'pitch' and 'roll' that far surpass the definitions that are used in the context of Euler angles.

- A gait using corrective actions driven by fused angle feedback mechanisms is built around an open-loop central pattern-driven gait generator, and is tested on many different robotic platforms.
- A new self-stable omnidirectional gait generator called the KGG is developed. It strikes a balance between the security and simplicity of hand-crafted gaits, and the advantages of analytically computed and optimised gaits. More than just make the robot walk, the KGG directly embeds a myriad of corrective actions that can be commanded and activated by higher level controllers to systematically allow preservation of balance during walking.
- A controller for the corrective actions of the KGG is formulated based on the tilt phase space. The diversity of the actuated corrective actions arguably covers the majority of humanlike strategies by which biped robots can balance during walking.

In addition to these key contributions, the following contributions are also made by this thesis:

- The novel *nonlinear tripendulum model* is developed for the purposes of dynamic step size adjustment.
- A 3D attitude estimator is realised that automatically uses the concept of fused yaw in the absence of magnetometer data for heading estimation purposes.
- Methods for calibration of the Inertial Measurement Unit (IMU) and subsequent processing of sensor data are presented, including in particular a method for online gyroscope bias autocalibration.
- An actuator control scheme is realised that makes use of a servo motor model and feed-forward torque estimation scheme to improve the compliant tracking of dynamic joint trajectories.
- The notion of tilt vector addition is introduced in the context of 3D rotation theory.
- All developed algorithms and methods are released fully open source online, in the C++ and sometimes also Matlab languages.

The full version[5] of this thesis (Allgeuer, 2020) makes the following extended contributions:

- Two useful non-standard kinematic pose spaces are presented, and used to formulate a cost-based solution to the humanoid leg inverse kinematics problem. A generalised arm inverse kinematics method is also developed that can place any functional point of the arm on any desired ray through the shoulder.

5 The current document is an abridged version of the full thesis, and was required in order to meet submission guidelines concerning the length of the thesis.

- A humanoid kinematic model is used in conjunction with the developed attitude estimator to formulate CoM state and stepping motion estimation processes.
- The capture step gait from Missura (2015) is refined in terms of its open-loop gait, state estimation and implementation. A rigorous data-based tuning approach is also documented.
- Methods for sensor and robot calibration are presented, in particular for the purposes of magnetometer calibration and data processing.
- A method for the online autocalibration of the magnetometer hard offset is developed.
- The novel notion of referenced rotations is introduced.
- A deeper investigation of 3D rotations is performed, including a walkthrough of 3D rotation theory, and many further properties of the new yaw-tilt rotation representations.
- A number of innovative non-standard mathematical filters and functions are detailed and used throughout the various presented gaits and calibration algorithms.

Although not a direct contribution of this thesis, further contributions to the released Humanoid Open Platform ROS Software (and associated robot platforms) were made by the author in the course of completing this thesis. These more indirect contributions include work in the areas of soccer behaviours, demonstration routines, state machines (Allgeuer and Behnke, 2013), servo communication routines, microcontroller firmwares (CM730/CM740), robot hardware design and construction. More general contributions to the ROS software framework included contributions in the areas of robot control, Gazebo simulation, real-time logging, dynamic configuration, visualisation tools, hardware interfaces, network interfaces, and general motion modules.

1.2 PUBLICATIONS

Parts of this thesis have been previously published in conference proceedings. The academic publications that contributed to, and form part of this thesis, are listed as follows.

Allgeuer and Behnke (2014) Presents an attitude estimator, based on 3D nonlinear complementary filtering, that fuses 3-axis gyroscope, accelerometer and optionally magnetometer data into a robust quaternion estimate of orientation.

P. Allgeuer and S. Behnke (2014). "Robust Sensor Fusion for Robot Attitude Estimation". In: *International Conference on Humanoid Robots (Humanoids)*. Madrid, Spain.

Allgeuer and Behnke (2015) Introduces and thoroughly investigates the properties of the fused angles and tilt angles rotation parameterisations, motivated by the task of representing the orientation of a balancing body.

P. Allgeuer and S. Behnke (2015). "Fused Angles: A Representation of Body Orientation for Balance". In: *International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany.

Allgeuer and Behnke (2016) Presents a complete omnidirectional closed-loop gait based on the direct feedback of orientation deviation estimates expressed in terms of fused angles. The feedback controls the step timing, in addition to the virtual walking slope and five further types of corrective actions.

P. Allgeuer and S. Behnke (2016). "Omnidirectional Bipedal Walking with Direct Fused Angle Feedback Mechanisms". In: *International Conference on Humanoid Robots (Humanoids)*. Cancún, Mexico.

Video 9.1: https://youtu.be/xnzJi2hTfAo (page 153)

Allgeuer and Behnke (2018a) Presents a multifaceted feedback controller, based on the tilt phase space, that uses step timing and eight other corrective actions to stabilise a core keypoint-based gait. The feedback controller is free of any physical model of the robot, very computationally inexpensive, and requires only a single 6-axis IMU.

P. Allgeuer and S. Behnke (2018a). "Bipedal Walking with Corrective Actions in the Tilt Phase Space". In: *International Conference on Humanoid Robots (Humanoids)*. Beijing, China.

Video 11.2: https://youtu.be/spFqqktZ1s4 (page 206)
Video 11.6: https://youtu.be/OSvHgVIYquc (page 215)
Video 11.5: https://youtu.be/yY0kpUZpjO4 (page 214)

Allgeuer and Behnke (2018b) Investigates and delineates in great detail the advantages that fused angles have over Euler angles for representing orientations in balance-related scenarios. Points of comparison include the locations of the singularities, the associated parameter sensitivities, the level of mutual independence of the parameters, and the axisymmetry of the parameters.

P. Allgeuer and S. Behnke (2018b). "Fused Angles and the Deficiencies of Euler Angles". In: *International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain.

Video: https://youtu.be/GVEdK0BuzG4

Allgeuer and Behnke (2018c) Introduces the tilt phase space rotation parameterisation, and extensively explores the properties of this new representation. Previously unexplored properties of the general notion of tilt rotations are also presented.

> P. Allgeuer and S. Behnke (2018c). "Tilt Rotations and the Tilt Phase Space". In: *International Conference on Humanoid Robots (Humanoids)*. Beijing, China.

1.3 OUTLINE

The remainder of this thesis is organised as follows:

Chapter 2: Presents related work for the field of bipedal locomotion.

Chapter 3: Details the actuator control scheme that is used to compensate actuator commands.

Chapter 4: Details the IMU calibration routines, including the online gyroscope bias autocalibration scheme.

Chapters 5 and 6: Establish the tilt angles, fused angles and tilt phase space rotation representations, and critically compare them to Euler angles.

Chapter 7: Details the attitude estimator used for trunk orientation estimation.

Chapter 8: Details the Central Pattern Generator (CPG), and how it is used to make robots walk open-loop.

Chapter 9: Presents the direct fused angle feedback controller, which is a balance controller that adds closed-loop stabilising mechanisms to the CPG.

Chapter 10: Introduces the Keypoint Gait Generator (KGG), which is an analytically computed open-loop gait generator that directly incorporates the option for a wide array of corrective actions in its trajectory generation scheme.

Chapter 11: Details the tilt phase controller, which is a controller that activates the corrective actions of the KGG in order to maintain balance during walking.

Chapter 12: Makes closing remarks about the achievements and contributions of the thesis, including recommendations for future work.

2 RELATED WORK

In this chapter, some of the most prominent and relevant examples of the state of the art in balanced bipedal walking are presented, to establish a background of the methods that exist in literature for such applications.

2.1 ZMP-BASED GAIT GENERATION

Many modern bipedal gaits are, in a wide variety of ways, based on the concept of the Zero Moment Point (ZMP). The ZMP of a bipedal robot is the point within the support polygon of the robot, i.e. within the convex hull of the ground contact patches, about which the net resultant ground reaction wrench has zero moment parallel to the plane of contact (Vukobratović and Borovac, 2004). If such a point does not exist because it occurs outside the support polygon, the point is instead referred to as the 'fictitious' ZMP, but often for simplicity this technical distinction is not made. The ZMP is of interest to situations of walking and balancing as it can be used to formulate a (conservative) stability criterion. If during a walking motion the actual physical ZMP remains within the support polygon at all times, the motion is guaranteed to be stable and the robot will not fall.

Early methods of ZMP-based gait stabilisation and balance control, such as for example the one used for the ASIMO robot (Hirai et al., 1998), used ZMP feedback control loops that effectively monitored the difference between the measured and desired ZMP points, and applied foot position, posture and body trajectory adjustments to counteract any unbalanced tipping moments. The desired ZMP in this case was calculated from the ideal walking pattern using a detailed physical model of the robot. Later methods of ZMP-based gait generation constructed reference ZMP trajectories based on desired walking motions, and used these to compute reference trajectories for the Centre of Mass (CoM). These trajectories were then tracked as closely as possible, generally with the use of inverse kinematics and an appropriate control law.

Given a piecewise polynomial ZMP trajectory, Harada et al. (2004) analytically derived a formulation for the reference CoM trajectory. This was later extended by Morisawa et al. (2007) to allow modifications of foot placement, while still maintaining the analytic solution approach. Harada et al. (2006) presented a method for simultaneously and analytically planning CoM and ZMP trajectories in real-time by connecting newly calculated trajectories to previous ones at the begin-

ning of double support. Morisawa et al. (2009) extended this approach to allow footstep locations to be treated as free variables, allowing reactive steps for push recovery to be realised. This was integrated with a state feedback controller in Morisawa et al. (2010) so that disturbances could be rejected either with step adaptations or disturbance suppression methods depending on their severity.

Yi et al. (2011) presented a gait for the DARwIn-OP robot that uses a step controller that plans a ZMP trajectory two steps into the future, and generates corresponding CoM trajectories in closed form directly from the Linear Inverted Pendulum Model (LIPM) equations. Jerk minimisation is not considered, and the jerk of the CoM is in fact discontinuous in the middle of each support transition, but this does not seem to have a significant effect on the stability of the gait. A push recovery controller is integrated with the walking controller, and allows ankle, hip and stepping strategies to be applied when disturbances are detected. The activations of the individual push recovery strategies are learnt over many real-world trials within a reinforcement learning setting.

2.1.1 Preview Control

A basis for many works is ZMP tracking with preview control, first introduced in the context of bipedal walking by Kajita et al. (2003). A series of footsteps are planned and used to define a reference ZMP trajectory with the use of suitable heuristics. This trajectory is often locally fixed once it has been computed, but often also recomputed online in response to tracking errors and/or disturbances (Nishiwaki and Kagami, 2010; Tedrake et al., 2015). Based on the Linear Inverted Pendulum Model described in Kajita et al. (2001), an optimal preview controller is constructed that utilises state feedback, integral tracking error feedback, and preview action based on the ZMP reference for a given future time window (Kajita et al., 2003). This controller acts to minimise ZMP tracking error under consideration of the planned future steps, and allows the robot to walk in relatively controlled and non-dynamic situations.

Many variations and follow-up works of the original ZMP preview control idea have been published over the years. For instance, Kim et al. (2019) added a control performance model that approximates the tracking performance of the CoM (including the effects of the implemented low-level tracking controller) using a mass-spring-damper between the desired CoM and real CoM. This slightly relieves the dependence on high performance tracking ability of the robot, and increases the applicability of preview control to compliant robots. Kajita et al. (2018) also investigated the use of Spatially Quantised Dynamics (SQD) in combination with preview control for stretched-knee bipedal walking. While the lateral motions of the CoM are handled by a preview control

approach, the sagittal walking trajectories are first designed kinematically, before being quantised spatially based on the hip joint position, and subsequently refined using Differential Dynamic Programming (DDP) to respect the nonlinear spatially quantised LIPM equations. During execution of the thereby attained walking motions, the SQD LIPM equations are once again used to return deviating measured states to the desired trajectory. The real benefit of using the SQD formulation is admittedly unclear to the authors of the paper themselves (Kajita et al., 2018). They also identify a lacking theoretical background for their use of local optimisation for feedback control, and need to permanently manually enforce non-zero sagittal reference velocities in order to avoid division-by-zero singularities arising from their spatial quantisation.

Variable step timing is rarely addressed by ZMP-based gait generation methods, especially in terms of online replanned reactions to unforeseen disturbances. Kryczka et al. (2015) have presented a ZMP-based method for incorporating some level of timing feedback into the trajectory optimisation process (mainly only lateral pushes towards the stance foot). ZMP preview control as per Kryczka et al. (2013) is used to generate the nominal gait pattern using the LIPM. In addition to this, CoM and ZMP feedback controllers are used to overcome the compliance of the robot. If during walking the error in the CoM position exceeds a certain threshold, the gait trajectory is replanned from the current state using nonlinear optimisation of the following two step timings and the following sagittal and lateral step positions. The aim of the replanned gait trajectories is to return the robot to the nominal CoM position and velocity that it should have at the beginning of a subsequent step. The replanned gait patterns are executed using a preview control scheme, and are spliced into the nominal gait patterns using interpolation over a small time window.

When a disturbance to a stationary robot is detected, Urata et al. (2011) optimise two future footstep locations and one future footstep time, selecting them from a number of computed ZMP/CoM trajectory pairs for which the CoM trajectory does not diverge. The CoM trajectories are calculated explicitly using singular LQ preview regulation, based on the corresponding heuristically generated ZMP reference. The executed motion plan is fixed to always consist of exactly three steps, but this is sufficient to produce very convincing disturbance rejection, albeit on specially designed hardware that is able to execute the planned motions with extreme fidelity.

2.1.2 Model Predictive Control

Given a method for calculating a reference ZMP trajectory, an alternative approach to optimising a CoM trajectory to match this ZMP trajectory is given by Model Predictive Control (MPC). A landmark paper

in this direction is Wieber (2006), in which a Linear Model Predictive Control scheme is developed that solves a continuous sequence of finite-horizon optimal control problems, and in each cycle only executes the first time step of the resulting calculated CoM trajectory (with the help of a tracking controller). While not the first to use MPC in the setting of bipedal locomotion (Azevedo et al., 2002), the innovation was to use simple LIPM dynamics and ZMP reference trajectories (generated as for ZMP preview control) for the purpose of optimisation. This allows the crux Quadratic Program (QP) to be solved analytically using matrix manipulations instead of numerical techniques, and thereby allows the scheme to work in real-time. In each time step, the optimal control problem tries to minimise CoM jerk and deviations from the ZMP reference trajectory, and as a form of feedback uses the current measured robot configuration as the initial state. While this already allows certain robustness to disturbances, Wieber (2006) also investigated reduction of the QP to minimising just the CoM jerk, but under explicit consideration of ZMP location constraints. This formulation of the QP no longer has an analytic solution, but allowed for more aggressive ZMP/CoM trajectory pairs, leading to greater disturbance rejection abilities.

To overcome the restriction of having fixed footstep locations, Diedam et al. (2008) extended the work of Wieber (2006) to include the translational locations of the footsteps in the optimisation task. Nominal footstep locations are generated as before, but instead of being used directly, these are used to penalise (via the cost function) the generation of steps that deviate from them. Additional inequality constraints are added to the MPC to ensure that only feasible steps are produced. Stephens and Atkeson (2010) also included footstep locations in their linear MPC scheme, which was targeted at push recovery for a torque-controlled robot. Their method however was constrained to push recovery from standing to standing in a fixed number of steps (generally one) with fixed timing.

Allowing non-fixed footstep yaws and heights of the CoM during walking introduces significant nonlinearities to the Linear Inverted Pendulum Model. Griffin and Leonessa (2019) proposed an MPC scheme based on the time-varying Divergent Component of Motion (DCM) dynamics (Hopkins et al., 2014) that allows both height (predefined CoM height trajectories) and foot orientation changes during walking. Smooth DCM trajectories with fixed step times are generated using DCM acceleration as the control input, and executed using reverse-time numerical integration and feedback control. Piecewise linear approximations of the constraints, in particular in relation to the foot orientations, leads to a Mixed-Integer Quadratically Constrained Program (MIQCP) that cannot quite be solved fast enough to be real-time.

Further examples of works that consider free foot placements and time-variant CoM heights include Kuindersma et al. (2014) and Brasseur et al. (2015), both of which conservatively bound nonlinearities using linear functions in order to allow more efficient solution methods. Building on the latter of these two works, Pajon and Wieber (2019) introduced a linear MPC scheme that incorporates a 3D capturability constraint for greater passive safety of the generated walking motions. All generated future trajectories of the robot CoM and Centre of Pressure (CoP) are constrained to end at their preview horizon with a dynamically balanced convergent stopping motion, i.e. an exponential continuation of the previewed CoM trajectory that in limit $t \to \infty$ settles to a stable position that involves the CoM stopping somewhere over the final support foot.

The main issue with bipedal gaits based on the Model Predictive Control method is their high computational intensity. A mix of linearisations, approximations and design choices must generally be made to reduce the complexity of the problem to a tractable level, but even so, a large amount of computation is required in every cycle, only to be thrown away after executing merely the very first time step of the calculated trajectory. Some other walking methods, for example as described by Feng et al. (2015), replan their calculated CoM trajectories only once per single support phase, but this to some extent sacrifices immediate responsiveness to disturbances.

2.2 CAPTURE POINT AND DIVERGENT COMPONENT OF MOTION

The concept of capture points and capture regions (the set of all capture points) was first introduced by Pratt et al. (2006). Most generally, a capture point of a robot in a certain state is a point on the ground that if stepped on brings the robot to a complete stable stop. Although complicated to calculate for general bipedal robots, for simple models like the LIPM there is a unique capture point that can easily be calculated based on the CoM position and velocity. Push recovery of a real robot based on the capture point was first demonstrated by Pratt et al. (2009). Later, Englsberger et al. (2011) presented a landmark walking gait based on stabilisation of the divergent component of the LIPM capture point dynamics.

Koolen et al. (2012) presented a treatise on the N-step capturability of walking robots as it relates to three simple gait models based on the 3D LIPM, and refined the definition of the capture point to include the instantaneous capture point. This is essentially the same as the original definition of the capture point, only without the restrictions of dynamic reachability, i.e. without consideration of step length and time limitations. Pratt et al. (2012) demonstrated a gait based on the instantaneous capture point. The current capture region is first

estimated based on the current state and capturability analysis of the 3D LIPM with finite-sized feet, before a desired footstep is calculated based on that. A state machine is used to break down the gait cycle into nine different stages, and appropriate leg motions are generated to perform the required footstep using individual swing and stance leg controllers. Griffin et al. (2017) incorporated some notion of step timing adjustment into their planned capture point trajectories for walking. Utilising a QP optimisation scheme, adjustments to an existing planned capture point trajectory were undertaken to produce plans that reduced the amount of required knee bend (while remaining kinematically feasible).

Assuming flat ground and a constant CoM height, Koolen et al. (2016) presented a gait for the Atlas robot that generates instantaneous capture point trajectories at the beginning of each single support phase, and calculates desired Centroidal Moment Pivot (CMP) trajectories using a Proportional-Integral (PI) control law aimed at capture point tracking. The desired CMP is used to construct a desired linear momentum rate of change of the robot, which is subsequently unified with contact information and constraints in a low-level QP aimed at providing joint accelerations and ground reaction wrenches to the inverse dynamics used to calculate the required joint torques. Although a constant CoM height is assumed, the momentum-based control framework was successfully used both on rough terrain and stairs.

Parallel to the development of the instantaneous capture point and its applications to bipedal walking, Takenaka, Matsumoto and Yoshiike (2009) developed the essentially equivalent notion of the Divergent Component of Motion (DCM), and used it as a relaxed boundary constraint for transitioning between cyclic gait patterns on the ASIMO robot. Takenaka, Matsumoto, Yoshiike and Shirokura (2009) extended this method of pattern generation to running by including consideration of vertical and flywheel models in parallel to the three-mass Linear Inverted Pendulum Model already used for forwards locomotion planning. In addition to enforcing continuous ZMP and DCM locations at the transition points between the calculated gait patterns (by adding trapezoidal offsets to the planned ZMP waveforms), continuous flywheel moments were also enforced to control the inclination of the torso. Takenaka, Matsumoto, Yoshiike, Hasegawa et al. (2009) include details of the balance controller used to stabilise the upper body position during running of the ASIMO robot. In addition to joint angle, ground reaction force and body inclination control schemes, an implemented model ZMP control scheme uses horizontal and rotational sagittal accelerations of the torso based on feedback to stabilise the robot. This results in possible adaptations of step placement and timing in order to keep the robot balanced while satisfying all kinematic constraints.

The concepts of the instantaneous capture point and DCM were extended to 3D by Englsberger et al. (2015), and lead to the development of the notions of the Enhanced Centroidal Moment Pivot (eCMP) and Virtual Repellent Point (VRP). The 3D DCM is a point in space a certain distance in front of the CoM in the direction that it is instantaneously moving. That is,

$$\boldsymbol{\xi} = \mathbf{x} + b\dot{\mathbf{x}}, \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^3$ is the position of the CoM, $\dot{\mathbf{x}} \in \mathbb{R}^3$ is its instantaneous velocity, and b is a time-constant. The eCMP is a point that encodes the sum of all external forces $\mathbf{F}_{ext}$ (excluding gravity), and is located behind the CoM, in the exact opposite direction to which $\mathbf{F}_{ext}$ points. Specifically,

$$\mathbf{r}_{ecmp} = \mathbf{x} - \frac{b^2}{m}\mathbf{F}_{ext}, \tag{2.2}$$

where m is the mass of the robot. The VRP is equivalent to the eCMP, only with consideration of the force of gravity as well, i.e.

$$\mathbf{r}_{vrp} = \mathbf{x} - \frac{b^2}{m}\mathbf{F}_{tot}, \tag{2.3}$$

where $\mathbf{F}_{tot} = \mathbf{F}_{ext} + \mathbf{F}_g$. One can observe that the eCMP and VRP only differ in their z-coordinate, by the constant height $b^2 g$, and that the resultant DCM dynamics simply become

$$\dot{\boldsymbol{\xi}} = \frac{1}{b}(\boldsymbol{\xi} - \mathbf{r}_{vrp}). \tag{2.4}$$

This can be interpreted as the VRP point always exponentially 'pushing away' the DCM along the ray that connects them. Englsberger et al. (2015) use this fact to plan reference DCM trajectories for walking on uneven terrain based on VRP points that are chosen to be fixed at a constant height above the required footsteps. These are then tracked using a force-based DCM tracking control law based on modifications of the VRP. The associated reference CoM trajectories are calculated from the DCM trajectories using the convergent component of the CoM dynamics. Extensions that include continuous double support phases and heel-to-toe eCMP trajectories (as opposed to a fixed eCMP at the foot centre) are also presented.

Many modern methods of gait generation and control rely on the Divergent Component of Motion. Englsberger et al. (2017) and Mesesan et al. (2018) use polynomial splines to generate VRP and DCM trajectories analytically in closed form, and generate multi-step reference trajectories based on this. Englsberger et al. (2017), in particular, provide an explicit mechanism for push recovery via step adjustment, and propose a momentum-based disturbance observer for estimating and overcoming continuous perturbations. While only providing simulation results, real-robot experiments with the TORO

robot were later published in Englsberger et al. (2018), in the context of a QP-based whole-body control framework. Khadiv et al. (2016) also presented a method for incorporating step adjustments into a DCM planning and tracking scheme. Its application to a simulated robot with passive ankles is demonstrated, but it was found that its performance was highly dependent on the manually chosen step timing.

Hopkins et al. (2014) extended the concept of the DCM to a time-varying version, and used it to construct stepping trajectories with generic variability in the CoM height. Stair climbing using a pre-planned CoM height trajectory was demonstrated, albeit only in simulation. The scheme runs in real-time, but no provision is made for real-time step adjustment or trajectory replanning as a response to disturbances. Caron et al. (2019) used a whole-body admittance controller in conjunction with a DCM observer and feedback controller to perform stair climbing with the HRP-4 robot in an Airbus factory. Caron (2019) investigated the applicability of a whole-body admittance control strategy in combination with the Variable-Height Inverted Pendulum (VHIP) model. A linear feedback controller is devised for the nonlinear VHIP model that, where feasible, acts similar to linear control of the DCM, and utilises a CoM height variation strategy otherwise (when the ZMP reaches the edge of the support polygon). Henze et al. (2016) developed a passivity-based whole-body controller in purely Cartesian formulation that was used, amongst other things, for multi-contact balance control. The controller was later extended by Mesesan et al. (2019) to a generalised tasks formulation, and applied to walking of the TORO robot over compliant mattresses and pebble beds. Walking on grass and flat terrain with edge contacts was also demonstrated, albeit with relatively static and long step times of 1.2–1.7 s.

2.3 ROBOCUP WALKING APPROACHES

The RoboCup Humanoid Soccer competition is an environment where the analytical or academic beauty of a method is completely second to its proven robustness and functionality on the field. Interestingly, there are very few examples of successful RoboCup teams that use walking approaches similar to any of the finely crafted methods described above.[1] There could be many reasons for this, ranging from time and resources of the teams, to the necessity of having a method that can be retuned on short notice before any given soccer match, but one significant factor is likely the robot hardware. Most robots participating in the RoboCup Humanoid League do not even have the quality of actuators and sensors that would be required to reliably track a planned CoM or DCM trajectory, let alone with much higher step frequencies, highly dynamic changes of walking plans,

1 With notable exception, for example, of Team THORwIn (McGill et al., 2015).

moving obstacles, and completely unpredictable ground, ball and push disturbances.

Team Rhoban (Allali et al., 2019) won the KidSize league three years in a row using a walking engine `QuinticWalk` that extended their previous open source gait engine `IKWalk` (Rouxel et al., 2015) to use 5^{th} order polynomial splines, continuous gait command accelerations, and explicit double support phases (Allali et al., 2018).[2] The generated gait trajectories are executed open-loop, without consideration of any ZMP stability criteria or modelling of the dynamics, but a stabilisation module is used to defer commanding of the support transitions until it is deemed to be the right time based on the foot force sensors. This is used as a form of timing feedback to help avoid desynchronisation of the desired and actual lateral oscillations of the robot.

The widely successful Standard Platform League (SPL) RoboCup team B-Human (Röfer et al., 2019) has in the last years used a gait based on the rUNSWift Nao gait (Hengst, 2014). A manually tuned open-loop gait is stabilised using ankle feedback in the pitch direction, and timing feedback in the lateral direction. The ankle feedback is implemented in the form of pitch offsets that are proportional to low-pass filtered gyroscope measurements. The timing feedback is implemented by estimating the lateral location of the CoP using foot force sensors, and adjusting the start of the leg lifting profile to always occur right before the CoP crosses the zero line. During gameplay, an online learning scheme monitors the peaks in the observed gyroscope measurements and uses this to successively guess and refine suitable gains for the ankle pitch feedback scheme.

The SPL RoboCup team Nao Devils (Hofmann et al., 2018) uses a walk engine based directly on the original preview control gait by Kajita et al. (2003). One slight modification to the original method is the introduction of an additional cart-spring-damper to the model, forming the so-called Flexible Linear Inverted Pendulum Model (Urbann et al., 2015). Although original walking results were qualitatively moderate,[3] the method was successively refined over following years to improve performance. For instance, the delay in responsiveness of walking intent induced by the 1 s preview window was reduced by resetting the ZMP generation after each step (previously this was done only when tracking errors exceeded a configured threshold). A Proportional-Derivative (PD) controller controlling all leg joints except the hip yaw based on the gyroscope and angle sensors was also added.

Team THORwIn of the AdultSize league (formally Team DARwIn of the KidSize league) were widely successful in both leagues from the years 2011 to 2015 (McGill et al., 2015). Originally using the gait developed for the DARwIn-OP (see Yi et al., 2011 on page 10), for

2 The dominant team of the TeenSize and AdultSize leagues in recent years was Team NimbRo, which used exactly the approaches to walking presented in this thesis.

3 Video published alongside Urbann et al. (2015): `https://youtu.be/q3byPqjQ5a0`

the THOR robot they extended it to a hybrid locomotion controller that can switch online between two different walking controllers—a ZMP preview controller using linear quadratic optimisation and a simpler ZMP controller based on a closed-form solution to the LIPM equations (Yi et al., 2013). Smaller push disturbances were dealt with through reactive ankle pitch offsets and a mode whereby the robot stops walking and lowers its CoM to dampen out oscillations.

Although it has only seen isolated uses at RoboCup competitions, a distinct and impressive approach to balanced walking has been developed by Missura of team NimbRo in the form of the capture step framework (Missura, 2015). This approach adjusts both the step position and timing to preserve balance, based on the prediction of the CoM trajectory using the LIPM. The main advantage of this method is that it does not require forces, torques or the ZMP to be measured, making it more suitable for low-cost robots. More details of this method, including a discussion of both its features and shortcomings, can be found in a dedicated chapter in Allgeuer (2020).

2.4 DISCUSSION

It is a general problem of most ZMP preview control, MPC, capture point and DCM-based gaits that good actuator tracking performance and sensor feedback is required to make it work. This limits the applicability of such approaches to high quality hardware, and/or smaller robots that have favourable torque-to-weight ratios, especially if this is in combination with proportionally large feet (e.g. the Nao robot). Many state of the art approaches also require CoP/ZMP measurement through integrated force-torque sensors as well as precise physical models to make the planned motions feasible. These criteria are often not met when using lower cost robots such as the igus Humanoid Open Platform, in part due to lacking sensors, but mainly due to the limited robot stiffness and actuation quality.

With respect to the state of the art in bipedal walking, the following generalised observations can be made:

- Only select few methods can deal with true dynamic and real-time online adjustments of step timing for the purpose of balance. This is a problem, as in the face of even small disturbances lateral oscillations can lead to significant falls.[4]

- Many methods are evaluated using experiments that only perform a few forwards steps at a time (possibly over uneven terrain or stairs) and then stop. While this demonstrates that locomotion is fundamentally possible with the method, it is difficult to ascertain how robust the method truly is, and whether it would

4 Good demonstration: https://youtu.be/l9uvBD9zmsw

for example be suitable for real-life dynamically disturbed omni-directional walking over the full duration of a 10 minute soccer half-time (e.g. as required for RoboCup).

- Many state of the art methods that rely on tracking of generated motion plans take relatively slow (but large) steps (>1 s per step), as the motion of the robot generally needs to be quite controlled for the tracking to work well.
- Publications frequently show experiments in simulation only, with real-robot experiments lagging behind or only possible with great subsequent effort and adaptation. Englsberger et al. (2013) and the TORO robot is an example of this, with real-robot TORO walking experiments only being published five years later in Englsberger et al. (2018).

By contrast, the methods for walking presented in this thesis are:

- Targeted at low-cost robots with an Inertial Measurement Unit (IMU) and joint encoders, but no options for force sensing and/or meaningful CoP/ZMP localisation,
- Targeted at robots with position-controlled actuators (limiting the compliance and controllability of the interactions the robot can have with the ground),
- Equipped with step timing feedback in combination with many other stabilising mechanisms to ensure preservation of balance,
- Based on the idea of stabilising a core semi-stable open-loop gait with the use of feedback mechanisms,
- Highly dynamic,[5] in that they allow the robots to go to their very limits of balance while still trying to recapture control, and,
- Aimed at allowing continuous walking for long periods of time (more than 10 minutes), even in the face of significant unknown and uncategorised disturbances.

5 As opposed to the frequently observed 'quasi-static' walking style seen on more expensive platforms.

3 ACTUATOR CONTROL

In order for any motions generated for a robot to work as intended, robust, accurate and efficient tracking of the computed actuator trajectories is paramount. Even though the Dynamixel servo motors have an inbuilt configurable Proportional-Integral-Derivative (PID) position control loop, it is hard to tune this loop so that it works appropriately and equally in all situations. The approach of simply using very high gain settings, thereby making the robot very stiff, is not appropriate because it leads to problems with

- Overloading and overheating, leading to temporary or even permanent servo failure,
- Little to no damping in the motions, leading to motions that are not smooth, and prone to unwanted oscillation, and
- Increased self-disturbances of the robot, due to much harsher and sharper contacts—in particular impacts with the ground.

One fundamental limitation of using just the PID control loops inside each servo is that each control loop is operating completely independently. They have no global picture of the state of the robot, what the robot is trying to achieve, and what amount of torque they can each expect to require to follow their respective commanded trajectories. PID control loops are also agnostic to the specific contributing factors *why* a servo does not necessarily manage to follow its input commands, and can only react to disturbances when they have already been measured in the output, at which point it is too late to avoid them, and difficult to avoid them in a systematic way. In this chapter, we describe in detail the actuator control scheme, which seeks to use torque estimation and feed-forward compensation (in addition to servo-internal proportional feedback) to improve actuator tracking in a systematic targeted way. A general discussion of the pipeline how joint commands are generated, compensated and sent out to the servos can be found in Allgeuer (2020).

3.1 SERVO MOTOR MODEL

The servo motor model is a mathematical and electromechanical model of the behaviour of the used servo motors that incorporates both Direct Current (DC) motor and mechanical friction characteristics. It is used to allow for compensated control of the servos.

3.1.1 DC Motor Model

The Dynamixel servo motors used in the robots are built around standard brushed DC motors driven by a Pulse Width Modulation (PWM) approach. This means that we can model the motor as an effectively continuously variable voltage V_m applied to an armature resistance in series with an ideal motor winding, as shown in Figure 3.1. The armature current I flows through the armature resistance R, producing a voltage drop of RI from Ohm's law, and further through the ideal motor winding, which produces the torque τ on the output shaft. As a result of the rotation of the output shaft within the magnetic field of the motor, a voltage E is induced across the winding (due to electromagnetic induction) and is referred to as the back-Electromotive Force (EMF) voltage. The back-EMF voltage is proportional to the angular velocity ω of the output shaft, and is explicitly given by

$$E = k_e \omega, \tag{3.1}$$

where k_e is the so-called back-EMF constant of the motor. We conclude from Kirchhoff's voltage law, as applied to Figure 3.1, that

$$V_m = RI + E \tag{3.2a}$$
$$= RI + k_e \omega. \tag{3.2b}$$

As a simple model of a DC motor, we know that the torque generated by the motor is proportional to the current flowing through its windings, i.e.

$$\tau = k_t I, \tag{3.3}$$

where k_t is the so-called motor torque constant, and is generally considered to be equal to the back-EMF constant k_e. As a result,

$$I = \frac{\tau}{k_t}, \tag{3.4}$$

and substituting this into Equation (3.2b) gives

$$V_m = \frac{R}{k_t}\tau + k_e \omega. \tag{3.5}$$

While ω is the angular velocity of the shaft of the DC motor, the effect of gearing means that the angular velocity $\dot{q}$ of the servo motor joint axis is proportionally different, i.e.

$$\omega = k_g \dot{q}, \tag{3.6}$$

where k_g is the gear reduction ratio. The output torque τ_q at the joint axis is also scaled by the effect of gearing,

$$\tau_q = k_g \tau, \tag{3.7}$$

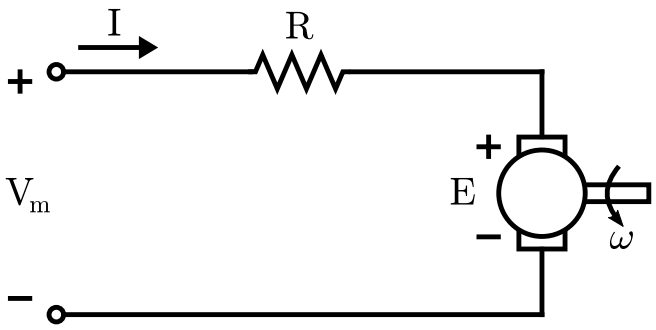


Figure 3.1: A simple electrical model of a DC motor. The voltage V_m applied to the motor results in an armature current I flowing through the armature windings of resistance R, and makes the motor spin with an angular velocity of ω. A back-EMF voltage E is generated as a result of the rotation of the output shaft in the magnetic field.

and is the sum of the true output torque τ_d, and the torque τ_f that is required to overcome friction. Thus, by rearrangement of Equation (3.7),

$$\tau = \frac{1}{k_g}(\tau_d + \tau_f). \tag{3.8}$$

From Equations (3.5) and (3.6), this leads to

$$V_m = \frac{R}{k_g k_t}\tau_d + k_g k_e \dot{q} + \frac{R}{k_g k_t}\tau_f. \tag{3.9}$$

The main question now is how to model the friction torque τ_f in terms of the joint angular velocity $\dot{q}$. As discussed in Schwarz and Behnke (2013), a suitable friction model involving both static and Coulomb friction terms, and an exponential transition (Stribeck curve) between them, is given by

$$\tau_f = \operatorname{sgn}(\dot{q})\big(\beta\tau_s + (1-\beta)\tau_c\big) + c_v\dot{q}, \tag{3.10}$$

where τ_s is the limit static friction torque, τ_c is the Coulomb friction torque, c_v is the viscous friction constant, and β is the interpolating factor

$$\beta = \exp\left(-\left|\frac{\dot{q}}{\dot{q}_s}\right|^{\delta}\right), \tag{3.11}$$

where $\dot{q}_s$ is the transition velocity between static and Coulomb friction, and δ is an empirical constant in the range 0.5 to 1.0 based on the material surfaces and properties. Based on Schwarz and Behnke (2013), a choice of $\dot{q}_s = 0.1\,\text{rad/s}$ and $\delta = 1.0$ was made for the servos in this thesis.

Substituting Equation (3.10) into Equation (3.9) gives

$$V_m = \frac{R}{k_g k_t}\tau_d + \left(k_g k_e + \frac{Rc_v}{k_g k_t}\right)\dot{q} + \frac{R\tau_c}{k_g k_t}\operatorname{sgn}(\dot{q})(1-\beta) + \frac{R\tau_s}{k_g k_t}\operatorname{sgn}(\dot{q})\beta,$$

which, with the definition of appropriate constants $\hat{\alpha}_0$, $\hat{\alpha}_1$, $\hat{\alpha}_2$ and $\hat{\alpha}_3$, can be simplified to

$$V_m = \hat{\alpha}_0 \tau_d + \hat{\alpha}_1 \dot{q} + \hat{\alpha}_2 \operatorname{sgn}(\dot{q})(1-\beta) + \hat{\alpha}_3 \operatorname{sgn}(\dot{q})\beta. \tag{3.12}$$

Recalling from Equation (3.11) that β is simply a function of $\dot{q}$ (the angular velocity of the servo motor joint axis), we can see that the effective voltage applied to the motor can therefore be expressed as a function of $\dot{q}$, τ_d (the true joint axis output torque) and four fixed constants.

3.1.2 Compensated Motor Control

The Dynamixel servo motors used in the robots are position-controlled, and have an internal position-based PID loop with configurable gains for control purposes. In order to centralise the control of the servos and overcome the limitations identified on page 21 of just using servo-level PID control loops, we nominally configure each servo to only use proportional (P) control, and find an approach to use the servo model for global control instead.

In pure proportional mode, the effective voltage applied to the motor is proportional to the battery voltage and the motor PWM duty cycle, which in turn is proportional to the position error and P gain. Thus, with incorporation of a proportionality constant K_c that governs the scaling from position errors to PWM duty cycles, the effective motor voltage V_m applied by the servo electronics to the DC motor at any given time is given by

$$V_m = V_b K_c K_P (q_d - q), \tag{3.13}$$

where V_b is the battery voltage, K_P is the P gain, q_d is the desired position of the servo, and q is the current position of the servo. Therefore, if we wish for the servo to follow a trajectory where at some instant the position needs to be q, the velocity needs to be $\dot{q}$, and the predicted amount of torque required to follow the trajectory is τ_d,[1] the required setpoint for the position P control loop in order to achieve this is given by

$$\begin{aligned} q_d &= q + \frac{1}{V_b K_c K_P} V_m, \\ &= q + \frac{1}{V_b K_c K_P} \left(\hat{\alpha}_0 \tau_d + \hat{\alpha}_1 \dot{q} + \hat{\alpha}_2 \operatorname{sgn}(\dot{q})(1-\beta) + \hat{\alpha}_3 \operatorname{sgn}(\dot{q})\beta \right) \\ &= q + \frac{1}{V_b K_P} \left(\alpha_0 \tau_d + \alpha_1 \dot{q} + \alpha_2 \operatorname{sgn}(\dot{q})(1-\beta) + \alpha_3 \operatorname{sgn}(\dot{q})\beta \right), \end{aligned} \tag{3.14}$$

1 That is, the predicted amount of true output torque on the joint axis required to follow the trajectory from a dynamics perspective (e.g. due to gravity, inertias, contacts, …), not including the torque that is required to overcome joint friction.

where the factor of K_c has been absorbed into the constants $\hat{\alpha}_*$ to give α_* (for $* = 0, 1, 2, 3$).

Equation (3.14) embodies the entire servo motor model, and forms an essential step of the greater robot control scheme. Given a servo motor, suitable values of $\alpha_\star$ are first identified and tuned (see below). Then, given the required q, $\dot{q}$ and τ_d of a joint trajectory in real-time, Equation (3.14) is evaluated (recall that β is a function of $\dot{q}$) and the resulting q_d is sent to the servo as its target position. K_P is known as it is the current P gain of the servo, and is calculated from the joint effort command, and V_b is known as it is measured and reported to the PC software in every cycle. Note that by design, even though q_d is sent as the target position of the servo, it is *not* intended that the servo actually reaches this position. The whole servo model calculation aims to ensure that if the servo tries to reach q_d, then it will instead only reach q (as actually desired) due to the effect of friction and external forces. The only missing step at this point is how to calculate the required τ_d, also known as the feed-forward torque, given the commanded joint trajectories of all the joints of a robot. This is addressed in Section 3.2.

The tuning of the α_* constants for the servo motors used in this thesis (mainly Dynamixel MX-106 servos) was done in Schwarz and Behnke (2013) using Iterative Learning Control (ILC). As performing motions on an entire free-standing robot is difficult and not suitably repeatable, a test bench was constructed whereby the torso of the NimbRo-OP was fixed to a table, and a realistic hip pitch trajectory was executed and evaluated for tracking accuracy. Estimates of the optimal values of α_* for tracking were successively refined with every execution of the trajectory. A total of 12 ILC iterations were required to converge the estimates of α_* to their final values, and a maximum trajectory deviation of approximately 0.02 rad was achieved. Refer to Schwarz and Behnke (2013) for more details on the method and results. The tuning results for later robots and servo types were not independently published.

3.2 FEED-FORWARD TORQUE ESTIMATION

As indicated previously, the only missing link in the pipeline of the application of the servo motor model that remains is the calculation at every instant of the desired true output torque τ_d. This torque is referred to as the feed-forward torque, and the calculation thereof is the subject of this section.

3.2.1 Single Support Models

Given the desired positions $\mathbf{q}$, velocities $\dot{\mathbf{q}}$, and accelerations $\ddot{\mathbf{q}}$ of all the joints in a robot at some instant in time, it is possible to calculate the associated required joint torques to achieve such a

state (under consideration of external force conditions) using inverse dynamics. A Unified Robot Description Format (URDF) model of the robot, which specifies all dimensions, offsets, link masses and inertia tensors, is available to the robot control process. This physical model is converted in software to a format that the Rigid Body Dynamics Library (RBDL) (Felis, 2017) can work with. Given any external force and loading conditions of the robot, specified as the magnitude and direction of gravity and any contact forces, the RBDL library can then use the Recursive Newton Euler Algorithm (RNEA) to calculate all corresponding internal joint torques.

One difficulty of the inverse dynamics calculation is that it is unable to automatically resolve indeterminate contacts. That is, if both feet of the robot are contacting the ground, then the amount of contact force and torque going through each foot cannot be calculated, and instead needs to be resolved heuristically. This is less a limitation per se of the RBDL library, but more a limitation of the availability of information about the dynamic state of the robot, i.e. the position, velocity and acceleration of the robot's free-floating base relative to the environment. In order to overcome this limitation, we introduce the concepts of *single support models* and *support coefficients*.

A single support model of the robot relative to a particular link is the dynamic model that assumes that the nominated link is rigidly fixed in space (with 6 DoF reaction forces and torques), while the rest of the robot can move freely. One of these single support models is created for the trunk link, as well as for each tip link, i.e. link at an end of the kinematic tree (e.g. foot, hand and head links). At each instant in time, a support coefficient in the range $[0, 1]$ is specified for each single support model, and expresses the proportion of the weight of the robot that is expected to be carried by the associated link at that time. The sum of the support coefficients of all single support models at any instant should always be 1.0 for obvious reasons. Seeing as during walking only the feet of a humanoid robot intentionally touch the ground, often only the support coefficients of the left foot and right foot are considered. These two support coefficients are often just referred to as the support coefficients of the left and right leg respectively. The single support models and associated support coefficients are the heuristic entity that allow the case of indeterminate ground contacts to be resolved.

3.2.2 Joint Torque Estimation

Given the desired positions $\mathbf{q}$, velocities $\dot{\mathbf{q}}$, and accelerations $\ddot{\mathbf{q}}$, the joint torques required to overcome and accelerate the link masses and inertias are first evaluated using the trunk link single support model. No gravity or other external forces are applied in this first inverse

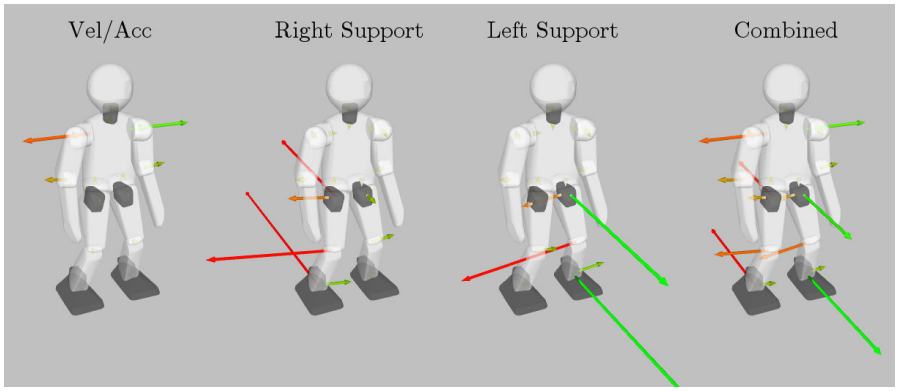


Figure 3.2: Visualisation of the feed-forward joint torque estimation method for the case of a standing robot that is accelerating its right arm forwards and left arm backwards. The left-most image shows the first step, which considers (relative to a fixed trunk) the joint velocities and accelerations, and the resulting torques required to overcome link inertias. The centre images show the torques required to overcome gravity based on the right and left foot single support models. The right-most image shows the result of combining all three previously calculated torques using superposition, where support coefficients of 0.5 are used for each foot.

dynamics computation, as this is handled separately in a second step, and the trunk link rarely has a non-zero support coefficient anyway.

In the second step, the inverse dynamics of all the available single support models are computed in turn, with just the positions $\mathbf{q}$ and the force of gravity and any other explicitly modelled external forces applied. Based on the principle of superposition, the resulting joint torques from each computation are combined together using the support coefficients as weights, and further additively combined with the joint torques that were computed in the first step (see Figure 3.2 for a visual example). This yields the final feed-forward joint torques that are used for the input τ_d of the servo motor model. As a necessary simplification for feedback stability reasons, the gravitational acceleration vector is always assumed to point in a fixed downwards direction relative to the trunk in every single support model. Perhaps unintuitively, the error in this assumption is actually empirically less than if orientation feedback is used, due to the complex nature of ground contacts.

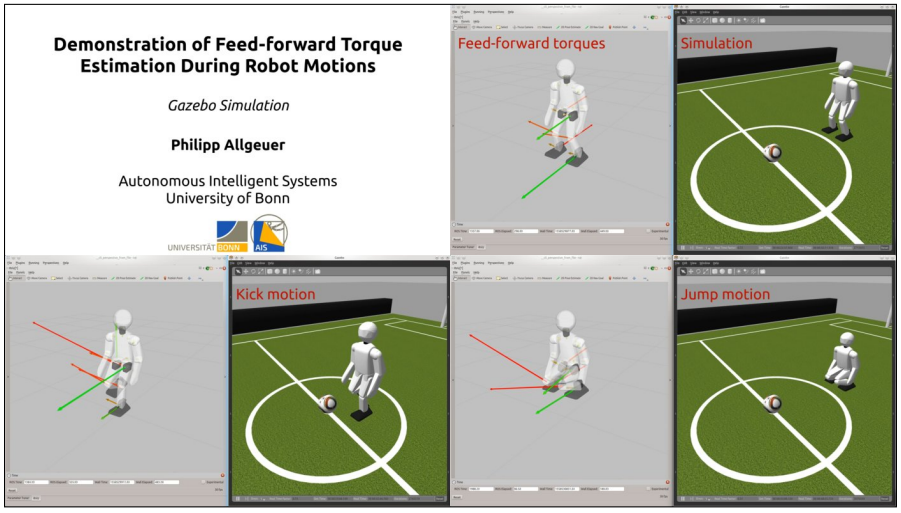
It may not immediately be obvious why the application of the joint velocities and accelerations in the first step needs to be separated from the remaining single support model evaluations. The core problem is that the foot-to-ground contacts are not always well-modelled by the assumption that the foot link is completely rigidly fixed in space, i.e. rigidly connected to the ground. If we consider the case that the robot commands a quick adjustment to the ankle pitch of both its feet, then in reality we know that the feet will likely take up a small part of the motion elastically, and react to the rest by temporarily lifting off the

ground at one of their edges until the robot starts tipping as a reaction. These non-rigid effects become more pronounced when the feet are located on soft surfaces, like for instance artificial grass. According to the single support models of the feet however, the ankles need to generate an enormous amount of torque in order to accelerate the entire torso and weight of the robot, and be able to change their pitch. Clearly, this does not reflect reality, and overestimates the torques required in order to perform such ankle motions. In the NimbRo-OP robot, this systematic overestimation sometimes led to dangerous oscillatory instabilities, in particular because the NimbRo-OP had very flexible feet. The approach of separating the calculation of the inertial joint torques to a separate evaluation of just the trunk link single support model solved this problem.

3.3 EXPERIMENTAL RESULTS

Figure 3.2 shows an example of the feed-forward torque estimation method being applied to a standing posture of the igus Humanoid Open Platform, in the instant that it started lifting its right arm forwards and left arm backwards. The four 3D visualisations show the pose of the robot in the first frame of the motion, i.e. the pose where the arms are still accelerating to move in their respective directions, along with the feed-forward torques that were calculated for that instant. The left-most image shows the result of the evaluation of the trunk link single support model (trunk link fixed in space), considering only the velocities and accelerations of each joint, and the resulting inertial torques. From the right-hand rule, it can be observed from the green block arrow on the left shoulder that the sign of the torque that is estimated to be required for the shoulder pitch does in fact accelerate the arm backwards, and that a similar but opposite torque on the right shoulder accelerates the right arm forwards. Non-zero torques are also estimated, as expected, for the respective elbow pitches, as the mass of the lower arms accelerating forwards/backwards results in torque being required to hold the positions of the elbow joints.

The centre two images in Figure 3.2 show the result of the evaluation of the right and left foot single support models respectively, considering only gravity, and not the velocities and accelerations of the joints. It can be observed in each case that the joints in the respective support leg required significant torque to hold the weight of the robot, and that the knee joint in the respective free leg actually required a small amount of torque in the opposite direction to keep the lower legs in their position against gravity. The right-most image in Figure 3.2 shows the superposition of the torques calculated in the previous three, with consideration of the commanded support coefficients κ_r and κ_l for the right and left legs respectively. Specifically, if $\boldsymbol{\tau}_i$ corresponds to the inertia-related torques from the left-most image, and $\boldsymbol{\tau}_r$ and $\boldsymbol{\tau}_l$



Video 3.1: Demonstration of the actuator control scheme, and in particular feed-forward torque estimation scheme, on the igus Humanoid Open Platform in Gazebo simulation.
https://youtu.be/4AQRvCpquC8
Demonstration of Feed-forward Torque Estimation During Robot Motions

correspond to the gravity-related torques from the centre two images, then the final feed-forward torques shown in the right-most image correspond to

$$\boldsymbol{\tau}_d = \boldsymbol{\tau}_i + \kappa_r \boldsymbol{\tau}_r + \kappa_l \boldsymbol{\tau}_l, \tag{3.15}$$

where in this case $\kappa_r = \kappa_l = 0.5$, as it was assumed that both legs carry an equal proportion of the weight of the robot ($\kappa_r = \kappa_l$), and

$$\kappa_r + \kappa_l = 1. \tag{3.16}$$

It can be observed from Figure 3.2 that the final combined feed-forward torques intuitively capture both the effects of gravity and the velocities and accelerations of the joints. A live demonstration of the commanded feed-forward torques during motions of an igus Humanoid Open Platform in Gazebo simulation is provided in Video 3.1.

An example of the effect of the full actuator control scheme on the commanded servo target positions is shown in Figure 3.3. The figure shows the robot in a balancing pose on its right leg, with the corresponding final feed-forward torques τ_d visualised using 3D block arrows at each joint. The left image shows the commanded joint positions from the motion modules, i.e. the servo target positions that would be sent if the actuator control scheme were disabled, and the right image shows the servo target positions that were computed using Equation (3.14) and subsequently sent out over the Dynamixel bus. The difference in leg joint positions between the left and right images can clearly be identified, and observed to correspond proportionally to the calculated amount of feed-forward torque in each of the respective

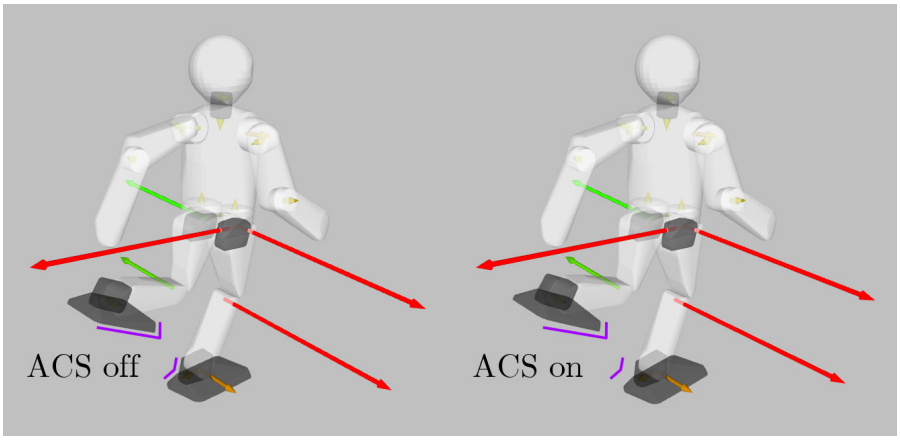


Figure 3.3: Comparison of the commanded servo target positions with and without the actuator control scheme enabled, for a robot balancing on its right leg. The left image shows the pose $\mathbf{q}$ that the robot is trying to achieve, and the right image shows the servo target positions $\mathbf{q}_d$ that are computed as a result of the calculated feed-forward torques (visualised by the 3D arrows). The target positions for the servos in the right leg are further forward than the pose $\mathbf{q}$ they are actually desired to reach (refer to the fixed purple lines), as when the weight of the robot comes into play, one can imagine that it would push the leg further backwards than commanded, towards $\mathbf{q}$.

joints.[2] Note that it makes sense, for example, that the actuator control scheme commands a pose for the right leg that is further forwards and outwards than the pose commanded by the motion modules, as intuitively one can see that the weight of the robot acts to press the leg back into the opposite direction again, towards the truly desired pose.

Figure 3.4 shows plots of the tracking performance of the left knee pitch joint during a walking experiment of the igus Humanoid Open Platform. A clear difference in tracking performance can be observed depending on whether the actuator control scheme is enabled or disabled, even if due to disturbances and real world inaccuracies the tracking with the scheme enabled is still not perfect. It was observed in the walking experiments that the battery life of the robot was longer with the actuator control scheme enabled, as opposed to disabled, suggesting that it can also increase the energy efficiency of robot motions. This observation is empirically supported by the servo model parameter tuning experiments of Schwarz and Behnke (2013), in which 40 full steps of the NimbRo-OP robot consumed 189 J with the servo model disabled, and 140 J with the model enabled.

2 In this case, the friction terms in Equation (3.14) had no effect as the commanded pose was stationary, i.e. $\dot{q} = 0$ for all joints.

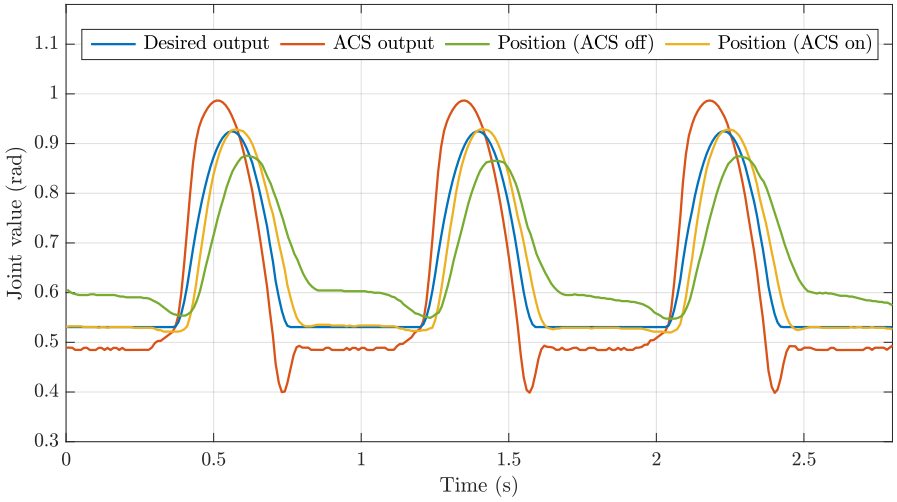


Figure 3.4: Plot of the left knee pitch tracking performance during a walking experiment on the igus Humanoid Open Platform. The desired output trajectory of the knee joint is given in blue. Passing this directly as the commanded servo targets results in the green output position waveform, which differs significantly from the desired waveform. With the actuator control scheme enabled however, the red waveform is computed in a feed-forward manner and used as the commanded servo targets instead. The resulting yellow position waveform corresponds much more closely to the intended waveform, albeit still with some signs of delay.

3.4 DISCUSSION

Together, the servo motor model (Section 3.1) and method for feed-forward torque estimation (Section 3.2) constitute the actuator control scheme. All in all, the actuator control scheme is able to compensate effectively for many factors, including battery voltage, servo P gain, gravity, joint friction, inertia, and the relative loadings of the legs. All of these factors influence how well the servos track their commanded trajectories, and by specifically considering and modelling them, the performance of the robot becomes significantly more consistent across the wide range of possible robot states and conditions.

One possible downside of the actuator control scheme is that it requires a relatively accurate and highly-tuned physical model of the robot. In the case of the robots used in this thesis, this is in general not a problem due to the use of Computer-Aided Design (CAD) and 3D printing, but it is potentially a problem for other more custom robots that do not have an accurate URDF model (with masses and inertias), and do not need one anyway for other higher level motion planning algorithms. If no masses and inertias are available for a robot, then the actuator control scheme can still be used, just with the feed-forward torques set to zero, or replaced with simplified calculated values based at each time step on the support coefficients and weight of the robot. Doing this would sacrifice the inertia compensation and the accuracy

of the gravity compensation, but would still leave the friction, battery voltage and servo P gain compensation intact.

Positive effects of the actuator control scheme include reduced servo overheating and wear, increased battery life due to increased energy efficiency, reduced issues with impacts and self-disturbances, and the ability to increase joint compliance while maintaining good levels of trajectory tracking. One particular benefit of the control scheme is that the motions performed by the robot become more consistent across the full spectrum of possible battery voltage values. This is particularly useful when designing, for example, keyframe motions, or for that matter essentially any other motions that have a significant open-loop component to them as well.

4 SENSOR CALIBRATION

All sensor data available to the robot is read in by a main robot control process at a rate of 100 Hz. In order to properly use the available sensor data—for instance for the estimation of the orientation of the robot (see Chapter 7)—it needs to be calibrated first. This involves calibrating the data, in particular, from the gyroscope and accelerometer sensors.

Some calibration methods rely on others already having been successfully performed. For example, the gyroscope bias can only be correctly estimated if the Inertial Measurement Unit (IMU) orientation and gyroscope scale calibrations have already been completed. As a result, the full calibration procedure of a robot is somewhat sensitive to order. A list of the IMU calibration procedures is given as follows, in the order that the calibrations are generally performed:

- IMU orientation calibration
- Gyroscope scale calibration
- Gyroscope bias calibration
- Online gyroscope bias autocalibration

A much more complete list and discussion of sensor calibration procedures is provided in Allgeuer (2020).

4.1 INERTIAL MEASUREMENT UNIT CALIBRATION

One very important class of robot calibrations is given by the IMU calibration procedures. In this section, we refer to the IMU as consisting of only the 3-axis gyroscope and accelerometer sensors, and assume that these are rigidly connected to each other.

4.1.1 IMU Orientation Calibration

The first aspect of the IMU that needs to be calibrated is the orientation offset of the IMU sensors relative to the main trunk link of the robot.[1] As the gyroscope and accelerometer chips are mounted in a fixed and aligned arrangement relative to one another, tuning only the offset of the *accelerometer* chip relative to the trunk suffices. This is done by sampling low-pass filtered values of the accelerometer at various well-defined orientations of the trunk, and computing the best-fitting

1 It is assumed that the sensors are rigidly attached to the trunk of the robot, so the required offset is constant, and thus can be tuned.

estimate of ${}^{B}_{I}R$, the orientation of the accelerometer (i.e. IMU) frame {I} relative to the body-fixed (i.e. trunk-fixed) frame {B}.

The tilt rotation component (see Section 5.3.3) of ${}^{B}_{I}R$ is computed from measurements of the 3-axis acceleration ${}^{I}\mathbf{a}$ when the robot trunk is manually positioned to be perfectly upright relative to gravity, i.e. so that the body-fixed and global z-axes align and point upwards. That is, if ${}^{I}\bar{\mathbf{a}}_{u}$ is the mean accelerometer value measured in this position, then we can naturally define

$$ {}^{I}\mathbf{z}_{B} = \frac{{}^{I}\bar{\mathbf{a}}_{u}}{\left\| {}^{I}\bar{\mathbf{a}}_{u} \right\|}, \tag{4.1} $$

and from this derive the required tilt rotation component. The remaining yaw rotation component of ${}^{B}_{I}R$ is resolved by averaging out the fused yaws calculated from the measurements of ${}^{I}\mathbf{a}$ that are obtained when the robot trunk is lying perfectly on its front, back and/or side. Joined together, the yaw and tilt rotation components together specify the complete IMU orientation offset ${}^{B}_{I}R$, as required.

4.1.2 Gyroscope Scale Calibration

Once the orientation of the IMU in the trunk is known, the magnitude of the gyroscope sensor values needs to be compensated for scale and temperature. The trunk orientation computed by the attitude estimator presented in Chapter 7 is used for this purpose. Only the gyroscope and accelerometer values are applied as inputs to the estimator, and the value of the openly integrated fused yaw component of the output (see Section 5.3.1) is monitored while the robot trunk is manually rotated about its upright z-axis (relative to gravity). The exact calibration procedure is given as follows:

1. Position the robot so that the trunk is perfectly upright relative to gravity.
2. Indicate to the software that the gyroscope scale calibration is starting.
3. **Rotation 1:** While keeping the trunk as upright as possible, yaw the robot about its (upright) z-axis by exactly $K \geq 1$ revolutions in a counterclockwise (CCW) direction. This corresponds to an applied yaw rotation of $\Psi_1 = 2\pi K$ radians.
4. Indicate to the software that the calibration midpoint has been reached.
5. **Rotation 2:** While still keeping the trunk as upright as possible, yaw the robot back to its original orientation by applying exactly K revolutions in the opposite direction. This corresponds to an applied yaw rotation of $\Psi_2 = -2\pi K$ radians.

6. Indicate to the software that the original orientation has been reached again.

Based on the measured 3-axis gyroscope values and changes in fused yaw throughout both rotations, an *unbiased* estimate of a gyroscope scale factor k_g can be obtained for the current measured temperature. Doing this once for a high temperature and once for a low temperature allows coerced linear interpolation to be used to select an appropriate scale factor $k_g(T)$ for any future measured temperature T. If ${}^I\hat{\mathbf{\Omega}}$ is a raw measured gyroscope value, the corresponding scale-corrected gyroscope value ${}^I\hat{\mathbf{\Omega}}_s$ is given by

$$ {}^I\hat{\mathbf{\Omega}}_s = k_g(T)\,{}^I\hat{\mathbf{\Omega}}. \tag{4.2} $$

Note that {I} is the coordinate frame defining the x, y and z-axes of the IMU sensors, and is therefore also the frame relative to which the gyroscope measurements are expressed.

4.1.3 Gyroscope Bias Calibration

The gyroscope sensor is modelled as having a non-zero bias $\mathbf{b}_\Omega$ that offsets all of the scale-corrected measured angular velocities ${}^I\hat{\mathbf{\Omega}}_s$. That is, if ${}^I\mathbf{\Omega}$ is the true angular velocity of the IMU at some instant in time, it is modelled that

$$ {}^I\hat{\mathbf{\Omega}}_s = {}^I\mathbf{\Omega} + \mathbf{b}_\Omega + \mathbf{w}_\Omega, \tag{4.3} $$

where $\mathbf{w}_\Omega$ is zero mean sensor noise. If the assumption is made that $\mathbf{b}_\Omega$ is a fixed constant over all time, then it is possible to calibrate the gyroscope bias only once, and simply subtract it from every future scale-corrected measurement ${}^I\hat{\mathbf{\Omega}}_s$, as

$$ {}^I\hat{\mathbf{\Omega}}_s - \mathbf{b}_\Omega = {}^I\mathbf{\Omega} + \mathbf{w}_\Omega. \tag{4.4} $$

If the robot (and therefore IMU) is completely stationary for some period of time, we know that ${}^I\mathbf{\Omega} = 0$, and therefore that

$$ {}^I\hat{\mathbf{\Omega}}_s = \mathbf{b}_\Omega + \mathbf{w}_\Omega. \tag{4.5} $$

As such, if we low-pass filter the obtained values of ${}^I\hat{\mathbf{\Omega}}_s$ during this period of time with a moderate 90% settling time $T_{s,M}$ (on the order of seconds), the effect of $\mathbf{w}_\Omega$ diminishes (as it is zero mean) and a highly accurate estimate of $\mathbf{b}_\Omega$ results. This process has been implemented as a gyroscope bias calibration procedure that can manually be triggered in the software. After waiting a number of seconds for the bias estimate to converge, the low-pass filtered value of ${}^I\hat{\mathbf{\Omega}}_s$ is captured and saved as the bias estimate $\mathbf{b}_\Omega$.

4.1.4 Online Gyroscope Bias Autocalibration

The assumption that the gyroscope bias is constant over all time is not always accurate. To overcome this, an online method of gyroscope bias autocalibration has been developed that allows even brief periods of time that the robot is stationary during operation to be automatically detected, and leveraged to improve the gyroscope bias estimate. The robot is considered to be stationary if the difference between ${}^{I}\hat{\boldsymbol{\Omega}}_s$—see Equation (4.2)—and a moderately low-pass filtered version of ${}^{I}\hat{\boldsymbol{\Omega}}_s$ (90% settling time $T_{s,M} \approx 2\,\text{s}$) is less than a given threshold in norm for a configured duration of time (≈1.5 s). The norm threshold is chosen so that it is not much larger than the general maximum magnitude of sensor noise $\mathbf{w}_\Omega$, so as to ensure that the robot really is stationary when the condition is satisfied. The gyroscope bias adjustment process starts (time $t_a = 0$) every time the robot is detected to become stationary, and stops whenever the threshold is even just temporarily exceeded.

In addition to the moderate low-pass filter running on ${}^{I}\hat{\boldsymbol{\Omega}}_s$, a low-pass filter with an even slower 90% settling time ($T_{s,S} \approx 8\,\text{s}$) is used in parallel, in order to improve the noise rejection ability of the estimated gyroscope bias $\mathbf{b}_\Omega$ when the robot is stationary for longer, i.e. when a larger t_a has been reached. Let $\boldsymbol{\Omega}_M$ be the filtered angular velocity from the moderate low-pass filter, and $\boldsymbol{\Omega}_S$ be the equivalent output angular velocity of the slower low-pass filter, which is automatically reset to the value of $\boldsymbol{\Omega}_M$ in the instant when $t_a = 0$, i.e. when the adjustment process starts. For as long as the robot is stationary, in each measurement cycle the estimated gyroscope bias $\mathbf{b}_\Omega$ is interpolated linearly towards the so-called target bias $\mathbf{b}_T$, given by

$$\mathbf{b}_T = u\boldsymbol{\Omega}_S + (1-u)\boldsymbol{\Omega}_M, \tag{4.6}$$

where u is 0 when the bias adjustment process starts, and interpolates up to 1 over the next $T_{s,S}$ seconds, i.e.

$$u = \text{interpolateCoerced}\big([0, T_{s,S}] \to [0,1],\, t_a\big). \tag{4.7}$$

In each cycle, the bias $\mathbf{b}_\Omega$ is updated using the formula

$$\mathbf{b}_\Omega \leftarrow \mathbf{b}_\Omega + \alpha_B(\mathbf{b}_T - \mathbf{b}_\Omega), \tag{4.8}$$

where $\alpha_B \in [0,1]$ is the smoothing factor of $\mathbf{b}_\Omega$, given by

$$T_{s,B} = \text{interpolateCoerced}\big([0, t_d] \to [T_{s,slow}, T_{s,fast}],\, t_a\big), \tag{4.9a}$$

$$\alpha_B = 1 - 0.10^{\frac{\Delta t}{T_{s,B}}}, \tag{4.9b}$$

where Δt is the measurement cycle time, t_d is a configured fade duration (≈1.2 s), and $T_{s,slow}$ and $T_{s,fast}$ are configured slow and fast gyroscope bias 90% settling times, respectively. Note that Equation (4.8)

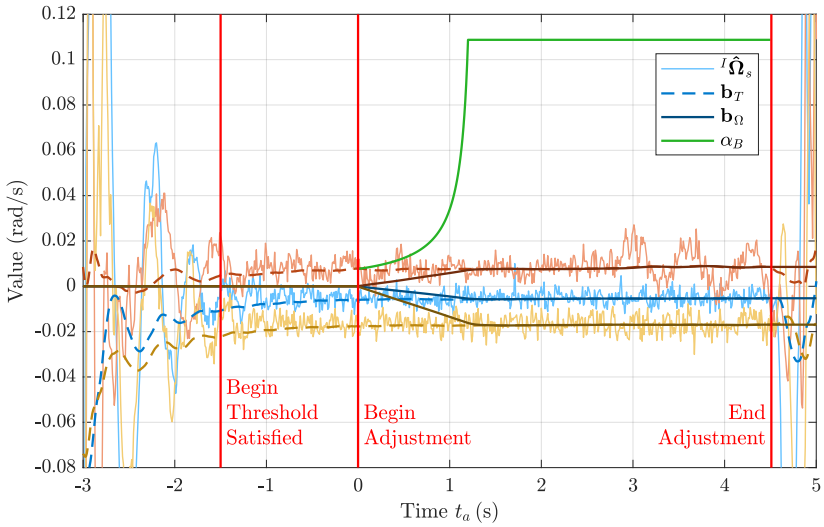


Figure 4.1: Time plots of the gyroscope autocalibration scheme in action. As indicated by the legend, the thin light lines correspond to the components of the scale-corrected gyroscope value $^I\hat{\mathbf{\Omega}}_s$, the thick dark lines correspond to the components of the estimated gyroscope bias $\mathbf{b}_\Omega$, and the dashed lines correspond to the components of the target bias $\mathbf{b}_T$, towards which $\mathbf{b}_\Omega$ is faded during the adjustment process. Note that before and after the beginning of adjustment, the value of $\mathbf{b}_T$ is just $\mathbf{\Omega}_M$, the filtered angular velocity from the moderate low-pass filter. The gyroscope bias smoothing factor α_B is shown for the interval where adjustment is active. After the norm threshold is violated at $t_a \approx 4.5\,\text{s}$, the estimated bias $\mathbf{b}_\Omega$ is no longer adjusted, and simply retains its value.

is the standard update equation for a first-order low-pass filter, but one where the smoothing factor α_B is continuously variable with time.

Effectively, the described autocalibration scheme attempts to extract from the measured gyroscope values as much useful information as possible, in as little time as possible, but without risking using the data too early and temporarily degrading the bias estimation accuracy. This is why the scheme initially waits a small amount of time for the robot to be stable, and then only weakly uses the data at first to improve the bias estimate, before becoming increasingly confident as further time elapses and the robot is still perceived to be stable. The use of low-pass filtering ensures that no sudden jumps in the bias estimate occur, and that the noise in the sensor can be robustly and reliably rejected. In addition to being automated and having a quicker responsiveness, these are both properties that do not apply to the manual calibration scheme presented in Section 4.1.3.

A plot of the gyroscope bias autocalibration method in action is shown in Figure 4.1. Initially, the robot is in motion, as can be seen from the gyroscope values, but after stopping and waiting a small amount of time for the motion of the torso to settle, the bias adjustment

process automatically triggers at $t_a = 0\,\text{s}$, and fades the initially zero bias estimate $\mathbf{b}_\Omega = (b_{\Omega x}, b_{\Omega y}, b_{\Omega z})$ to its appropriate filtered measured value. After only 1.25 s, the estimate reaches its final value, and only varies very slightly thereafter due to the residual effect of sensor noise.

4.2 SENSOR CALIBRATION IN THE BIGGER PICTURE

This chapter detailed how the raw acquired sensor data can be processed into a more useful form with the help of numerous calibration procedures. How this processed sensor data is filtered and used for higher level state estimation however (required for the purposes of motion feedback control), has not been addressed yet. Chapter 7 does this, and details how the calibrated sensor data can be fused together in an attitude estimation process to estimate the orientation of the robot. Before we can deal with the estimation of orientations and headings of the robot however and use these to construct bipedal walking gaits, we first need a greater understanding of rotations and general rotation formalisms in 3D. In particular, we need to be able to answer simple questions like

What is 'yaw' and how should we quantify it?

Is there more than one possible definition of yaw?

Which definition corresponds to our natural intuition?

Chapters 5 and 6, i.e. the next two chapters, deal with these and many other related questions, including for example questions pertaining to the concepts of 'pitch' and 'roll'. Building on this gained knowledge, Chapters 7 and 9 (and beyond) then develop effective algorithms for the tasks of state estimation and bipedal walking.

5

REPRESENTATIONS OF 3D ROTATIONS

In order to arrive at a gait that uses orientation feedback to maintain stability, the orientation of the robot must first be estimated and represented in a way that is both meaningful and useful. It is shown in Chapter 7 how 3D nonlinear filtering techniques can be used to estimate the orientation of a robot, but the final result is given by four quaternions parameters in the range from -1 to 1. On the face of it, without making any further conversions or calculations, it is not obvious how to interpret these four parameters so as to know basic things about the resulting estimated orientation, such as what the heading of the robot is, or how close the robot is to falling down in some direction. It is also not an easy task for a human to interpret the four parameters to understand what the exact nature of the orientation is. This chapter seeks to develop methods and concepts that can be used to extract meaning from such expressions of orientation, both for algorithmic purposes, for example to stabilise motions using orientation feedback, and for the ease of human interpretation and visualisation.

The applications of newly developed formulations of, for example, yaw are also useful deeper inside the state estimation itself. In Section 7.5.2, the concept of *fused yaw* is used to resolve a measured orientation of the Inertial Measurement Unit (IMU) in the case that no magnetometer is available. In this chapter, we take a systematic approach to placing the idea of fused yaw into context by developing and presenting three new highly intertwined representations of rotations, namely *tilt angles*, *fused angles* and the *tilt phase space*, that all depend fundamentally on this notion of yaw. Rotation representations, interchangeably referred to as rotation parameterisations, are simply put a way of assigning numeric values or *parameters* to a rotation. Here we are considering the three-dimensional space of 3D rotations, so the number of parameters must be at least three. Some rotation representations consist of exactly three parameters, like Euler angles and rotation vectors, but some consist of four, like quaternions and axis-angle pairs, and others consist of even more, like rotation matrices, which consist of nine parameters. A full review of existing rotation representations is given in Section 5.2.

While this chapter has been kept relatively focused on presenting the new rotation representations, how to work with them, and what basic properties they have, Chapter 6 is dedicated entirely to answering the question of why the development of new rotation representations was even necessary, and why Euler angles are not an appropriate solution.

All of the presented conversions and algorithms for working with tilt angles, fused angles and the tilt phase space have been implemented in both C++ and Matlab, and have been released open source for anyone to use (Allgeuer, 2018b; c), as a means of providing support with using these new representations. The libraries are seen as a test bed for the development of numerical rotation-related algorithms.

5.1 MOTIVATION AND AIMS

This work on 3D rotations was motivated by the development of gait stabilisation algorithms for bipedal robot platforms, or more generally, the analysis and control of balancing bodies in 3D. Walking bipedal robots are seen to be balancing bodies as they are under constant influence of gravity, and constantly aim to keep their state of balance upright and stable, despite changing support conditions and possible disturbances. The need for the development of new representations of 3D rotations emerged from the need to answer certain basic questions about the orientations of such walking robots—questions that were not well answered by any other existing rotation parameterisations.

5.1.1 Core Aims: Amount of Rotation in the Major Planes

Keeping with the context of walking bipedal robots, suppose we have a robot with a body-fixed coordinate frame {B} attached to its trunk. Assuming the global orientation of this frame ${}^{G}_{B}q$ can be estimated, e.g. using IMUs and other sensors, the state of balance of the robot is entirely encoded in this rotation. It is not uncommon for a gait to treat the sagittal and lateral planes of balance relative to the robot, i.e. the forwards/backwards and sideways planes of balance, separately, so for the purposes of constructing stabilising feedback it is useful to be able to answer questions like (see Figure 5.1):

- How rotated is the robot in the **transverse** plane, i.e. what is the heading of the robot?
- How rotated is the robot in the **sagittal** plane?
- How rotated is the robot in the **lateral** plane?

We are searching for a rotation representation that, amongst other things, can provide good answers to these questions. These are not the only requirements that we are looking for however, to find the 'ideal' rotation representation for the analysis and control of balancing bodies in 3D. Our further requirements and expectations are outlined in the remainder of this section (as well as in the one that follows).

As illustrated in Figure 5.2, given any orientation of the robot, we wish to be able to answer the three questions above with three *scalar angular values* that we can write in the three figurative boxes shown.

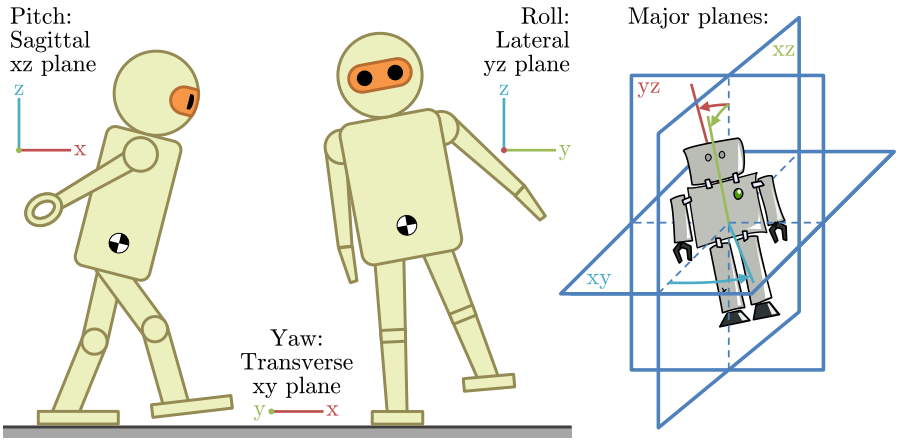


Figure 5.1: Illustration of the three major planes of balance, the sagittal **xz** plane (pitch), lateral **yz** plane (roll), and transverse **xy** plane (yaw). For the purpose of being able to analyse and control the state of balance of the robot, we wish to be able to quantify the 'amount of rotation' a robot has individually within each of these three planes, as suggested by the arrows on the right-hand image.

We will refer to these values as 'yaw', 'pitch' and 'roll' respectively, and together, these three values should define the entire orientation. By definition, the three values should quantify the 'amount of rotation', whatever that is chosen to mean, within the **xy**, **xz** and **yz** major planes (see Figure 5.1).[1] For ease of interpretation, we require the values to just be angles, i.e. in units of radians, and loosely speaking we should expect that bigger values lead to bigger rotations. As a final requirement (for now), in order to be intuitive and geometrically useful, the sole effect of applying a global z-rotation to the orientation of the robot should be to additively affect the yaw value. That is, if the robot is rotated by $\frac{\pi}{4}$ rad about the global positive z-axis, then only the yaw value should change, and up to angle wrapping, should change by exactly $+\frac{\pi}{4}$ rad. This property is referred to as yaw additivity.

The requirements listed so far are all relatively elemental and understandable, but in fact, other than Euler angles, no standard existing rotation representation is in a position to be able to satisfy these requirements, and to provide three values to write into the three boxes. The list of existing rotation representations is introduced in detail in the following section (see Section 5.2), but as an example, quaternions and rotation matrices clearly fail to provide any scalar angular values at all, let alone ones that directly quantify the amount of rotation in each of the three major planes, and rotation vectors in particular fail the required additivity property of yaw. Euler angles may at first *seem* to be a satisfactory solution to the given requirements, being a commonly accepted catch-all solution, but this is not entirely so. They can often enough be the correct choice for a task, such as for

1 This can approximately be thought of as the 'amount of rotation' about the corresponding z, y and x-axes respectively, but not quite.

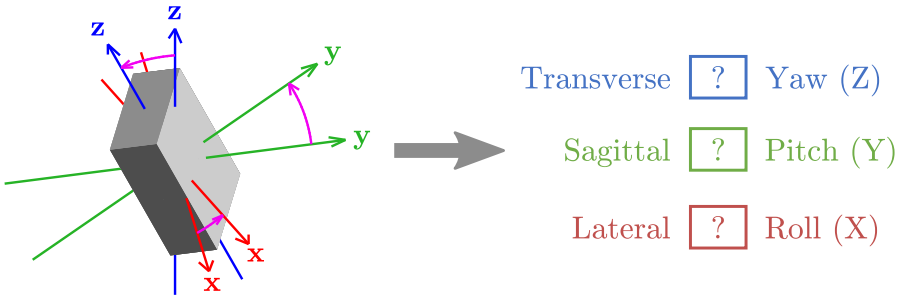


Figure 5.2: Given any 3D orientation, represented here on the left by the rotation of a rectangular prism, we wish to be able to write three scalar angular values in the three figurative boxes shown, where together the three values unambiguously define the entire orientation. The three values should relate to the 'amounts of rotation' within the three major planes, which can loosely be thought of as the amounts of rotation about the x, y and z-axes, and will be referred to as the 'roll', 'pitch' and 'yaw' of the rotation respectively.

example the modelling of gimbals or a colocated series of joints, but too often they are chosen simply because there does not *seem* to be a reasonable alternative. This thesis provides two such alternatives and dedicates an entire chapter (see Chapter 6) to an in-depth explanation of the shortcomings of Euler angles, and why they are unsuitable for balance-related tasks. In fact, other than for specific situations like just previously mentioned, where the task itself inherently and physically involves a series of well-defined elemental rotations, it is unclear what reason there would be to ever use Euler angles at all over a construct like the tilt phase space (introduced in this chapter). The main problems with Euler angles revolve around the location of their singularities, the lack of parameter axisymmetry, and the lack of complete parameter independence. All of these points are discussed in detail in Chapter 6, with an adjoining discussion of why the representations introduced in this chapter do not have any such problems.

5.1.2 Further Aims: Partitioning of Rotations into Yaw and Tilt

Three new rotation representations are introduced in this chapter, namely the tilt angles, fused angles and tilt phase space parameterisations. While tilt angles form an intermediary representation that intuitively demonstrates how to partition rotations into meaningful notions of *yaw* and *tilt*, namely the fused yaw and tilt rotation components, the other two representations follow suit, but reparameterise the tilt component to provide *pitch* and *roll* components, as required by the motivational scenario of the three boxes describing the amount of rotation in the three major planes (see Figures 5.1 and 5.2). The fused angles and tilt phase space parameterisations are easy ways

of quantifying 3D rotations, just like Euler angles are, but ways that have been specifically developed for balance-related scenarios and mobile robotics, and solve numerous problems and imperfections that Euler angles have. They describe the state of balance of a robot in an intuitive and problem-relevant way, and in particular the fused angles parameterisation offers a useful geometric interpretation as well. Most importantly however, both new parameterisations use *concurrent* specifications of pitch and roll, in that no order of rotations is imposed on the two like for Euler angles. That is, the tilt component in each case is a single atomic rotation that is quantified by the two scalar angular pitch and roll values, but no subdivision into separate pitch and/or roll rotations can logically be made. This is important as it enables the definitions of pitch and roll to be axisymmetric, a point that is discussed in more detail in Section 6.2.4.

Ultimately, when using rotation parameterisations to represent the orientation of a robot for the purpose of a balance-related task such as a bipedal gait, the greatest concern is how close the robot is to falling over. This can also be characterised as how far the robot is from being upright, so naturally the chosen way of partitioning rotations into yaw and tilt components should reflect that. Specifically, the yaw component should encapsulate the heading of the robot, i.e. the horizontal planar 360° bearing of the robot that it is deemed to be facing, and the tilt component should purely encapsulate the remaining heading-independent relation between the robot and the planar ground, i.e. how far the robot is from being upright no matter what direction it is facing. This is why the tilt component is sometimes referred to as a heading-independent balance state, as it should indicate what direction locally relative to the robot's heading the robot is falling in, if any.

How far a robot is from being upright can equally be seen as a measurement of how far the gravity vector is away from pointing straight down relative to the robot's body-fixed coordinate frame {B}. In fact, the direction of this gravity vector (i.e. $-{}^{B}\mathbf{z}_{G}$) is exactly what an accelerometer attached to the robot would measure under quasi-static conditions, and it can be seen that this vector is completely independent to any changes in heading, i.e. any applications of global z-rotations to {B}. This leads to the further aim that in order to suitably represent the heading-independent balance state, the tilt component of a rotation should be defined in such a way that there is a one-to-one correspondence in general to the set of possible measured gravity directions, and therefore also to the unit sphere of possible ${}^{B}\mathbf{z}_{G}$.

Rotations in general can always be viewed as a single rotation by an angle θ_a about a unit vector axis $\hat{\mathbf{e}} = (e_x, e_y, e_z)$ (see Section 5.2.2). This characterisation of rotations can also be used to set expectations for how the definitions of the yaw and tilt components should behave. Specifically, the z-component of $\hat{\mathbf{e}}$, namely e_z, encodes a measure of the

proportion of the rotation that is about the z-axis, where we recall that by convention the z-axis points 'upwards'. Thus, we should expect that the tilt component of a rotation has no e_z component, and that conversely, the yaw component of a rotation is purely a function of θ_a and e_z, and completely independent of e_x and e_y—the proportions of the rotation about the x and y-axes.

The method of partitioning 3D rotations into fused yaw and tilt rotation components, as presented in this chapter, satisfies all of the aforementioned expectations and requirements.

5.2 EXISTING ROTATION REPRESENTATIONS

Numerous ways of representing a rotation in three-dimensional Euclidean space have been developed and refined over the years. Many of these representations arose naturally from classical mathematics, and have found widespread use in areas such as physics, engineering and robotics. Different representations have different advantages and disadvantages, and which representation is suitable for a particular application depends on a wide range of considerations. Such considerations include:

- Ease of geometric interpretation, in particular in a form that is relevant to the particular problem,
- The range of singularity-free behaviour,
- Computational efficiency in terms of common operations such as rotation composition and vector rotation,
- Mathematical convenience, in terms of numeric and algebraic complexity and manipulability, and
- Algorithmic convenience, in the sense of a representation potentially possessing properties that can conveniently be exploited for a particular algorithm.

A wide range of existing rotation representations are reviewed in this section as a basis of comparison to the ones newly developed in this chapter. Due to the dimensionality of the space of 3D rotations, a minimum of three parameters is required for any such representation. A representation with exactly three parameters is referred to as minimal, while other representations with a greater number of parameters are referred to as redundant.

5.2.1 Rotation Matrices

A rotation can be represented as a linear transformation of coordinate frame basis vectors, expressed in the form of an orthogonal matrix of

unit determinant. Due to the strong link between such transformation matrices and the theory of direction cosines, the name direction cosine matrix is also sometimes used. The space of all rotation matrices is called the special orthogonal group $\mathrm{SO}(3)$, and is defined as

$$\mathrm{SO}(3) = \{R \in \mathbb{R}^{3\times 3} : R^T R = \mathbb{I},\ \det(R) = 1\}, \tag{5.1}$$

where $\mathbb{R}^{3\times 3}$ is the set of all 3×3 matrices with real entries. It is important to note that the special orthogonal group $\mathrm{SO}(3)$ has an exact one-to-one correspondence with the space of all 3D rotations, so it is often interchangeably used to denote it.

We first note from Equation (5.1) that all rotation matrices have the property that

$$R^T = R^{-1}. \tag{5.2}$$

Rotation of a vector $\mathbf{v} \in \mathbb{R}^3$ by a rotation matrix is given by matrix premultiplication. Furthermore, for a rotation from a coordinate frame {G} to another frame {B}, we have that

$$\begin{matrix}G\\B\end{matrix}R = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ {}^G\mathbf{x}_B & {}^G\mathbf{y}_B & {}^G\mathbf{z}_B \\ \downarrow & \downarrow & \downarrow \end{bmatrix} = \begin{bmatrix} \leftarrow & {}^B\mathbf{x}_G & \rightarrow \\ \leftarrow & {}^B\mathbf{y}_G & \rightarrow \\ \leftarrow & {}^B\mathbf{z}_G & \rightarrow \end{bmatrix}, \tag{5.3}$$

where ${}^G\mathbf{y}_B$, for example, is the column vector corresponding to the y-axis of frame {B} expressed in the coordinates of frame {G}. With nine parameters, rotation matrices are clearly a redundant parameterisation of the rotation space. They are quite useful in that they are free of singularities, easy to compose, and trivially expose the basis vectors of the fixed and rotated frames, but for many tasks they are not as computationally and/or numerically suitable as other representations. One problem in particular, is that the numerical reorthogonalisation of a nearly valid rotation matrix is a non-trivial and involved process.

5.2.2 Axis-angle and Rotation Vector Representations

By Euler's rotation theorem, every rotation in three-dimensional Euclidean space can be expressed as a single counterclockwise (CCW) rotation by up to π radians about some axis. As such, every 3D rotation can be assigned a vector-scalar axis-angle pair

$$\begin{aligned} A &= (\hat{\mathbf{e}}, \theta_a) \\ &\in \mathcal{S}^2 \times [0, \pi] \equiv \mathbb{A}, \end{aligned} \tag{5.4}$$

where $\hat{\mathbf{e}} = (e_x, e_y, e_z)$ is a unit vector corresponding to the axis of rotation, θ_a is the angular magnitude of the rotation, and $\mathcal{S}^2$ is the unit sphere in $\mathbb{R}^3$. The domain $\mathbb{A}$ is the set of all axis-angle pairs, where it should be noted that certain pairs correspond to the same final coordinate frame, like for example $(\hat{\mathbf{e}}, \pi)$ and $(-\hat{\mathbf{e}}, \pi)$. By convention,

for numerical and/or algorithmic benefits, the identity rotation is often chosen to be represented by $\hat{\mathbf{e}} = \mathbf{0}$, even though strictly speaking $\mathbf{0} \notin \mathcal{S}^2$. If we consider values of θ_a outside of the default $[0, \pi]$ interval, the following representational equivalences hold:

$$(\hat{\mathbf{e}}, \theta_a) \equiv (\hat{\mathbf{e}}, \theta_a + 2\pi k), \tag{5.5a}$$

$$(\hat{\mathbf{e}}, \theta_a) \equiv (-\hat{\mathbf{e}}, -\theta_a), \tag{5.5b}$$

for $k \in \mathbb{Z}$ an integer. Incidentally, these equivalences demonstrate how any $(\hat{\mathbf{e}}, \theta_a) \in \mathcal{S}^2 \times \mathbb{R}$ can be collapsed into an equivalent expression in the default domain $\mathbb{A} = \mathcal{S}^2 \times [0, \pi]$. A closely related concept to the axis-angle representation is the rotation vector, given by

$$\mathbf{e} = \theta_a \hat{\mathbf{e}} \in \mathbb{R}^3. \tag{5.6}$$

Effectively, rotation vectors are 3D vectors that encode the magnitude and axis of a 3D rotation in terms of their vector norm and direction.

While the axis-angle representation is redundant, the rotation vector representation, with only three parameters, is classified as minimal. Both representations however suffer from a general impracticality of mathematical and numerical manipulation. For example, no formula for rotation composition exists that is more direct than converting to quaternions and back. Rotation vectors do indeed provide three different angular values that quantify different dimensions of the rotation, but they do not intuitively or otherwise well-define the amount of rotation in the three major planes, and the z-component does not satisfy the necessary additivity condition (see Section 5.1.1) that is required for it to constitute a usable concept of yaw.

5.2.3 Quaternions

Quaternions in general are an extension of the complex numbers to four dimensions. While complex numbers define the imaginary number i such that $i^2 = -1$, and describe every possible complex element as a combination $a + bi$, quaternions define three such numbers i, j and k such that

$$i^2 = j^2 = k^2 = ijk = -1, \tag{5.7}$$

and describe every possible quaternion as a combination[2]

$$q = a + bi + cj + dk. \tag{5.8}$$

While this is the classic definition of general quaternions, it is more common and useful in the context of 3D rotations to view quaternions

2 Note that as a result of Equation (5.7), one can deduce that $ij = k$, $jk = i$ and $ki = j$. Thus, using these basic rules, products of quaternions can always be simplified down again to the canonical form given in Equation (5.8).

just as their four real coefficients (a, b, c, d). With a suggestive switch in coefficient labels, the set of all general quaternions becomes

$$\mathbb{H} = \{q = (w, x, y, z) \in \mathbb{R}^4\}. \tag{5.9}$$

The relation to the classic definition of quaternions is then given by

$$1 \equiv (1, 0, 0, 0) \in \mathbb{H}, \tag{5.10a}$$
$$i \equiv (0, 1, 0, 0) \in \mathbb{H}, \tag{5.10b}$$
$$j \equiv (0, 0, 1, 0) \in \mathbb{H}, \tag{5.10c}$$
$$k \equiv (0, 0, 0, 1) \in \mathbb{H}. \tag{5.10d}$$

For $q = (w, x, y, z)$, the w-component is referred to as the scalar part of q, and the remaining three components are referred to as the vector part of q. This leads to the vector notation of a quaternion q, namely

$$q = (w, x, y, z) = (q_0, \mathbf{q}), \tag{5.11}$$

where

$$q_0 = w \in \mathbb{R}, \tag{5.12a}$$
$$\mathbf{q} = (x, y, z) \in \mathbb{R}^3. \tag{5.12b}$$

In fact, all scalars $a \in \mathbb{R}$ and vectors $\mathbf{v} \in \mathbb{R}^3$ can be identified with their corresponding purely scalar or vector quaternions, given by

$$a \equiv (a, 0, 0, 0) \in \mathbb{H}, \tag{5.13a}$$
$$\mathbf{v} \equiv (0, v_x, v_y, v_z) \in \mathbb{H}. \tag{5.13b}$$

Pure rotations in 3D space can be represented by the set of quaternions of unit norm. The set of all quaternion rotations is thus

$$\mathbb{Q} = \{q \in \mathbb{H} : \|q\| = 1\}, \tag{5.14}$$

where we note in particular that for $q \in \mathbb{Q}$, this implies that

$$w^2 + x^2 + y^2 + z^2 = 1. \tag{5.15}$$

The four quaternion rotation parameters are sometimes also referred to as Euler-Rodrigues parameters. The space $\mathbb{Q}$ corresponds to the four-dimensional unit sphere $\mathcal{S}^3$, and is a double cover of the space of rotations $\mathrm{SO}(3)$, in that q and $-q$ both correspond to the same 3D rotation. That is,

$$q \equiv -q. \tag{5.16}$$

Quaternion rotations with $w \geq 0$ can be related to their corresponding axis-angle representation $(\hat{\mathbf{e}}, \theta_a) \in \mathbb{A}$, and thereby visualised to some degree in terms of what rotation they represent, using

$$q = (q_0, \mathbf{q}) = \left(\cos\tfrac{\theta_a}{2}, \hat{\mathbf{e}} \sin\tfrac{\theta_a}{2}\right), \tag{5.17a}$$
$$= \left(\cos\tfrac{\theta_a}{2}, e_x \sin\tfrac{\theta_a}{2}, e_y \sin\tfrac{\theta_a}{2}, e_z \sin\tfrac{\theta_a}{2}\right) \in \mathbb{Q}. \tag{5.17b}$$

The use of quaternions to express rotations generally allows for very computationally efficient calculations, and is grounded by the well-established field of quaternion mathematics. A crucial advantage of the quaternion representation is that it is free of singularities. On the other hand, as previously mentioned, it is not a one-to-one mapping of the special orthogonal group, as q and $-q$ both correspond to the same rotation. The redundancy of the parameterisation, as there are four parameters, also means that the unit magnitude constraint has to explicitly and sometimes non-trivially be enforced in numerical computations. Furthermore, no clear geometric interpretation of quaternions exists beyond the implicit relation to the axis-angle representation given in Equation (5.17). In light of the motivation and aims set out in Section 5.1, for applications relating to the analysis and control of balancing bodies in 3D, quaternions can be very helpful in performing state estimation (e.g. Chapter 7), but a final quaternion orientation of the trunk does not give any direct insight into the amount of rotation within the three major planes. As such, quaternions are very useful computationally, but do not by themselves provide the answers we are looking for.

5.2.4 Euler Angles

Instead of representing rotations as a single turn about a single axis, like for the axis-angle representation, it is also possible to extract more meaning by expressing rotations as a sequence of three rotations about three well-defined axes. Euler angles express a rotation as such a sequence of three elemental rotations, and rotate a coordinate frame about a predefined set of *coordinate axes* in a predefined order. The elemental rotations are either by convention extrinsic about the fixed global x, y and z-axes, or intrinsic about the local x, y and z-axes of the coordinate frame being rotated. For example, if the first elemental rotation has already been applied to the global frame {G} to get the intermediate frame {A}, then the next elemental rotation about, for instance, the y-axis, would be about $\mathbf{y}_G$ for extrinsic Euler angles, or $\mathbf{y}_A$ for intrinsic Euler angles (which is the more common case). Independently of whether the extrinsic or intrinsic convention is chosen, there are six possible choices for the predefined order of coordinate axis rotations where each axis is used only once (e.g. YZX), and a further six possible conventions where the first and third axes of rotation are the same (e.g. ZXZ). The former are commonly referred to as Tait-Bryan angles, and the latter are referred to as proper Euler angles. All possible Euler angles axis conventions are summarised in Table 5.1.

It is easy to see that all extrinsic Euler angles conventions are completely equivalent to the corresponding intrinsic Euler angles conventions, just with the order of axis rotations reversed. Whether

Table 5.1: A complete list of the possible Euler angles axis conventions, separated into Tait-Bryan angles and proper Euler angles. A convention of YZX, for example, means that the first elemental rotation is about the y-axis, followed by the z-axis, and then the x-axis. Whether the axes from the local (rotating) or global (fixed) frame are used depends on a further choice of whether the Euler angles are intrinsic or extrinsic, respectively.

Type	Order of axis rotations
Tait-Bryan angles	XYZ, XZY, YXZ, YZX, ZXY, ZYX
Proper Euler angles	XYX, XZX, YXY, YZY, ZXZ, ZYZ

the rotations are local or global just reverses the order in which the rotations are 'stacked'. Thus, without loss of generality, we only consider the intrinsic convention from here on in. We wish to evaluate Euler angles with respect to the aims set out in Section 5.1, and compare them to the new parameterisations that are developed in this thesis. As such, it is furthermore desirable to only consider axis conventions that have their three elemental rotations about all three different axes, i.e. Tait-Bryan angles, so that the amount of rotation within each of the three major planes can be quantified.

One of the aims that was specified in Section 5.1.1 was the property of yaw additivity. In the context of Euler angles, this means that if a global z-rotation is applied to a frame, then only the Euler angle corresponding to the z-axis should change, and up to angle wrapping, it should change by the exact amount of the z-rotation. This is clearly the case if and only if the axis convention places the z-rotation first. Thus, only two viable Euler angles conventions remain as candidates for analysis, the intrinsic ZYX and intrinsic ZXY Euler angles conventions. For completeness, both of these Euler angles conventions are presented in the next two sections, but unless explicitly otherwise stated, all further references to 'Euler angles' will be referring to intrinsic ZYX Euler angles. All arguments and properties that apply to the ZYX representation can however equivalently be reformulated to apply to the ZXY representation, so the choice is arbitrary.

The z, y and x-components of the Euler axis representation collectively correspond to the concepts of yaw, pitch and roll, respectively. Although initially promising, Euler angles do not suffice for the representation of the orientation of a body in balance-related scenarios. The main reasons for this are:

- The proximity of the gimbal lock singularities to normal working ranges, leading to unwanted artefacts due to the increased local parameter sensitivities in widened neighbourhoods of the singularities,

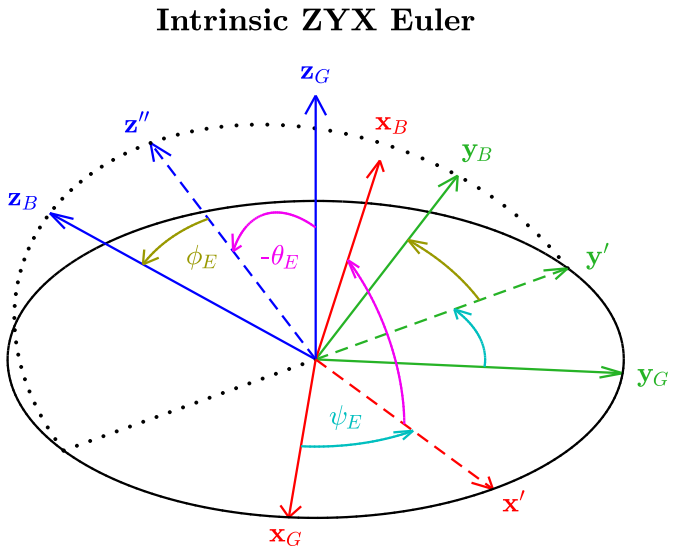


Figure 5.3: Definition of the intrinsic ZYX Euler angles convention for a rotation from {G} to {B}. The global frame {G} is first rotated CCW by the Euler yaw ψ_E about $\mathbf{z}_G$, then by the Euler pitch θ_E about $\mathbf{y}'$, and finally by the Euler roll ϕ_E about $\mathbf{x}'' \equiv \mathbf{x}_B$. The pitch rotation is labelled $-\theta_E$ because geometrically the labelled arc in this example is clockwise, and not CCW.

- The mutual dependence of the parameters, leading to a mixed attribution of which parameters contribute to which major planes of rotation,
- The fundamental requirement of an order of elemental rotations, leading to non-axisymmetric definitions of pitch and roll that do not correspond to each other in behaviour, and
- The asymmetry introduced by the use of a yaw definition that depends on the projection of one of the coordinate axes onto a fixed plane, leading to unintuitive non-axisymmetric behaviour of the yaw angle.

More details on all of these points are provided in Chapter 6.

5.2.4.1 *Intrinsic ZYX Euler Angles*

Let {G} denote the global reference frame, and let {B} be the body-fixed frame of which the orientation is being expressed. The intrinsic ZYX Euler angles representation consists of the following sequence of three rotations, as shown in Figure 5.3:

- First a rotation by the Euler yaw ψ_E about the original z-axis, which corresponds to $\mathbf{z}_G$,
- Then a rotation by the Euler pitch θ_E about the resulting intermediate y-axis, and

- Finally a rotation by the Euler roll ϕ_E about the once again resulting x-axis, which corresponds to $\mathbf{x}_B$.

The complete Euler angles rotation from {G} to {B} is then denoted by

$$\begin{aligned} {}^G_B E &= (\psi_E, \theta_E, \phi_E) \\ &\in (-\pi, \pi] \times [-\tfrac{\pi}{2}, \tfrac{\pi}{2}] \times (-\pi, \pi] \equiv \mathbb{E}. \end{aligned} \tag{5.18}$$

The representation is unique, except at the gimbal lock singularities, which correspond to $\theta_E = \pm\frac{\pi}{2}$, where for $\epsilon \in \mathbb{R}$ we have that

$$(\psi_E, \tfrac{\pi}{2}, \phi_E) \equiv (\psi_E - \epsilon, \tfrac{\pi}{2}, \phi_E - \epsilon), \tag{5.19a}$$
$$(\psi_E, -\tfrac{\pi}{2}, \phi_E) \equiv (\psi_E - \epsilon, -\tfrac{\pi}{2}, \phi_E + \epsilon). \tag{5.19b}$$

Beyond wrapping of the individual angles to the standard range of $(-\pi, \pi]$, one further representational equivalence is given by

$$(\psi_E, \theta_E, \phi_E) \equiv (\pi + \psi_E, \pi - \theta_E, \pi + \phi_E). \tag{5.20}$$

The rotation matrix R corresponding to the Euler angles rotation $E = (\psi_E, \theta_E, \phi_E)$ is given by

$$R = R_z(\psi_E) R_y(\theta_E) R_x(\phi_E) \tag{5.21a}$$
$$= \begin{bmatrix} c_{\psi_E} c_{\theta_E} & c_{\psi_E} s_{\theta_E} s_{\phi_E} - s_{\psi_E} c_{\phi_E} & c_{\psi_E} s_{\theta_E} c_{\phi_E} + s_{\psi_E} s_{\phi_E} \\ s_{\psi_E} c_{\theta_E} & s_{\psi_E} s_{\theta_E} s_{\phi_E} + c_{\psi_E} c_{\phi_E} & s_{\psi_E} s_{\theta_E} c_{\phi_E} - c_{\psi_E} s_{\phi_E} \\ -s_{\theta_E} & c_{\theta_E} s_{\phi_E} & c_{\theta_E} c_{\phi_E} \end{bmatrix}, \tag{5.21b}$$

where $R_y(\cdot)$ for example is the rotation matrix corresponding to a CCW rotation by angle $\cdot$ about the y-axis, and we denote $s_* \equiv \sin(*)$ and $c_* \equiv \cos(*)$. The reverse conversion from a rotation matrix R back to the Euler angles representation E is given by

$$\psi_E = \operatorname{atan2}(R_{21}, R_{11}), \tag{5.22a}$$
$$\theta_E = \operatorname{asin}(-R_{31}), \tag{5.22b}$$
$$\phi_E = \operatorname{atan2}(R_{32}, R_{33}), \tag{5.22c}$$

where R_{ij} is the i^{th}-row j^{th}-column entry of the rotation matrix R. The conversion from Euler angles to the quaternion representation $q = (w, x, y, z)$ is given by

$$q = q_z(\psi_E)\, q_y(\theta_E)\, q_x(\phi_E) \tag{5.23a}$$
$$= \begin{pmatrix} c_{\frac{\psi_E}{2}} c_{\frac{\theta_E}{2}} c_{\frac{\phi_E}{2}} + s_{\frac{\psi_E}{2}} s_{\frac{\theta_E}{2}} s_{\frac{\phi_E}{2}}, & c_{\frac{\psi_E}{2}} c_{\frac{\theta_E}{2}} s_{\frac{\phi_E}{2}} - s_{\frac{\psi_E}{2}} s_{\frac{\theta_E}{2}} c_{\frac{\phi_E}{2}}, \\ c_{\frac{\psi_E}{2}} s_{\frac{\theta_E}{2}} c_{\frac{\phi_E}{2}} + s_{\frac{\psi_E}{2}} c_{\frac{\theta_E}{2}} s_{\frac{\phi_E}{2}}, & s_{\frac{\psi_E}{2}} c_{\frac{\theta_E}{2}} c_{\frac{\phi_E}{2}} - c_{\frac{\psi_E}{2}} s_{\frac{\theta_E}{2}} s_{\frac{\phi_E}{2}} \end{pmatrix}, \tag{5.23b}$$

where $q_z(\cdot)$ for example is the quaternion corresponding to a CCW rotation by angle $\cdot$ about the z-axis, and we denote $\bar{*} \equiv \frac{1}{2}*$. The conversion from $q = (w, x, y, z)$ back to $E = (\psi_E, \theta_E, \phi_E)$ is given by

$$\psi_E = \text{atan2}(wz + xy, \tfrac{1}{2} - y^2 - z^2), \tag{5.24a}$$
$$\theta_E = \text{asin}(2(wy - xz)), \tag{5.24b}$$
$$\phi_E = \text{atan2}(wx + yz, \tfrac{1}{2} - x^2 - y^2). \tag{5.24c}$$

The conversion equations presented here are required later for analysis of the Euler angles representation, in particular for Chapter 6.

5.2.4.2 *Intrinsic ZXY Euler Angles*

The intrinsic ZXY Euler angles representation consists of the following sequence of three rotations:

- First a rotation by the ZXY Euler yaw $\psi_{\tilde{E}}$ about the original z-axis, which corresponds to $\mathbf{z}_G$,
- Then a rotation by the ZXY Euler roll $\phi_{\tilde{E}}$ about the resulting intermediate x-axis, and
- Finally a rotation by the ZXY Euler pitch $\theta_{\tilde{E}}$ about the once again resulting y-axis, which corresponds to $\mathbf{y}_B$.

The complete ZXY Euler angles rotation is then denoted by

$$\begin{aligned} {}^G_B\tilde{E} &= (\psi_{\tilde{E}}, \phi_{\tilde{E}}, \theta_{\tilde{E}}) \\ &\in (-\pi, \pi] \times [-\tfrac{\pi}{2}, \tfrac{\pi}{2}] \times (-\pi, \pi] \equiv \tilde{\mathbb{E}}. \end{aligned} \tag{5.25}$$

5.3 PARTITIONING ROTATIONS INTO YAW AND TILT

As thoroughly explained in Section 5.1, we wish to find a way of partitioning any 3D rotation into separate and independent *yaw* and *tilt* components, which together uniquely define the original rotation. We wish for this partition to be intuitive and meaningful, and further wish to parameterise the tilt component by equally meaningful angular notions of *pitch* and *roll*. This section addresses how to do exactly this, and formally introduces the *tilt angles*, *fused angles* and *tilt phase space* rotation parameterisations.

5.3.1 Fused Yaw

We begin by defining our notion of yaw, the fused yaw of a rotation. Suppose we have the global frame {G}, and any body-fixed frame {B}, as illustrated in Figure 5.4. The rotation that we are seeking to parameterise is the one from {G} to {B}. We first define the intermediate frame {A} by rotating {B} in such a way that $\mathbf{z}_B$ rotates onto $\mathbf{z}_G$ in the

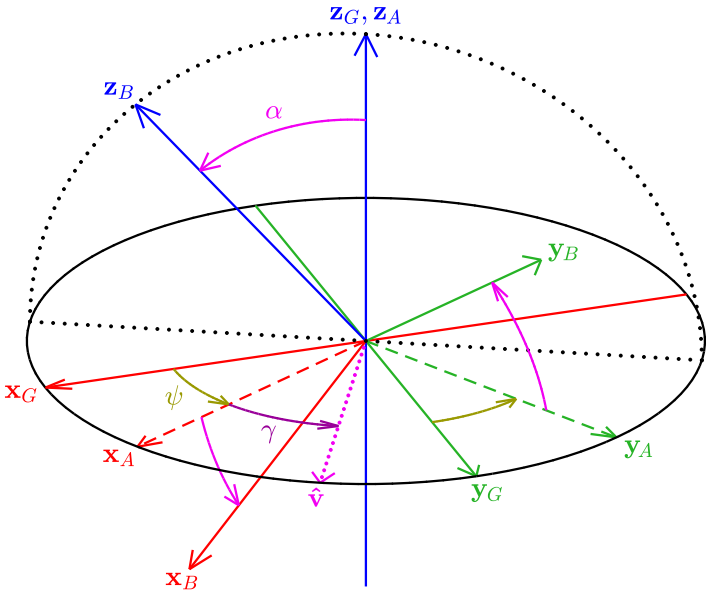


Figure 5.4: Definition of the tilt angles parameters $T = (\psi, \gamma, \alpha)$ for a rotation from {G} to {B}. The frame {G} is first rotated CCW about $\mathbf{z}_G$ by the fused yaw ψ to give the intermediate frame {A}, and this frame is then tilted CCW about $\hat{\mathbf{v}}$ by the tilt angle α to give {B}. The tilt axis $\hat{\mathbf{v}}$ is always in the global $\mathbf{x}_G\mathbf{y}_G$ plane, and is at an angle of γ relative to $\mathbf{x}_A$, as indicated. For aid of visualisation, note that $\mathbf{x}_B$ is pointing left-downwards out of the page, and $\mathbf{y}_B$ is pointing up-rightwards out of the page.

most direct way possible, within the plane through the origin that contains these two vectors. Note that this rotation of {B} to get {A} is in the opposite direction to the magenta arrows in Figure 5.4 (see in particular the arrow labelled 'α'). The fused yaw $\psi \in (-\pi, \pi]$ is then given by the signed angle of the pure CCW z-rotation from {G} to {A}, as labelled in the figure. If $\mathbf{z}_B$ and $\mathbf{z}_G$ point in exactly opposite directions, then {A} is not uniquely defined, and this situation is referred to as the fused yaw singularity. This corresponds to all situations where the rotation from {B} to {A} has a magnitude of π radians, i.e. 180°.

Mathematically, one can deduce that the fused yaw is given, in terms of the basic coordinate axis components of {G} and {B}, by

$$\psi = \text{wrap}\big(2\,\text{atan2}\big({}^{G}x_{By} - {}^{G}y_{Bx}, 1 + {}^{G}x_{Bx} + {}^{G}y_{By} + {}^{G}z_{Bz}\big)\big), \quad (5.26)$$

where wrap$(\cdot)$ is a function that wraps an angle to the range $(-\pi, \pi]$ by multiples of 2π. Note however that this equation is not numerically robust, and in fact fails for rotations that have an angle of exactly $\theta_a = \pi$, i.e. rotations by 180° about any axis, as atan2$(0, 0)$ emerges on the right-hand side. As a robust alternative, if ${}^{G}_{B}q = (w, x, y, z) \in \mathbb{Q}$ is the quaternion rotation from {G} to {B}, the nominal mathematical definition of the fused yaw is given by

$$\psi = \text{wrap}\big(2\,\text{atan2}(z, w)\big). \quad (5.27)$$

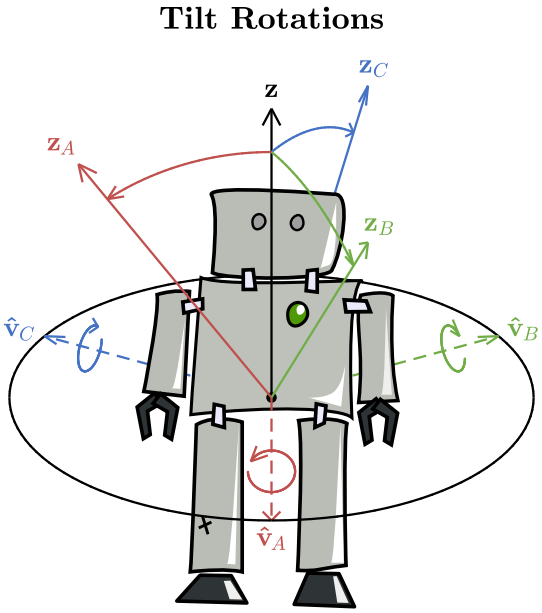


Figure 5.5: Three different examples of tilt rotations applied to an upright standing robot. The corresponding axes of rotation in the horizontal **xy** plane, $\hat{\mathbf{v}}_*$, are shown in dashed. Note that the direction of rotation is always CCW.

This is the simplest and most fundamental definition of the fused yaw, and fails with $\text{atan2}(0,0)$ if and only if $w = z = 0$, which corresponds exactly to the fused yaw singularity described above. For convenience, at the fused yaw singularity the fused yaw ψ is generally defined to be zero, as this makes definitions such as for example Equation (5.28) consistent with other characterisations.

5.3.2 Tilt Rotations

In reference to Figure 5.4, the fused yaw ψ parameterises the first component of the rotation from {G} to {B}, namely the rotation from {G} to {A}, but the rotation component from {A} to {B} still remains. This component is referred to as the tilt rotation component of {B} relative to {G}, and it is easy to see that this tilt component itself has a fused yaw of zero. In fact, in general all rotations with zero fused yaw are referred to exclusively as tilt rotations, or pure tilt rotations, and the set of all tilt rotations is denoted by

$$\text{TR}(3) = \{R \in \text{SO}(3) : \psi = 0\} \subset \text{SO}(3). \tag{5.28}$$

While the fused yaw of a rotation relates to the change in heading that it induces, the tilt rotation component of a rotation relates to how much, and in what direction, the originally upright z-axis 'tips over' towards the horizontal ground plane. Three different examples

of tilt rotations, as applied to an upright standing robot, are shown in Figure 5.5 for illustrative purposes of this concept.

The following are completely equivalent characterisations of the set of all (pure) tilt rotations:

- All rotations that have a fused yaw of zero,
- All rotations that correspond to a single rotation about a vector in the horizontal **xy** plane,
- All rotations for which the axis of rotation has no component in the z-direction, i.e. $e_z = 0$, and,
- All rotations that have a quaternion z-component of zero.

Most of these characterisations follow naturally and easily from the definition of fused yaw in Section 5.3.1. The last one can be observed from Equation (5.27), by substituting in $z = 0$ and considering the cases separately where w is positive or negative.

5.3.2.1 *Z-vector Parameterisation*

Given the above definition and characterisations of tilt rotations, we now wish to be able to parameterise the two-dimensional tilt rotation component of a rotation, i.e. the component from {A} to {B}, just like the parameter ψ was used to parameterise the one-dimensional yaw rotation component from {G} to {A}. The space of all tilt rotations is, as stated, two-dimensional, so two or more parameters are required for the task, or exactly two in order to be minimal. While three novel such parameterisations are introduced in this chapter, two of which specifically aim to do this in a way that reflects the angular amounts of rotation independently in the sagittal and lateral planes, we can already preliminarily parameterise tilt rotations using existing tools that we already have, namely rotation matrices and quaternions.

By definition, the tilt rotation component relates to the relative directions of the two z-axes $\mathbf{z}_G$ and $\mathbf{z}_B$, and the nature of the direct planar 3D rotation between them (see Figure 5.4). Thus, it can be observed that two immediate options for parameterising tilt rotations are just these z-axis vectors themselves, namely the global z-vector ${}^B\mathbf{z}_G$, and the local z-vector ${}^G\mathbf{z}_B$. In terms of the basic coordinate axis components of {G} and {B}, these can trivially be identified as

$$ {}^B\mathbf{z}_G = ({}^Bz_{Gx}, {}^Bz_{Gy}, {}^Bz_{Gz}), \tag{5.29a} $$

$$ {}^G\mathbf{z}_B = ({}^Gz_{Bx}, {}^Gz_{By}, {}^Gz_{Bz}). \tag{5.29b} $$

We note that these are unit vectors, and thus use three interdependent parameters to express the required two degrees of freedom, and that they correspond exactly to the last row and column of the rotation matrix ${}^G_BR$, respectively.

The local z-vector ${}^{G}\mathbf{z}_{B}$ is a valid parameterisation for tilt rotation components, but not in general a useful one because its x and y-components depend on the fused yaw ψ. This means that if the heading of a body {B} is changed, the fused yaw changes additively as required, but the local z-vector ${}^{G}\mathbf{z}_{B}$ also changes despite the fact that the tilt rotation component itself has not changed. The global z-vector ${}^{B}\mathbf{z}_{G}$ does not have this problem however, as

$$ {}^{B}\mathbf{z}_{G} \equiv {}^{B}\mathbf{z}_{A}, \tag{5.30} $$

and therefore remains constant as expected. Thus, by convention, when referring to the z-vector parameterisation of a rotation, or just the z-vector of a rotation, this is unambiguously taken to mean the global one, namely ${}^{B}\mathbf{z}_{G}$. The z-vector parameterisation is meaningful and relevant in application scenarios mainly because it corresponds to the direction that an accelerometer attached to the body would measure under quasi-static conditions, as indicated in Section 5.1.2.

Given any unit global z-vector ${}^{B}\mathbf{z}_{G}$, the corresponding tilt rotation is uniquely defined, as the required intermediate frame {A} can be retrieved by rotating frame {B} directly onto $\mathbf{z}_{G}$, as per the definition in Section 5.3.1. Conversely, given a tilt rotation component, ${}^{B}\mathbf{z}_{G}$ is clearly uniquely defined. As such, the z-vector parameterisation is, as required, a one-to-one mapping of the space of tilt rotations, at least everywhere except for the fused yaw singularity, where $\mathbf{z}_{G}$ and $\mathbf{z}_{B}$ point in opposite directions. All tilt rotations for which this is the case are equally assigned ${}^{B}\mathbf{z}_{G} = (0, 0, -1)$ in terms of the z-vector parameterisation.

5.3.2.2 *Quaternion Parameterisation*

Being just a special type of 3D rotation, tilt rotation components can also be parameterised by their corresponding quaternion rotation from {A} to {B}, namely

$$ {}^{A}_{B}q \equiv (w, x, y, 0), \tag{5.31} $$

where ${}^{A}_{B}q$ must be a unit quaternion, so

$$ w^2 + x^2 + y^2 = 1. \tag{5.32} $$

Note that the above w, x and y parameters are not the same as for the quaternion representation ${}^{G}_{B}q$ of the *entire* rotation from {G} to {B}. Also, it should be noted that the quaternion z-component of tilt rotations is *always* zero, hence why the z-component can be omitted in Equation (5.31). Like for the z-vector parameterisation, the quaternion parameterisation of the tilt rotation component once again corresponds to three interdependent parameters that together express the required two degrees of freedom of the space of tilt rotations. A difference to the z-vector parameterisation however, is that ${}^{A}_{B}q$ and $-{}^{A}_{B}q$ are treated

as equivalent parameter values. The various different tilt rotations at the fused yaw singularity also correspond to different values of the w, x and y parameters, as opposed to all corresponding to the same value, like for the z-vector parameterisation. This fundamental difference is discussed further in Section 5.5, also in relation to other representations and how they behave at the fused yaw singularity.

5.3.3 Tilt Angles Representation

It can be seen from the definition of fused yaw and tilt rotations that, geometrically, every rotation can be divided into a pure z-rotation, followed by a rotation about an axis in the horizontal **xy** plane. As shown in Figure 5.4, the former is the yaw rotation component, from {G} to {A}, and the latter is the tilt rotation component, from {A} to {B}, as previously defined. One natural way of parameterising the tilt rotation component is using the axis-angle representation, defined in Section 5.2.2. We choose an axis-angle pair $(\hat{\mathbf{v}}, \alpha) \in \mathbb{A}$ such that $\alpha \in [0, \pi]$, thus representing the tilt rotation component as a single CCW rotation by up to 180° about a vector $\hat{\mathbf{v}}$ in the horizontal **xy** plane. The angle α is the magnitude of the tilt rotation component, and is referred to as the tilt angle of the rotation from {G} to {B}, and the vector $\hat{\mathbf{v}}$ is referred to as the tilt axis. The CCW angle from $\mathbf{x}_A$ to $\hat{\mathbf{v}}$ about the global z-axis $\mathbf{z}_G$ is referred to as the tilt axis angle of {B}, and is denoted γ. Both $\mathbf{x}_A$ and $\hat{\mathbf{v}}$ are in the global **xy** plane, so $\mathbf{z}_G$ is perpendicular to them, and γ is well-defined. The tilt angle α, tilt axis $\hat{\mathbf{v}}$, and tilt axis angle γ are all labelled in Figure 5.4.

It is easy to see that the tilt rotation component from {A} to {B} is completely defined by the parameter pair (γ, α). Thus, together with the fused yaw ψ, we arrive at the tilt angles parameterisation of the rotation from {G} to {B}, namely

$$\begin{aligned} {}^{G}_{B}T &= (\psi, \gamma, \alpha) \\ &\in (-\pi, \pi] \times (-\pi, \pi] \times [0, \pi] \equiv \mathbb{T}. \end{aligned} \tag{5.33}$$

The tilt angles parameters corresponding to the identity rotation are given by $(0, 0, 0) \in \mathbb{T}$. It can be seen by construction that all rotations possess a tilt angles representation, but it is not always necessarily unique. Most notably, when $\alpha = 0$, the γ parameter can be arbitrary with no effect. Mathematically, the tilt axis angle γ and tilt angle α can be expressed in terms of the basic coordinate axis components of {G}, or more specifically, the components of the z-vector ${}^{B}\mathbf{z}_G$, as

$$\gamma = \operatorname{atan2}(-{}^{B}z_{Gx}, {}^{B}z_{Gy}), \tag{5.34a}$$

$$\alpha = \operatorname{acos}({}^{B}z_{Gz}). \tag{5.34b}$$

The tilt axis $\hat{\mathbf{v}}$ can be expressed in terms of γ and the fused yaw ψ, as

$${}^{G}\hat{\mathbf{v}} = \big(\cos(\psi + \gamma), \sin(\psi + \gamma), 0\big). \tag{5.35}$$

Relative to the intermediate frame {A}, or for tilt rotations, this is just

$$^{A}\hat{\mathbf{v}} = \big(\cos\gamma, \sin\gamma, 0\big). \tag{5.36}$$

While tilt angles themselves do not directly address all the requirements and aims that were set out in Section 5.1, they constitute a useful intermediate representation for defining and analysing other parameterisations of the tilt rotation component, and give a somewhat direct and 'low-level' view of the partition of rotations into yaw and tilt components. By optionally extending the domain of α to $[0, \infty)$, tilt rotations of arbitrary magnitude can also be parameterised and numerically manipulated. This, for example, is not possible with either the z-vector or quaternion parameterisations of the tilt rotation component, and can be useful for dealing with rotations that exceed 180° in magnitude, as opposed to just considering the final coordinate frame that these so-called unbounded rotations result in.

5.3.4 Fused Angles Representation

Although the tilt angles parameterisation is a helpful start in understanding tilt rotations, we wish to be able to parameterise the tilt rotation component in such a way that it reveals concurrent definitions of pitch and roll, as discussed in Sections 5.1.1 and 5.1.2. To do this, we consider just the tilt rotation component of a rotation from {G} to {B}, namely the component from {A} to {B}, as shown in Figure 5.6. We define the fused pitch θ as the signed angle between the global z-vector $\mathbf{z}_G$ ($= \mathbf{z}_A$) and the $\mathbf{y}_B\mathbf{z}_B$ plane, and the fused roll ϕ as the signed angle between $\mathbf{z}_G$ and the $\mathbf{x}_B\mathbf{z}_B$ plane, as illustrated in the figure. The signs of θ and ϕ are defined to be the same as the signs of $-{}^{B}z_{Gx}$ and ${}^{B}z_{Gy}$, respectively. In fact, the fused pitch and roll are exactly given by

$$\theta = \operatorname{asin}(-{}^{B}z_{Gx}), \tag{5.37a}$$

$$\phi = \operatorname{asin}({}^{B}z_{Gy}). \tag{5.37b}$$

Loosely speaking, rotations with positive fused pitch can be thought of as tendentially 'forwards' rotations that have a positive CCW component about the y-axis, and positive fused roll rotations can be thought of as tendentially 'rightwards' rotations that have a positive CCW component about the x-axis.

By inspection of their mathematical and geometric definitions, it can be seen that the fused pitch and roll only uniquely specify the tilt rotation component of a rotation up to the z-hemisphere, that is, whether $\mathbf{z}_B$ and $\mathbf{z}_G$ are mutually in the same unit hemisphere or not. To resolve this ambiguity, the fused hemisphere, or just hemisphere, of a rotation is defined as

$$h = \operatorname{sign}({}^{B}z_{Gz}), \tag{5.38}$$

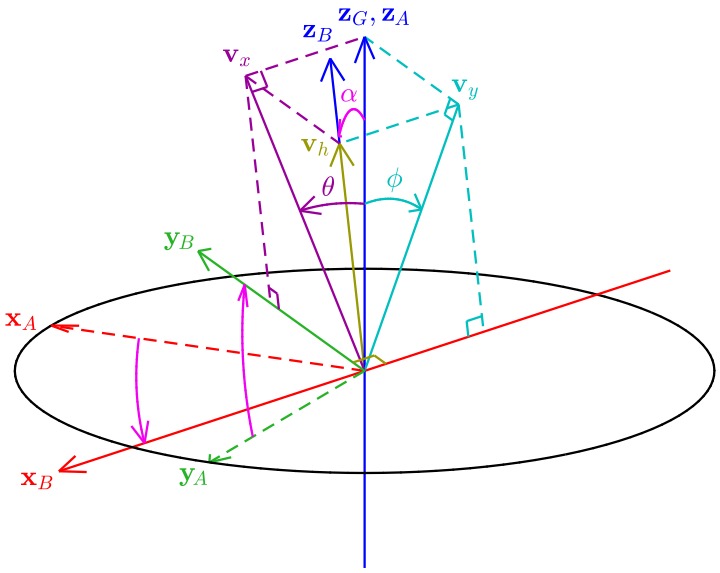


Figure 5.6: Definition of the fused angles parameters (θ, ϕ, h) for a tilt rotation component from {A} to {B}. Refer to Figure 5.4 for a visual definition of the intermediate frame {A}, and the fused yaw ψ used to define it. The frame {A} is rotated onto frame {B} through a tilt angle of α in such a way that $\mathbf{z}_A \equiv \mathbf{z}_G$ rotates directly onto $\mathbf{z}_B$. The fused pitch θ and fused roll ϕ are defined as the angles between $\mathbf{z}_G$ and the $\mathbf{y}_B\mathbf{z}_B$ and $\mathbf{x}_B\mathbf{z}_B$ planes respectively. The hemisphere h is $+1$ if $\mathbf{z}_G$ and $\mathbf{z}_B$ are mutually in the same hemisphere, and -1 if not. For aid of visualisation, note that $\mathbf{x}_B$ is pointing left-downwards parallel to the page, and $\mathbf{y}_B$ is pointing up-leftwards out of the page.

where $\text{sign}(\cdot)$ is a sign function that takes on only the values ± 1. Note that $\text{sign}(\cdot)$ differs from the normal sign function $\text{sgn}(\cdot)$ in that $\text{sign}(0) = 1$, whereas $\text{sgn}(0) = 0$.

Using the notion of the fused hemisphere, the triplet (θ, ϕ, h) becomes a complete description of the tilt rotation component of a rotation, just like the parameter pair (γ, α) is. Thus, together with the fused yaw ψ, we define the fused angles parameterisation of the rotation from {G} to {B} as

$$\begin{aligned} {}^{G}_{B}F &= (\psi, \theta, \phi, h) \\ &\in (-\pi, \pi] \times [-\tfrac{\pi}{2}, \tfrac{\pi}{2}] \times [-\tfrac{\pi}{2}, \tfrac{\pi}{2}] \times \{-1, 1\} \equiv \hat{\mathbb{F}}. \end{aligned} \tag{5.39}$$

The fused angles parameters corresponding to the identity rotation are given by $(0, 0, 0, 1) \in \hat{\mathbb{F}}$. A hat is used for the domain $\hat{\mathbb{F}}$ in Equation (5.39), because the given definition is in fact a *superset* of the true domain $\mathbb{F}$ of the fused angles parameterisation. From Equation (5.37), it can be seen that

$${}^{B}z_{Gx} = -\sin\theta, \tag{5.40a}$$

$${}^{B}z_{Gy} = \sin\phi. \tag{5.40b}$$

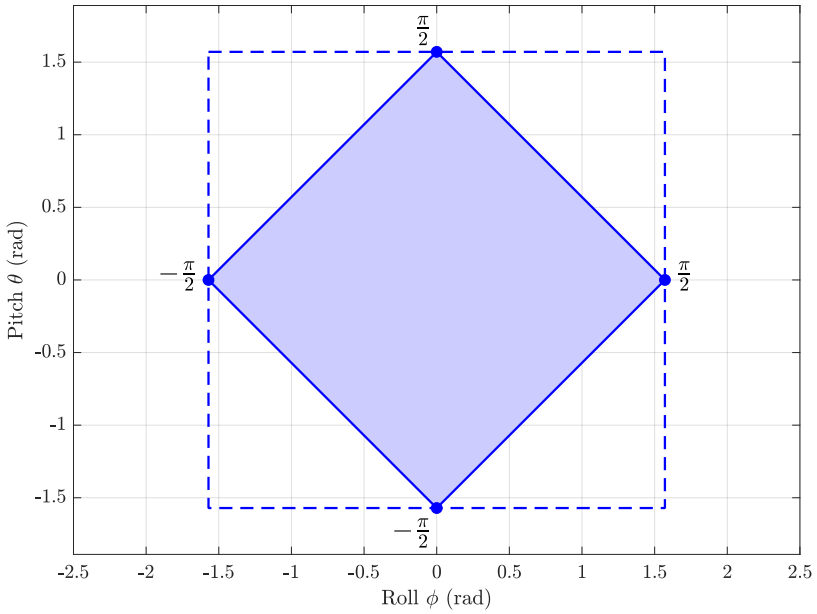


Figure 5.7: An illustration of the true domain of the fused pitch and roll. While nominally the domain is $(\phi, \theta) \in [-\frac{\pi}{2}, \frac{\pi}{2}] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$, we see through the application of the sine sum criterion $|\theta| + |\phi| \leq \frac{\pi}{2}$ that the true domain is given by the shaded diamond-shaped region. The nominal square domain corresponds to $\hat{\mathbb{F}}$, and the true diamond domain corresponds to $\mathbb{F}$.

As ${}^B\mathbf{z}_G$ is a unit vector, we must have that

$$ {}^Bz_{Gx}^2 + {}^Bz_{Gy}^2 \leq 1, \tag{5.41} $$

so from Equation (5.40), we deduce that θ and ϕ must satisfy

$$ \sin^2\theta + \sin^2\phi \leq 1. \tag{5.42} $$

This is referred to as the sine sum criterion, and given that the fused pitch and roll are both restricted to a domain of $[-\frac{\pi}{2}, \frac{\pi}{2}]$, this is completely equivalent to

$$ |\theta| + |\phi| \leq \tfrac{\pi}{2}. \tag{5.43} $$

The region in the fused pitch vs. fused roll space that is defined by the sine sum criterion is shown in Figure 5.7. The true domain $\mathbb{F}$ of the fused angles representation is given by the restriction of $\hat{\mathbb{F}}$ to this region. The domains of the fused yaw and hemisphere parameters remain unchanged however.

5.3.5 Tilt Phase Space

As an alternative to fused angles for parameterising the tilt rotation component of a rotation in a way that yields pitch and roll components,

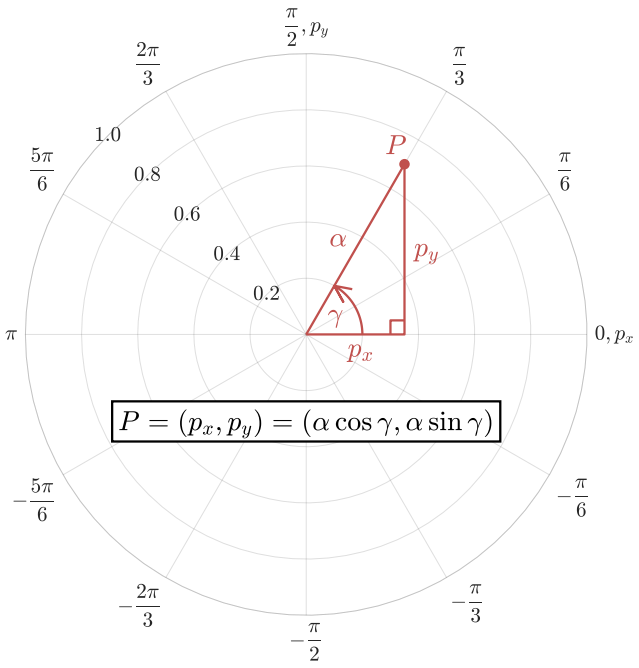


Figure 5.8: A polar plot of the 2D tilt phase space parameters $(p_x, p_y) \in \mathbb{P}^2$. The x and y-axes correspond to p_x and p_y respectively, and in terms of polar coordinates, γ is the polar angle and α is the radius. Note that although the radial scale in this plot only goes up to 1.0, this is just an arbitrary choice, and can of course be extended to π or more.

we now introduce the closely related tilt phase space parameterisation. The tilt phase space combines the ability of fused angles to quantify the amounts of rotation in the major planes, with the ability of tilt angles to optionally also represent tilt rotations over 180°. The tilt phase space also enables a more vectorial view of rotations, while still adhering to the partition of rotations into fused yaw and tilt components, as required. There are two main variants of the tilt phase space, relative and absolute, and for each case there is the choice to either consider all three parameters (3D), or only the two that quantify the tilt rotation component (2D). If referring to just 'the tilt phase space', without a qualifier, by default this refers to the relative tilt phase space, and from the context it is usually clear whether just the 2D pitch and roll parameters are being considered, or whether the 3D parameters, including the yaw, are being considered. The relative and absolute tilt phase spaces are defined as follows.

5.3.5.1 *Relative Tilt Phase Space*

Given the tilt angles representation ${}^G_B T = (\psi, \gamma, \alpha)$ of a rotation from {G} to {B}, the phase roll p_x, phase pitch p_y, and fused yaw p_z are defined by

$$p_x = \alpha \cos \gamma, \tag{5.44a}$$
$$p_y = \alpha \sin \gamma, \tag{5.44b}$$
$$p_z = \psi. \tag{5.44c}$$

The 3D relative tilt phase space representation is then given by

$${}^G_B P = (p_x, p_y, p_z) \in \mathbb{R}^3 \equiv \mathbb{P}^3. \tag{5.45}$$

The relative tilt phase parameters corresponding to the identity rotation are given by $(0, 0, 0) \in \mathbb{P}^3$. Often, when working with tilt rotations, the fused yaw component is either zero or irrelevant. In such cases, the 2D relative tilt phase space representation can be used instead, given by

$${}^G_B P = (p_x, p_y) \in \mathbb{R}^2 \equiv \mathbb{P}^2, \tag{5.46}$$

which essentially just omits the fused yaw component p_z. The 2D formulation of the tilt phase space is the most frequently encountered one in this thesis, specifically because of its application to bipedal walking (e.g. in Chapters 10 and 11). It should be noted that $\mathbb{R}$ was used in full generality in Equations (5.45) and (5.46) for the definition of the tilt phase space domains $\mathbb{P}^2$ and $\mathbb{P}^3$, in order to naturally be able to represent rotations of all magnitudes (i.e. unbounded rotations). As is frequently the case however, if we are dealing strictly with rotations where only the final orientation of the coordinate frame matters, then the true domain of the tilt phase space would rather be

$$\mathbb{P}^3 \equiv \bar{\mathcal{D}}^2(\pi) \times (-\pi, \pi], \tag{5.47a}$$
$$\mathbb{P}^2 \equiv \bar{\mathcal{D}}^2(\pi), \tag{5.47b}$$

where $\bar{\mathcal{D}}^2(\pi)$ is the closed 2-dimensional disc of radius π centred at the origin. That is, put more simply, in the phase pitch vs. phase roll space, $\bar{\mathcal{D}}^2(\pi)$ corresponds to a circle of radius π and all of its interior.

Plots of phase pitch p_y against phase roll p_x are customarily drawn on a polar plot to highlight the relationship between the 'Cartesian' tilt phase coordinates and the 'polar' tilt angles coordinates α and γ, as made explicit in Figure 5.8. The circular boundary of a polar plot also well-represents the natural domain of all tilt rotations if only bounded rotations, i.e. tilt rotations of up to 180°, are considered, as this domain is circular as well, as just discussed. An example of three different pure tilt rotations, and what their respective tilt phase space coordinates are as viewed in a polar plot, is shown in Figure 5.9.

The tilt phase parameters were defined in terms of the tilt angles parameters in Equation (5.44), but as discussed in Section 5.3.3, the tilt

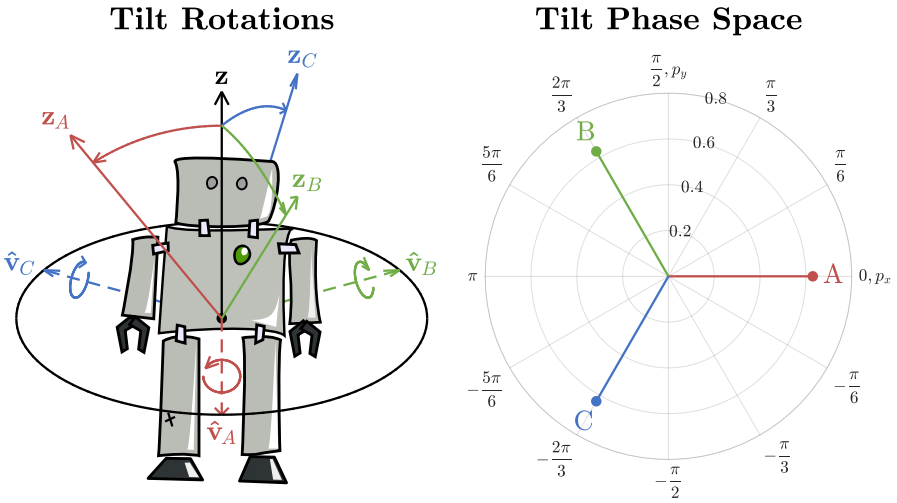


Figure 5.9: Three different examples of tilt rotations applied to an upright standing robot (left), and the corresponding points in the 2D tilt phase space (right). The respective CCW axes of rotation in the horizontal **xy** plane, $\hat{\mathbf{v}}_*$, are shown in dashed. Note, for example, that A is a rotation about the positive x-axis, i.e. $\hat{\mathbf{v}}_A = (1, 0, 0)$, so its corresponding p_y component is zero, and its p_x component is positive (can be seen in the phase plot on the right).

axis angle γ and the tilt angle α are not always unique in describing a particular tilt rotation component. Thus, it is important to check that the tilt phase space is nonetheless well-defined, and furthermore that it is unique, continuous and smooth (infinitely differentiable) everywhere except for at the fused yaw singularity. These are critically important properties of the tilt phase space, as they imply that the representation can for example safely be differentiated and related to angular velocities. The well-definedness, continuity and smoothness of the tilt phase space, in particular at the identity tilt rotation ${}^G_B P = (0, 0) \in \mathbb{P}^2$, is demonstrated in Allgeuer (2020).

5.3.5.2 *Absolute Tilt Phase Space*

The absolute tilt phase space shares the same definition as the relative tilt phase space, only with the absolute tilt axis angle $\tilde{\gamma} = \gamma + \psi$ being used instead of γ. That is,

$$\tilde{p}_x = \alpha \cos \tilde{\gamma}, \tag{5.48a}$$
$$\tilde{p}_y = \alpha \sin \tilde{\gamma}, \tag{5.48b}$$
$$\tilde{p}_z = \psi. \tag{5.48c}$$

If a clear distinction between $\tilde{\gamma}$ and γ is required, then γ can be referred to as the relative tilt axis angle. The triplet

$$\begin{aligned} {}^G_B\tilde{T} &= (\psi, \tilde{\gamma}, \alpha) \\ &\in (-\pi, \pi] \times (-\pi, \pi] \times [0, \pi] \equiv \tilde{\mathbb{T}}, \end{aligned} \tag{5.49}$$

is sometimes referred to as the absolute tilt angles parameterisation of a rotation, and once again, if clear distinction is required, the standard tilt angles representation can be referred to as the relative tilt angles.

Based on Equation (5.48), the 3D absolute tilt phase space representation is given by

$$\overset{G}{\underset{B}{\tilde{P}}} = (\tilde{p}_x, \tilde{p}_y, \tilde{p}_z) \in \mathbb{R}^3 \equiv \tilde{\mathbb{P}}^3, \tag{5.50}$$

and the 2D absolute tilt phase space representation is given by

$$\overset{G}{\underset{B}{\tilde{P}}} = (\tilde{p}_x, \tilde{p}_y) \in \mathbb{R}^2 \equiv \tilde{\mathbb{P}}^2, \tag{5.51}$$

The absolute tilt phase parameters corresponding to the identity rotation are $(0, 0, 0) \in \tilde{\mathbb{P}}^3$, as before. Geometrically, the absolute tilt axis angle $\tilde{\gamma}$ is equivalent to the CCW angle from $\mathbf{x}_G$ to $\hat{\mathbf{v}}$ (see Figure 5.4), instead of the angle from $\mathbf{x}_A$ to $\hat{\mathbf{v}}$, as for the relative tilt axis angle γ. The relation between the absolute and relative tilt phase spaces is given by

$$\tilde{p}_x = p_x \cos\psi - p_y \sin\psi, \tag{5.52a}$$
$$\tilde{p}_y = p_x \sin\psi + p_y \cos\psi, \tag{5.52b}$$
$$\tilde{p}_z = p_z, \tag{5.52c}$$

for the conversion from relative to absolute, and

$$p_x = \tilde{p}_x \cos\psi + \tilde{p}_y \sin\psi, \tag{5.53a}$$
$$p_y = -\tilde{p}_x \sin\psi + \tilde{p}_y \cos\psi, \tag{5.53b}$$
$$p_z = \tilde{p}_z, \tag{5.53c}$$

for the conversion from absolute to relative. These equations essentially correspond to simple 2D rotations by $\pm\psi$ for (p_x, p_y) and $(\tilde{p}_x, \tilde{p}_y)$.

The absolute tilt phase space is only a slight variation of the relative tilt phase space, and it should be noted that for tilt rotations the two representations are identical anyway. That is,

$$p_z = \psi = 0 \implies \overset{G}{\underset{B}{\tilde{P}}} \equiv \overset{G}{\underset{B}{P}} \tag{5.54}$$

All previous arguments about bounded vs. unbounded rotations, the true bounded domain of the parameters, phase plots, continuity and smoothness and so on, hold just the same for the absolute tilt phase space. In fact, all results that hold for one of the two spaces in general correspond trivially to completely analogous results for the other.

5.3.5.3 *Tilt Vector Addition*

If one takes two different tilt rotations and combines them as sequential rotations, the result is in general not a tilt rotation, and depends on the order in which the two rotations are combined. In mathematical terms,

this is to say that the binary operation given by standard rotation composition is not *closed* on the set of all tilt rotations, and not *commutative*. This can be problematic, if for example in an orientation feedback controller it is required to combine the contributions of multiple tilt rotation feedback paths, e.g. for Proportional-Integral-Derivative (PID) feedback like in Section 11.2, as using standard rotation composition cannot result in an unambiguous final tilt rotation. The 2D tilt phase space provides a way of defining a useful and meaningful addition operator for tilt rotations that is closed, commutative, and like standard rotation composition, associative. This new binary operation '$\oplus$' is referred to as tilt vector addition, and for $P_1, P_2 \in \mathbb{P}^2$ is defined by

$$P_1 \oplus P_2 = (p_{x1} + p_{x2},\ p_{y1} + p_{y2}) \in \mathbb{P}^2. \tag{5.55}$$

In terms of tilt angles this is equivalent to

$$(\alpha_1 c_{\gamma_1},\ \alpha_1 s_{\gamma_1}) \oplus (\alpha_2 c_{\gamma_2},\ \alpha_2 s_{\gamma_2}) = (\alpha_1 c_{\gamma_1} + \alpha_2 c_{\gamma_2},\ \alpha_1 s_{\gamma_1} + \alpha_2 s_{\gamma_2}), \tag{5.56}$$

where the standard abbreviations c_* and s_* for $\cos(*)$ and $\sin(*)$ have been used. If we let $P_3 \in \mathbb{P}^2$ denote the tilt rotation corresponding to the sum $P_1 \oplus P_2$, and let (γ_3, α_3) denote the associated tilt angles parameters, then we can also write

$$(\gamma_1, \alpha_1) \oplus (\gamma_2, \alpha_2) = (\gamma_3, \alpha_3), \tag{5.57}$$

noting that this is not just elementwise vector addition, but rather shorthand for applying Equation (5.56) and then converting the result back into tilt angles. The action of tilt vector addition is illustrated in Figure 5.10.

Although tilt vector addition is nominally defined in terms of the relative tilt phase space, completely analogous definitions of tilt vector addition naturally hold for the absolute tilt phase space as well. An important observation however, which supports the self-consistency of the definition of tilt vector addition, is that regardless of whether two tilt rotations are added in the relative or absolute spaces, the outcome is the same. This means that if two tilt rotations (or tilt rotation components for rotations of equal fused yaw) are expressed in terms of their relative and absolute tilt phase space parameters and added using tilt vector addition, the resulting tilt rotations expressed as, for example, quaternions are identical.

5.4 ROTATION REPRESENTATION CONVERSIONS

Given the many different existing rotation representations and the new ones that have been introduced in this chapter, it is important to be able to relate them to each other and convert between them. This section summarises all relevant conversion equations, and also sheds light on the intricate connections between the various representations,

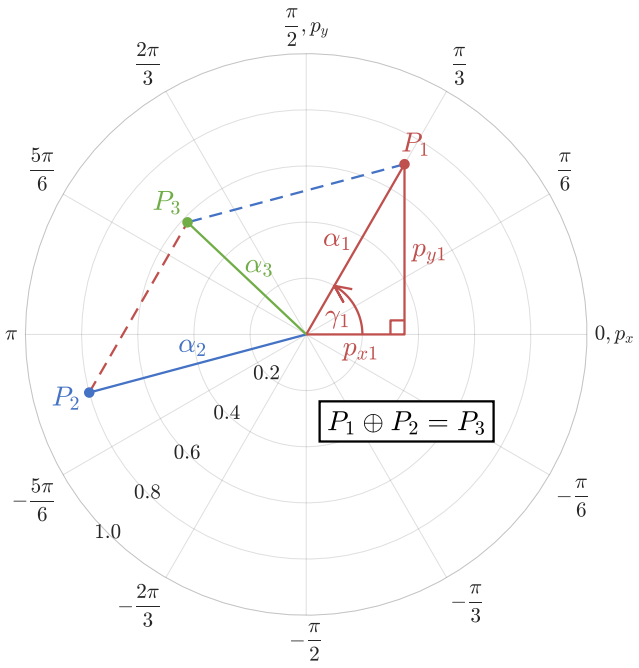


Figure 5.10: An illustration of tilt vector addition in the 2D tilt phase space, where as indicated we have that $P_1 \oplus P_2 = P_3$. When expressed in terms of the tilt phase space, it can be seen that tilt vector addition is equivalent to standard vector addition. In terms of tilt angles, we have in this case that a tilt of 0.7 rad about $\gamma_1 = 60°$, added to a tilt of 0.8 rad about $\gamma_2 = 195°$, gives a combined tilt of 0.5814 rad about $\gamma_3 = 136.6°$.

as required for later parts of this thesis. Conversions from the z-vector parameterisation to other forms have been omitted, as these are implicit from the definitions that have been given in this chapter so far, e.g. like in Equations (5.34) and (5.37). Conversions to and from the tilt phase space are also in general only presented in terms of the relative tilt phase space, as Equations (5.52) and (5.53) can then be used to relate this to the absolute tilt phase space. Throughout this section, as well as throughout the remainder of this thesis, the shorthand is used that $c_* \equiv \cos(*)$, $s_* \equiv \sin(*)$ and $\bar{*} \equiv \frac{1}{2}*$.

5.4.1 From Tilt Angles To

Given a tilt angles rotation $T = (\psi, \gamma, \alpha) \in \mathbb{T}$, the corresponding rotation matrix is given by

$$R = R_z(\psi) R_{\hat{\mathbf{v}}}(\alpha) \tag{5.58a}$$

$$= \begin{bmatrix} c_\gamma c_{\gamma+\psi} + c_\alpha s_\gamma s_{\gamma+\psi} & s_\gamma c_{\gamma+\psi} - c_\alpha c_\gamma s_{\gamma+\psi} & s_\alpha s_{\gamma+\psi} \\ c_\gamma s_{\gamma+\psi} - c_\alpha s_\gamma c_{\gamma+\psi} & s_\gamma s_{\gamma+\psi} + c_\alpha c_\gamma c_{\gamma+\psi} & -s_\alpha c_{\gamma+\psi} \\ -s_\alpha s_\gamma & s_\alpha c_\gamma & c_\alpha \end{bmatrix}, \tag{5.58b}$$

where $R_z(\cdot)$ is a CCW rotation by angle $\cdot$ about the z-axis, and $R_{\hat{\mathbf{v}}}(\cdot)$ is a CCW rotation by angle $\cdot$ about the vector $\hat{\mathbf{v}} = (c_\gamma, s_\gamma, 0)$. The z-vector parameterisation is thus given by the bottom row

$$ {}^{B}\mathbf{z}_{G} = (-\sin\alpha\sin\gamma,\ \sin\alpha\cos\gamma,\ \cos\alpha). \tag{5.59} $$

The quaternion corresponding to T, on the other hand, is given by

$$ \begin{aligned} q &= q_z(\psi)q_{\hat{\mathbf{v}}}(\alpha) && \text{(5.60a)} \\ &= (c_{\bar{\alpha}}c_{\bar{\psi}},\ s_{\bar{\alpha}}c_{\bar{\psi}+\gamma},\ s_{\bar{\alpha}}s_{\bar{\psi}+\gamma},\ c_{\bar{\alpha}}s_{\bar{\psi}}). && \text{(5.60b)} \end{aligned} $$

In specific reference to Section 5.3.2.2 and Equation (5.31), the quaternion corresponding to the tilt rotation component of a rotation is thus given by

$$ \begin{aligned} {}^{A}_{B}q &= (w, x, y, 0) && \text{(5.61a)} \\ &= (\cos\tfrac{\alpha}{2},\ \sin\tfrac{\alpha}{2}\cos\gamma,\ \sin\tfrac{\alpha}{2}\sin\gamma,\ 0). && \text{(5.61b)} \end{aligned} $$

The tilt phase representation of the rotation T is clear by definition, i.e. from Equations (5.44) and (5.48), and aside from the fused yaw ψ, which is common to both representations, the fused angles $F = (\psi, \theta, \phi, h)$ can be calculated from the tilt angles parameters using

$$ \begin{aligned} \sin\theta &= \sin\alpha\sin\gamma, && \text{(5.62a)} \\ \sin\phi &= \sin\alpha\cos\gamma. && \text{(5.62b)} \end{aligned} $$

The fused hemisphere h is given by

$$ \begin{aligned} h &= \operatorname{sign}(\cos\alpha) && \text{(5.63a)} \\ &= \begin{cases} 1 & \text{if } \alpha \leq \frac{\pi}{2}, \\ -1 & \text{if } \alpha > \frac{\pi}{2}, \end{cases} && \text{(5.63b)} \end{aligned} $$

where we recall that $\alpha \in [0, \pi]$. Interestingly, from Equation (5.62) we note that

$$ \sin^2\theta + \sin^2\phi = \sin^2\alpha. \tag{5.64} $$

This, amongst other things, gives new insight into why the sine sum criterion in Equation (5.42) must hold.

5.4.2 From Fused Angles To

Given a fused angles rotation $F = (\psi, \theta, \phi, h) \in \mathbb{F}$, the tilt angles parameters can be calculated using

$$ \begin{aligned} \gamma &= \operatorname{atan2}(\sin\theta, \sin\phi), && \text{(5.65a)} \\ \alpha &= \operatorname{acos}\left(h\sqrt{1 - \sin^2\theta - \sin^2\phi}\right), && \text{(5.65b)} \end{aligned} $$

where for numerical computation one may choose to use the identity

$$1 - \sin^2\theta - \sin^2\phi \equiv \cos(\theta+\phi)\cos(\theta-\phi). \tag{5.66}$$

Most conversions involving fused angles involve the calculation of the tilt angles parameters γ and α, followed by the use of the appropriate tilt angles conversion formulas from Section 5.4.1. This is why the tilt angles representation is sometimes referred to as an *intermediate representation*. Despite this however, in most cases there are still slight simplifications or alternatives that can be found, either to make the calculation somewhat more direct, or to highlight mathematical links between fused angles and other representations that are useful for analysis. For instance, the rotation matrix representation of F can be expressed as

$$R = \begin{bmatrix} c_\gamma c_{\gamma+\psi} + c_\alpha s_\gamma s_{\gamma+\psi} & s_\gamma c_{\gamma+\psi} - c_\alpha c_\gamma s_{\gamma+\psi} & s_\psi s_\phi + c_\psi s_\theta \\ c_\gamma s_{\gamma+\psi} - c_\alpha s_\gamma c_{\gamma+\psi} & s_\gamma s_{\gamma+\psi} + c_\alpha c_\gamma c_{\gamma+\psi} & s_\psi s_\theta - c_\psi s_\phi \\ -s_\theta & s_\phi & c_\alpha \end{bmatrix}, \tag{5.67}$$

leading to a z-vector parameterisation of

$$\begin{aligned} {}^B\mathbf{z}_G &= (-\sin\theta,\, \sin\phi,\, \cos\alpha) && \text{(5.68a)} \\ &= \left(-\sin\theta,\, \sin\phi,\, h\sqrt{1-\sin^2\theta-\sin^2\phi}\right). && \text{(5.68b)} \end{aligned}$$

The quaternion q corresponding to the fused angles F can also be expressed partially in terms of the fused angles parameters using

$$q = \begin{cases} \frac{\tilde{q}_p}{\|\tilde{q}_p\|} & \text{if } h = 1, \\ \frac{\tilde{q}_n}{\|\tilde{q}_n\|} & \text{if } h = -1, \end{cases} \tag{5.69}$$

where the unnormalised positive and negative hemisphere quaternions $\tilde{q}_p$ and $\tilde{q}_n$ are given by

$$\begin{aligned} \tilde{q}_p &= \left(c_{\bar{\psi}}(1+c_\alpha),\, s_\phi c_{\bar{\psi}} - s_\theta s_{\bar{\psi}},\, s_\phi s_{\bar{\psi}} + s_\theta c_{\bar{\psi}},\, s_{\bar{\psi}}(1+c_\alpha)\right), && \text{(5.70a)} \\ \tilde{q}_n &= \left(s_\alpha c_{\bar{\psi}},\, c_{\bar{\psi}+\gamma}(1-c_\alpha),\, s_{\bar{\psi}+\gamma}(1-c_\alpha),\, s_\alpha s_{\bar{\psi}}\right). && \text{(5.70b)} \end{aligned}$$

The respective quaternion norms are analytically given by

$$\begin{aligned} \|\tilde{q}_p\| &= \sqrt{2(1+\cos\alpha)} = 2\cos\tfrac{\alpha}{2}, && \text{(5.71a)} \\ \|\tilde{q}_n\| &= \sqrt{2(1-\cos\alpha)} = 2\sin\tfrac{\alpha}{2}. && \text{(5.71b)} \end{aligned}$$

Note that both cases in Equation (5.69) can actually be used everywhere in both fused hemispheres, but the normalisation of $\tilde{q}_p$ becomes numerically sensitive towards the bottom ($\alpha = \pi$) of the negative hemisphere, and the normalisation of $\tilde{q}_n$ similarly becomes numerically sensitive towards the top ($\alpha = 0$) of the positive hemisphere. This

can be seen from the quaternion norms given in Equation (5.71). It should also be noted that the value of α does not need to be computed in order to evaluate Equation (5.70), only $\cos\alpha$ and $\sin\alpha$, which can be obtained from Equations (5.64) and (5.65b) directly.

The conversion from fused angles to the tilt phase space is also nominally performed via calculation of α and γ, but the two spaces can also be more directly related using

$$p_x = \frac{\sin\phi}{\operatorname{sinc}\alpha} = \left(\frac{\alpha}{\sin\alpha}\right)\sin\phi, \tag{5.72a}$$

$$p_y = \frac{\sin\theta}{\operatorname{sinc}\alpha} = \left(\frac{\alpha}{\sin\alpha}\right)\sin\theta, \tag{5.72b}$$

where

$$\operatorname{sinc}\alpha = \begin{cases} \frac{\sin\alpha}{\alpha} & \text{if } \alpha \neq 0, \\ 1 & \text{if } \alpha = 0, \end{cases} \tag{5.73}$$

is the cardinal sine function, a smooth function of α.

5.4.3 From Tilt Phase Space To

The conversions from the tilt phase space $P = (p_x, p_y, p_z) \in \mathbb{P}^3$ to all other representations mentioned in this chapter are nominally performed via the tilt angles representation, the conversion to which is given by

$$\psi = p_z, \tag{5.74a}$$

$$\gamma = \operatorname{atan2}(p_y, p_x), \tag{5.74b}$$

$$\alpha = \sqrt{p_x^2 + p_y^2}. \tag{5.74c}$$

The only exception is the fused angles representation, which can be related to the tilt phase space directly using Equation (5.74c), followed by

$$\sin\theta = p_y \operatorname{sinc}\alpha, \tag{5.75a}$$

$$\sin\phi = p_x \operatorname{sinc}\alpha. \tag{5.75b}$$

The tilt phase space is another example of why the tilt angles representation is sometimes referred to as an *intermediate representation*.

5.4.4 From Quaternion To

Given the quaternion $q = (w, x, y, z) \in \mathbb{Q}$, the corresponding rotation matrix is given by

$$R = \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-wz) & 2(xz+wy) \\ 2(xy+wz) & 1-2(x^2+z^2) & 2(yz-wx) \\ 2(xz-wy) & 2(yz+wx) & 1-2(x^2+y^2) \end{bmatrix}, \tag{5.76}$$

leading to a z-vector parameterisation of

$$ {}^{B}\mathbf{z}_{G} = \Big(2(xz - wy),\, 2(yz + wx),\, 1 - 2(x^2 + y^2)\Big). \tag{5.77} $$

As given in Equation (5.27), the quaternion parameters define the fused yaw ψ as

$$ \psi = \operatorname{wrap}\big(2\operatorname{atan2}(z, w)\big), \tag{5.78} $$

where $\operatorname{wrap}(\cdot)$ is a function that wraps an angle to the range $(-\pi, \pi]$ by multiples of 2π. The remaining tilt angles parameters are then given by

$$ \gamma = \operatorname{atan2}(wy - xz, wx + yz), \tag{5.79a} $$

$$ \alpha = \operatorname{acos}(2(w^2 + z^2) - 1) \tag{5.79b} $$

$$ = 2\operatorname{acos}\Big(\sqrt{w^2 + z^2}\Big), \tag{5.79c} $$

and the remaining fused angles parameters are given by

$$ \theta = \operatorname{asin}\big(2(wy - xz)\big), \tag{5.80a} $$

$$ \phi = \operatorname{asin}\big(2(wx + yz)\big), \tag{5.80b} $$

$$ h = \operatorname{sign}\big(w^2 + z^2 - \tfrac{1}{2}\big). \tag{5.80c} $$

Care has to be taken with the combination of Equation (5.78) and Equation (5.79a) near the fused yaw singularity $\alpha = \pi$, which is equivalent to $w = z = 0$. The fused yaw has an essential discontinuity there, so both equations necessarily become numerically sensitive in that region. Due to the resulting effects, the calculated tilt parameters may not accurately describe the original quaternion. Assuming a rotation is at the fused yaw singularity, more numerically consistent equations for the tilt angles parameters are $\psi = 0$, $\alpha = \pi$, and

$$ \gamma = \operatorname{atan2}(y, x). \tag{5.81} $$

Once the tilt angles parameters are known, the tilt phase parameters can trivially be calculated using Equations (5.44) and (5.48). Alternatively, the tilt phase parameters can be calculated more directly from the quaternion parameters using

$$ p_x = \frac{\alpha}{S}(wx + yz), \tag{5.82a} $$

$$ p_y = \frac{\alpha}{S}(wy - xz), \tag{5.82b} $$

$$ p_z = \operatorname{wrap}\big(2\operatorname{atan2}(z, w)\big), \tag{5.82c} $$

where

$$ \alpha = \operatorname{acos}\big((w^2 + z^2) - (x^2 + y^2)\big), \tag{5.83a} $$

$$ S = \tfrac{1}{2}\sin\alpha = \sqrt{(w^2 + z^2)(x^2 + y^2)}\,. \tag{5.83b} $$

Similar to the need for Equation (5.81), if $S = 0$ then

$$(p_x, p_y) = \begin{cases} (0,0) & \text{if } w^2 + z^2 \geq x^2 + y^2, \\ \pi \dfrac{(x,y)}{\|(x,y)\|} & \text{otherwise.} \end{cases} \tag{5.84}$$

It should be noted that if the operand of $\text{acos}(\cdot)$ in Equation (5.83a) is divided by $(w^2 + z^2) + (x^2 + y^2)$, which is nominally 1 for a unit quaternion $q = (w, x, y, z)$, then the entire conversion algorithm to the tilt phase space is quaternion magnitude independent.

5.4.5 From Rotation Matrix To

The conversions from rotation matrices to other representations is discussed in detail in Allgeuer (2020). As a brief excerpt however, the tilt rotation component parameters for the tilt angles representation can be calculated as

$$\gamma = \text{atan2}(-R_{31}, R_{32}), \tag{5.85a}$$
$$\alpha = \text{acos}(R_{33}), \tag{5.85b}$$

and the corresponding fused angles parameters can be calculated as

$$\theta = \text{asin}(-R_{31}), \tag{5.86a}$$
$$\phi = \text{asin}(R_{32}), \tag{5.86b}$$
$$h = \text{sign}(R_{33}). \tag{5.86c}$$

5.5 SINGULARITY ANALYSIS

When examining rotation representations, it is important to identify and precisely quantify any singularities. It was shown by Stuelpnagel (1964) that it is topologically impossible to have a one-to-one and global three-dimensional, i.e. minimal, parameterisation of the rotation group without any singular points. For an n-dimensional parameterisation to be one-to-one and global without any singular points, there must exist a differentiable one-to-one map with differentiable inverse that carries the rotation space $\text{SO}(3)$ to the required parameter domain subset of $\mathbb{R}^n$. Thus, for $n = 3$, it must be the case for every parameterisation that either the map is not one-to-one, not differentiable, or not inverse differentiable.

Of greatest concern to the mathematical and practical applications of particular rotation representations is the location and prevalence of singular points. The possible types of singularities of rotation representations can be characterised to primarily include:

(i) Rotations that do not possess a unique parameterised representation,

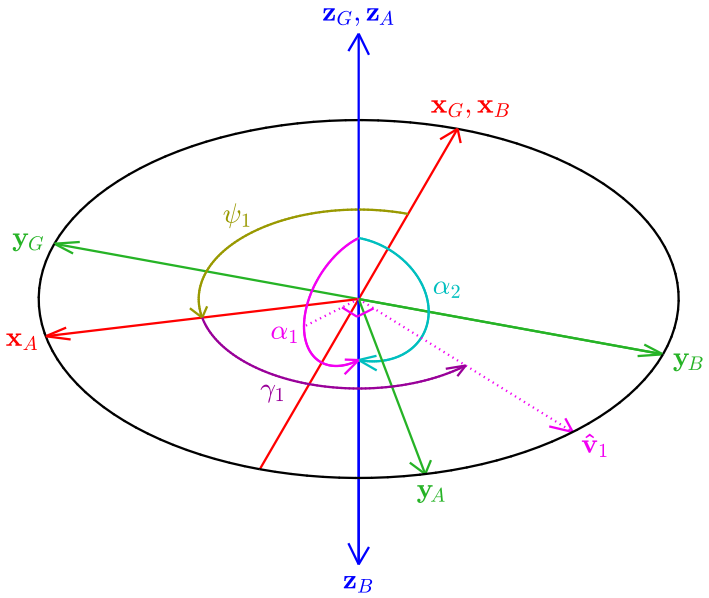


Figure 5.11: At the fused yaw singularity, i.e. $\mathbf{z}_B = -\mathbf{z}_G$, there are multiple combinations of yaw and tilt that can take a global frame {G} to the local frame {B}. In this example, a yaw rotation of ψ_1 followed by a tilt rotation of (γ_1, α_1) (i.e. a CCW rotation of {A} by $\alpha_1 = \pi$ radians about $\hat{\mathbf{v}}_1$), is equivalent to a yaw rotation of $\psi_2 = 0$ (i.e. no yaw rotation) followed by a tilt rotation of $(0, \alpha_2)$ (i.e. a CCW rotation of $\alpha_2 = \pi$ radians about $\mathbf{x}_G$).

(ii) Sets of parameters that do not correspond to a unique rotation,

(iii) Rotations in the neighbourhood of which the sensitivity of the rotation to parameters map is unbounded, and,

(iv) Sets of parameters in the neighbourhood of which the sensitivity of the parameters to rotation map is unbounded.

It should be noted that strictly speaking, all cyclic parameters that loop around at $\pm\pi$ break the mathematical condition of differentiability, as this requires continuity. This is not seen to result in true singularities or discontinuities however, as the cyclic parameters are clearly viewed as having a cyclic topology, for which the jump at $\pm\pi$ is not a discontinuity.

5.5.1 Fused Yaw Singularity

The most prominent singularity present in the new representations developed in this thesis is the fused yaw singularity, which has been mentioned before in Section 5.3.1. The geometric definition of the fused yaw ψ requires a direct planar rotation to be constructed from $\mathbf{z}_B$ to $\mathbf{z}_G$, but this rotation is not uniquely defined in situations where $\mathbf{z}_B$ and $\mathbf{z}_G$ are antiparallel, i.e. when they point in opposite

directions. As shown in Figure 5.11, in such situations there are multiple combinations of yaw and tilt that can be used to arrive at the final body-fixed frame {B}, leading to an ambiguity in the definition of the associated parameters. The following are all completely equivalent characterisations of rotations at the fused yaw singularity:

- Rotations for which the body-fixed z-axis points vertically downwards,
- Rotations by 180° about an axis in the **xy** plane, i.e. pure 180° tilt rotations,
- Rotations for which ${}^{G}z_{Bz} \equiv {}^{B}z_{Gz} = -1$, i.e. $\mathbf{z}_B = -\mathbf{z}_G$,
- Rotation matrices for which $R_{33} = -1$,
- Quaternion rotations for which $w = z = 0$,
- Tilt angles rotations for which $\alpha = \pi$,
- Fused angles rotations for which $\theta = \phi = 0$ and $h = -1$,
- Tilt phase space rotations for which $\|(p_x, p_y)\| = \sqrt{p_x^2 + p_y^2} = \pi$.

Conceptually, the fused yaw singularity can be seen to be as 'far away' from the identity rotation as possible, as all fused yaw singular rotations are 180° away from the identity, and no two rotations in the rotation space *can* be separated by more than that. This is particularly useful for applications involving balancing bodies, as in particular for most cases of mobile robotics, a completely inverted pose is the most uncommon part of the rotation space to occur.

Based on the above characterisations of rotations at the fused yaw singularity, expressions can be developed for the formats of the various rotation representations at the fused yaw singularity. For instance:

$$T_{sing} = (\psi, \gamma, \pi), \tag{5.87a}$$
$$F_{sing} = (\psi, 0, 0, -1), \tag{5.87b}$$
$$P_{sing} = (p_x,\ p_y,\ \psi), \tag{5.87c}$$
$$q_{sing} = (0, x, y, 0) \tag{5.87d}$$
$$= \left(0,\ \cos(\gamma + \tfrac{\psi}{2}),\ \sin(\gamma + \tfrac{\psi}{2}),\ 0\right), \tag{5.87e}$$

where $p_x^2 + p_y^2 = \pi^2$ and $x^2 + y^2 = 1$. For rotation matrices, the format of the fused yaw singularity is given by the following expressions:

$$R_{sing} = \begin{bmatrix} R_{11} & R_{12} & 0 \\ R_{12} & -R_{11} & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{5.88a}$$
$$= \begin{bmatrix} x^2 - y^2 & 2xy & 0 \\ 2xy & y^2 - x^2 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{5.88b}$$

$$= \begin{bmatrix} \cos(2\gamma+\psi) & \sin(2\gamma+\psi) & 0 \\ \sin(2\gamma+\psi) & -\cos(2\gamma+\psi) & 0 \\ 0 & 0 & -1 \end{bmatrix}, \tag{5.88c}$$

where $R_{11}^2 + R_{12}^2 = 1$ in Equation (5.88a), and $x^2 + y^2 = 1$ in Equation (5.88b).

As per the characterisation of singularities given at the start of this section, the fused yaw singularity is a singularity of types (i) and (iii) for tilt angles and the tilt phase space, and types (i), (ii) and (iii) for fused angles. The difference comes about from the way that the fused angles tilt rotation component parameters are equal for all fused yaw singular rotations, which is not the case for the other two representations. The singularity is of type (iii) because it constitutes an essential discontinuity in the map from the rotation space to the respective fused yaws. As a result, given any fused yaw singular rotation R, and any neighbourhood U of R in the rotation space $\mathrm{SO}(3)$, for every $\psi \in (-\pi, \pi]$ there exists a rotation in U with a fused yaw of ψ. Despite this discontinuity, by convention when calculating the fused yaw of a singular rotation, the value of $\psi = 0$ is used. This makes sense because the value of zero emerges naturally from the evaluation of $\mathrm{atan2}(0,0)$, which occurs in all formulas for ψ when the input rotation is fused yaw singular, and because it is most natural to parameterise pure tilt rotations in a way that has zero yaw. Despite the convention of using $\psi = 0$ however, for tilt angles and the tilt phase space it is still nonetheless possible to unambiguously interpret sets of singular parameters with $\psi \neq 0$. This is done, for example for the parameter set $T = (\psi, \gamma, \pi) \in \mathbb{T}$, by first geometrically applying the yaw rotation that is defined by ψ, followed by the tilt rotation that is defined by γ and π. This yields a well-defined and unambiguous final rotation in $\mathrm{SO}(3)$ corresponding to the tilt angles parameters T.

5.5.2 Other Singularities

In knowledge of the nature of the fused yaw singularity and its effects also on the other parameters, the remaining singularities in the new representations developed in this thesis can be determined by examining just pure tilt rotations of up to but not including 180°. This follows directly from the general continuity of the fused yaw, and the characterisation that the fused yaw singular rotations are pure 180° tilt rotations. Consider for example the fused angles parameterisation (θ, ϕ, h) of a tilt rotation. The entries $R_{ij} \in [-1, 1]$ of the associated rotation matrix are well-known to be a continuous, and in fact smooth, function of the underlying 3D rotation. As such, from Equation (5.86) and the continuity of the standard $\mathrm{asin}(\cdot)$ function, it can be seen that the fused pitch θ and the fused roll ϕ are continuous over the entire rotation space. Furthermore, the hemisphere parameter of the fused angles representation is uniquely and unambiguously defined over the

rotation space. As a result, despite its discrete and thereby technically discontinuous nature, the hemisphere parameter is not considered to be the cause of any singularities in the fused angles representation.

It can be seen from Equations (5.87e) and (5.88c) that at the fused yaw singularity, the fused yaw ψ and tilt axis angle γ are linearly dependent in the sense that only the value of $\gamma + \frac{\psi}{2}$ matters. Thus, the tilt axis angle, like the fused yaw ψ, also suffers from a singularity at $\alpha = \pi$ due to the fused yaw singularity. In addition to this however, similar to the nature of polar coordinates at the origin, the γ parameter has an essential discontinuity at $\alpha = 0$. The local sensitivity of γ around this discontinuity is not in general a problem however, as the closer α gets to zero, the proportionally less effect the value of γ actually has on the rotation that the parameters represent. This is reflected, for example, by Equation (5.60b) in that all uses of γ are premultiplied by $\sin\frac{\alpha}{2}$, which tends to zero for small values of α. Equation (5.85a) demonstrates through the continuity of $\operatorname{atan2}(\cdot,\cdot)$ away from $(0,0)$ that the tilt axis angle γ is in fact continuous everywhere except for the aforementioned cases of $\alpha = 0$ and $\alpha = \pi$, as these conditions correspond exactly to $R_{31} = R_{32} = 0$. Similarly, Equation (5.85b) demonstrates from the continuity of $\operatorname{acos}(\cdot)$ that the tilt angle α is continuous everywhere, even if it can be observed that there are cusps at $\alpha = 0$ and π.

The singularity analysis for the tilt phase space is slightly more complicated, as the relation to the rotation matrix parameters is somewhat less direct, but the tilt phase space is in fact continuous and smooth everywhere away from the fused yaw singularity. Given the definition of the tilt phase space in terms of the tilt angles parameters γ and α, it is not immediately clear why this should be the case, but a proof involving the smoothness of the cardinal sine function has been provided in Allgeuer (2020).

5.6 SELECTED PROPERTIES OF YAW-TILT ROTATIONS

The fused angles, tilt angles and tilt phase space representations all possess a remarkable number of sometimes subtle, yet powerful, properties that turn out to be useful in many applications and contexts. A small selection of such mathematical and geometric properties are presented in this section. A continuation of many more such properties is presented in Allgeuer (2020).

5.6.1 Links Between Quaternions and Fused Yaw

For rotations away from the fused yaw singularity $\alpha = \pi$, that is, for rotations where the fused yaw is well-defined and unambiguous, inspection of Equation (5.60) reveals that the z-component

$$z = \cos\tfrac{\alpha}{2} \sin\tfrac{\psi}{2} \tag{5.89}$$

of a quaternion $q = (w, x, y, z) \in \mathbb{Q}$ is zero if and only if the fused yaw is zero. This comes about because the $\cos\frac{\alpha}{2}$ term is non-zero for $\alpha \in [0, \pi)$. Expressed in terms of mathematical notation, we have that

$$\psi = 0 \iff z = 0, \tag{5.90}$$

and consequently, from Equation (5.17b), that

$$\psi = 0 \iff e_z = 0. \tag{5.91}$$

This is a remarkable and not to be undervalued property of the fused yaw, as it asserts that the fused yaw is zero if and only if there is no component of rotation about the vertical z-axis. This, by contrast, is not true for any other definition of yaw. It can further be observed that the quaternion q_f corresponding to the fused yaw component of a rotation can be constructed by zeroing the x and y-components of the quaternion $q = (w, x, y, z)$ and renormalising. This leads to the equation

$$q_f = \frac{1}{\sqrt{w^2+z^2}}(w, 0, 0, z). \tag{5.92}$$

and like Equation (5.90) makes great intuitive sense, as we are essentially zeroing the x and y-components of the axis of rotation of q, leaving just the z-axis yaw rotation component. Once again, Equation (5.92) and this kind of thinking does not apply to any other definition of yaw. The ability to so easily and directly calculate q_f leads to one way of removing the fused yaw component of a quaternion—something that is a surprisingly common operation—using the expression

$$q_t = q_f^* q \tag{5.93a}$$

$$= \frac{1}{\sqrt{w^2+z^2}}\big(wq + z(z, y, -x, -w)\big), \tag{5.93b}$$

where q_f^* is the conjugate, and therefore inverse, of q_f (recall that q_f is a unit quaternion). As an alternative to Equation (5.93), the fused yaw can also be calculated directly using Equation (5.78) and manually removed using the fact that

$$q_f = q_z(\psi) = (\cos\tfrac{\psi}{2},\, 0,\, 0,\, \sin\tfrac{\psi}{2}). \tag{5.94}$$

Equations (5.92) and (5.93b) fail if and only if $w = z = 0$, which is precisely equivalent to $\alpha = \pi$, the fused yaw singularity.

5.6.2 Links Between Fused Angles and Euler Angles

Even though the interpretations of the variables are quite different, and the nature of the domains do not correspond, purely mathematically it can be observed from Equations (5.22b) and (5.86a) that the ZYX Euler pitch is equal to the fused pitch, and from Equation (5.86b) that the ZXY Euler roll is equal to the fused roll. That is,

$$\theta_E = \theta, \tag{5.95a}$$

$$\phi_{\widetilde{E}} = \phi. \tag{5.95b}$$

As such, fused angles can be seen to—with an adaptation of the domains and geometric interpretation—unite the ZYX Euler pitch and ZXY Euler roll with a novel and meaningful concept of yaw, to form a useful representation of rotations.

5.7 DISCUSSION

In this section, we discuss how the developed rotation representations relate to the requirements that were set out in Section 5.1, and provide application examples for them.

5.7.1 Rotation Representation Aims

As described in detail in Section 5.1, the work on 3D rotations in this chapter was motivated by the analysis and control of balancing bodies in 3D, and set out to develop new rotation representations, specifically the fused angles and tilt phase space representations, to address certain gaps in the field of existing rotation representations. A condensed summary of the aims that were listed for the new representations is as follows—the developed representations should:

Aim 1: Partition rotations into independent yaw and tilt rotation components that are then independently parameterised,

Aim 2: Quantify the amount of rotation in the **xy**, **xz** and **yz** major planes as scalar angular values that can meaningfully be referred to as 'yaw', 'pitch' and 'roll',

Aim 3: Define a notion of yaw that satisfies yaw additivity, i.e. global z-rotations should purely additively affect the yaw,

Aim 4: Define notions of pitch and roll that are concurrent, i.e. with no forced order of application that prioritises one of the two,

Aim 5: Have a tilt rotation component that encapsulates the heading-independent balance state of a robot, i.e. how far in any direction the robot is from being upright,

Aim 6: Have a tilt rotation component that has a direct correspondence to the set of possible accelerometer-measured gravity directions, and,

Aim 7: Given that $(\hat{\mathbf{e}}, \theta_a) \in \mathbb{A}$ is the axis-angle representation, have a tilt rotation component that has no e_z component, and a yaw rotation component that is purely a function of θ_a and e_z.

As discussed and demonstrated throughout this chapter, all of these aims have been addressed by the fused angles and tilt phase space representations. The notion of fused yaw for example, presented in Section 5.3.1, quantifies the amount of rotation in the **xy** plane (Aim 2), satisfies yaw additivity (Aim 3), and explicitly partitions rotations into independently parameterised yaw and tilt rotation components, as required by Aim 1. The fused pitch and roll (θ, ϕ), and phase roll and pitch (p_x, p_y), are also two different definitions of pitch and roll that are concurrent (Aim 4), as in each case there is no delineable order of rotations, and that quantify the amount of rotation in the **xz** and **yz** major planes. It should be noted that due to the required yaw additivity and yaw/tilt parameter independence, the major planes being referred to in all instances are the major planes of frame {A}, as defined in Section 5.3.1 and Figure 5.4. This makes intuitive sense, as {A} is like an untilted and upright heading-local frame, meaning that rotations in the $\mathbf{x}_A\mathbf{z}_A$ and $\mathbf{y}_A\mathbf{z}_A$ planes really do correspond to heading-local sagittal and lateral rotations, as desired. The described nature of {A} also justifies why the tilt rotation component from {A} to {B} is a heading-independent balance state (Aim 5). The reason why there is a direct correspondence (Aim 6) between tilt rotations and the set of possible gravity directions as measured by a quasi-static accelerometer, i.e. ${}^B\mathbf{z}_G$, was discussed in Section 5.3.2.1. The final remaining aim, Aim 7, can be seen to be satisfied by the definition of fused yaw and tilt, by consideration of Equations (5.17b), (5.27) and (5.61).

The main differences between the fused angles and tilt phase space representations are magnitude axisymmetry (discussed later in Section 6.2.4.3), the way that rotations in the negative fused hemisphere are handled, and the optional ability to represent unbounded rotations in the case of the tilt phase space. The tilt phase space also facilitates tilt vector addition, a commutative and unambiguous way of concurrently adding tilt rotations, which finds use in multiple practical scenarios.

5.7.2 Application Examples

The rotation representations developed in this chapter can be applied, with advantages, in many scenarios. Many uses of the representations are described throughout this thesis and the associated open source software releases, but as a general example, quadrotors for instance

need to tilt in the direction they wish to accelerate, so a smooth and concurrent way of representing such tilt in terms of pitch and roll, using a suitable mix of tilt angles, fused angles, and in particular the tilt phase space, can be of great benefit. Similar arguments for the use of these representations also apply to the scenarios of balance and bipedal locomotion, where the tilt rotation component is particularly relevant, because as previously mentioned, it encapsulates the entire heading-independent balance state of the robot, with no extra component of rotation about the z-axis. Indeed, due also to the parallel between tilt rotations and the values measured by accelerometer sensors (Aim 6 above), tilt rotations can be seen to be a natural and therefore practically useful split of orientations into yaw and tilt.

Figure 5.1 illustrates how for the application of bipedal walking the fused angles ψ, θ, ϕ and/or the tilt phase space parameters p_x, p_y, p_z can be identified as the 'roll', 'pitch' and 'yaw', and used to independently quantify the amounts of rotation in the lateral, sagittal and horizontal (i.e. heading) planes of walking, respectively. This allows the motion, stability and state of balance to be measured and controlled separately in each of these three major directions of walking. Examples of gait stabilisation schemes that work in this way include the one presented in Chapter 9, which operates largely independently in the sagittal and lateral planes based on the fused pitch and fused roll orientation values. Another example is the balance feedback controller presented in Chapter 11, which based on the many advantageous properties of the tilt phase space, constructs numerous pure tilt rotation feedback components. It is in the nature of feedback controllers, e.g. PID-style controllers, to produce control inputs of arbitrary magnitude based on a set of gains, so the unbounded nature of the tilt phase space allows this to be handled in a clean way to full effect. The combining of various tilt rotation feedback components is also naturally handled by the tilt phase space via tilt vector addition.

On a lower level, the representations developed in this chapter have also been used, for example, in Chapter 10 for the constraint-based generation of gait trajectories. Most notably, the tilt phase space is used to separate the yaw and tilt of the feet at each so-called gait keypoint, and then interpolate between them using a method of orientation cubic spline interpolation (more details in Allgeuer, 2020). This ensures that the yaw and tilt profiles are individually exactly as intended in the final 3D foot trajectories, especially seeing as the yaw profiles come from the commanded step sizes, and the tilts come separately from the feedback controller. Tilt vector addition is also required, because multiple feedback paths need to contribute to each final foot tilt. The summed foot tilts are not guaranteed to be in range though, so the unbounded nature of the tilt phase space helps in being robust to 'wrapping around' prior to foot tilt saturation. As such, the presented

work on rotations allows for the effective and robust generation of gait trajectories using such methods.

As a final note, it is reiterated that open source software libraries in both C++ (Allgeuer, 2018c) and Matlab (Allgeuer, 2018b) have been released to support the development and use of algorithms involving the tilt angles, fused angles and/or tilt phase space representations. Although these new representations are clearly the focus, it should be noted that the released libraries equally support all standard rotation representations as well, like the Euler angles, rotation matrix and quaternion representations.

6 WHY NOT EULER ANGLES?

Three new rotation representations, all based on a novel way of partitioning 3D rotations into yaw and tilt components, were introduced in the previous chapter. Detailed explanations were given as to why the representations are required, and what properties they are desired to have (see Section 5.1). A condensed list of aims was given in Section 5.7.1. Despite hints having been given as to why it is the case, one question has so far remained however—

Why are Euler angles not good enough for the job?

This chapter aims to clearly and resolutely answer this question by presenting a comparative analysis between Euler angles, fused angles and the tilt phase space, in specific reference to the motivations and aims that were provided in the previous chapter, and why Euler angles do not fulfil them. Further problems and illogicalities of Euler angles are also presented, in explicit contrast to how the situation is different for fused angles and the tilt phase space.

The reasons why the other existing rotation representations, like for example quaternions, do not address our rotation representation needs is relatively clear from Section 5.1, and has already been discussed previously, e.g. at the end of Section 5.2.3. It should be clearly noted however, that rotation matrices and quaternions are of course nonetheless *very* useful for all kinds of computations and algorithms in this thesis, including ones that involve the new representations. For instance, the attitude estimator in Chapter 7 is based on the fused yaw, but formulated in terms of rotation matrices and quaternions, and the final estimated orientation is expressed in terms of fused angles and/or the tilt phase space in order to be useful for the various gait feedback controllers that use it.

6.1 EULER ANGLES CONVENTIONS

This chapter relies heavily on the definitions and results that were presented in Chapter 5. To understand the ensuing comparative analysis, it is important, for example, to be familiar with Section 5.3, and in particular the definitions and properties of the selected two Euler angles conventions, given in Section 5.2.4. One should recall that due to the reverse equivalence between intrinsic and extrinsic Euler angles, the convention that the z-axis points 'upwards', and the desire to express rotation components about all three principal axes, the only

two relevant Euler angles conventions are the intrinsic ZYX and ZXY Euler angles conventions, given respectively by

$$\overset{G}{B}E = (\psi_E, \theta_E, \phi_E) \in (-\pi, \pi] \times [-\tfrac{\pi}{2}, \tfrac{\pi}{2}] \times (-\pi, \pi] \equiv \mathbb{E}, \tag{6.1a}$$

$$\overset{G}{B}\tilde{E} = (\psi_{\tilde{E}}, \phi_{\tilde{E}}, \theta_{\tilde{E}}) \in (-\pi, \pi] \times [-\tfrac{\pi}{2}, \tfrac{\pi}{2}] \times (-\pi, \pi] \equiv \tilde{\mathbb{E}}. \tag{6.1b}$$

All problems or results that are derived for one of these two conventions can analogously be derived for the other, so without loss of generality, in this chapter we will mainly focus on the ZYX Euler angles convention, and refer to it ubiquitously as *the* 'Euler angles' representation. If the ZXY convention is needed, then it will be referred to explicitly as the 'ZXY Euler angles', with the 'ZXY Euler yaw', 'ZXY Euler roll' and 'ZXY Euler pitch' components, and the associated *tilded* notation will be used, as given in Equation (6.1b).

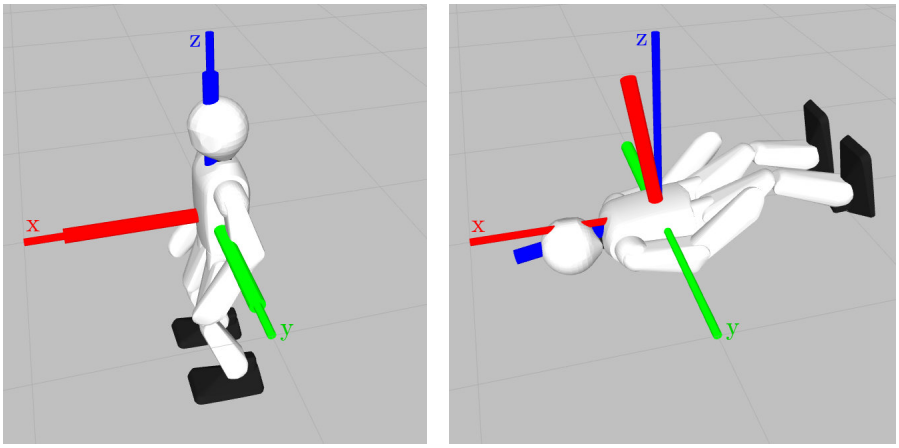
6.2 PROBLEMS WITH EULER ANGLES

There are many different problems with Euler angles, but before these are meticulously mathematically explored, here are some examples where one can intuitively tell[1] that something is not behaving as it really should:

1. Consider a humanoid robot where the body-fixed z-axis points up through its head and the x-axis points forwards out of its chest, as shown in Figure 6.1a. If the yaw of a rotation is zero, then one would expect that no significant z-rotation or change in heading occurs. Nonetheless, when for example combining a Euler yaw of $0°$ with a Euler pitch of $-80°$ and a Euler roll of $180°$, the resulting rotation leaves the robot lying on its back facing the opposite direction to the one it started in, as shown in Figure 6.1b. The fact that a large change in heading has occurred is supported by the axis-angle representation of the rotation, which is π radians about the vector $(0.766, 0, 0.643)$, which has a significant non-zero z-component.

2. Consider the same humanoid robot, with a pure pitch rotation of $\pm 75°$ applied to it, so that it is close to lying on its front or its back. If a humanoid robot falls over and tries to get up again, these are not unexpected orientations to occur. If the heading-independent balance state were to be defined as the orientation of the robot with its Euler yaw removed (as opposed to its fused yaw), then one can observe that this balance state would be very unstable in situations like the one described. If the robot rotates just $10°$ locally about its z-axis, the Euler yaw would change by $34.3°$. For pure pitch rotations of $\pm 85°$, this increases to changes in Euler yaw of $45.1°$ for rotations about the z-axis of just $5°$.

1 In consideration of Section 5.7.1



(a) Robot prior to rotation (b) After rotation by $E(0°, -80°, 180°)$

Figure 6.1: Consider the rotation of a robot by the ZYX Euler angles rotation $E = (0°, -80°, 180°)$. If the robot is initially upright and facing to the left, as shown in (a), then the final pose of the robot is lying on its back and facing to the right, as shown in (b). Although the ZYX Euler yaw of this rotation is zero, clearly a vast change in the intuitive understanding of 'heading' has occurred, demonstrating a clear shortcoming of the ZYX Euler angle representation.

This sensitivity is clearly not desired within the normal working ranges of the robot.

3. While it is clear by assumption that the z-axis points 'upwards', along the direction of gravity, the choice of whether the x and y-axes point 'forwards' or 'sidewards', or in another diagonal direction, is essentially arbitrary. Thus, the concepts of pitch and roll, and the parameters expressing them, should behave analogously to each other, just in 90° differing directions. This is not the case at all for the Euler pitch and Euler roll parameters, as they do not even share the same domain ($[-\frac{\pi}{2}, \frac{\pi}{2}]$ vs. $(-\pi, \pi]$).

4. For ZYX Euler angles, the x-axis about which the final x-rotation is performed has a zero y-component relative to the initial yaw-rotated intermediate frame (i.e. the supposed 'heading-local' frame), but a non-zero z-component. This means that the Euler yaw and roll actually both contribute to the final heading of the robot, which is not desired. In fact, at the extremes when the Euler pitch is $\pm\frac{\pi}{2}$, the Euler yaw and roll become completely interchangeable, as demonstrated by Equation (5.19).

Summarised into a more generalised and abstract list, the core failings of the Euler angles representations are as follows:

A) **Singularities and Local Parameter Sensitivities:**
The gimbal lock singularities are in close proximity to normal working ranges, making them hard to avoid, and leading to

unwanted artefacts due to the increased local parameter sensitivities in widened neighbourhoods of the singularities.

B) **Mutual Independence of Rotation Parameters:**
The Euler parameters have mutual interdependencies, leading to a mixed attribution of which parameters contribute to which major planes of rotation.

C) **Axisymmetry of Yaw:**
The definition of the Euler yaw depends implicitly on axis projection, leading to unintuitive non-axisymmetric behaviour of the yaw angle.

D) **Axisymmetry of Pitch and Roll:**
The definition of Euler pitch and roll requires a fundamental assumption of the order of elemental rotations, leading to non-axisymmetric definitions that do not correspond to each other in domain and/or behaviour.

Each of these core issues are investigated in detail in the following sections, with a specific focus on demonstrating why Euler angles have these problems, and fused angles and the tilt phase space do not.

Of the seven aims given in Section 5.7.1, Euler angles only satisfy two of them, Aim 3 and Aim 6. For instance, as described just above in Example 2, the problem of parameter sensitivities (Problem A) prevents the definition of a meaningful heading-independent balance state based on Euler pitch and roll, refuting Aim 5. The issues relating to axisymmetry (Problems C and D) also complicate or even violate Aims 2 and 4. The problem of parameter interdependencies (Problem B) further contribute to this, but also explain why Aims 1, 5, and 7 are not met.

6.2.1 Singularities and Local Parameter Sensitivities

We recall from Section 5.5 that it was shown by Stuelpnagel (1964) that it is topologically impossible to have a global one-to-one three-dimensional parameterisation of the rotation group without any singular points. That is, every three-dimensional parameterisation of the rotation space must have at least one of the following:

(i) A rotation that does not have a unique set of parameters,

(ii) A set of parameters that does not specify a unique rotation,

(iii) A rotation in the neighbourhood of which the sensitivity of the map from rotations to parameters is unbounded,

(iv) A set of parameters in the neighbourhood of which the sensitivity of the map from parameters to rotations is unbounded.

The Euler angles representation is singular at gimbal lock, i.e. when the Euler pitch $\theta_E = \frac{\pi}{2}$ or $-\frac{\pi}{2}$. Based in part on Equation (5.19), it can be seen that the Euler yaw ψ_E and Euler roll ϕ_E both have essential discontinuities there. In reference to the characterisation of singularities given above, the singularities at both gimbal lock scenarios are of type (i) and (iii), for both ψ_E and ϕ_E. It is critical to compare this to the case for fused angles and the tilt phase space, which both only have the one fused yaw singularity in the parameter ψ, which occurs for $\alpha = \pi$. Thus, these two representations have a single singularity in a single parameter that does not affect the heading-independent balance state, while Euler angles have two singularities in two parameters, one of which directly affects it. As such, fused angles and the tilt phase space can represent heading-local states of balance completely without singularities, while this is not the case for Euler angles.

The fused yaw singularity is also 'maximally far' from the identity rotation, requiring a 180° tilt rotation in any direction to get there, while the two Euler angle singularities are only 90° away, which is close to, if not in, normal working ranges. In fact, the increased parameter sensitivity of the Euler yaw and roll near gimbal lock has noticeable effects even for tilt rotations of only 65° for instance. Sudden sensitive changes in Euler yaw and roll can occur if even small changes in orientation are made, even though the resulting rotation remains essentially pure pitch in nature. Suppose we consider pure pitch rotations of up to 90°, and calculate the sensitivity of the Euler yaw and roll parameters relative to small infinitesimal rotations about the local body-fixed z-axis. The result is shown in Figure 6.2, along with similarly calculated sensitivities for the fused yaw, fused roll and phase roll parameters. While the fused yaw maintains a perfect sensitivity of 1 for all pitch angles and the fused roll and phase roll parameters slowly increase in sensitivity as the local z-rotation becomes closer to being a global x-rotation, the Euler yaw and roll can both be seen to have strongly divergent sensitivities that have noticeable effects for even moderate pitch rotations. Consequently, it can be seen that the Euler yaw component of a rotation cannot in general meaningfully be removed, as due to the highly sensitive Euler roll, even moderate tilts can experience large z-rotations in the rotation that remains, even though this should actually only be the contribution of 'pitch' and 'roll'.

6.2.2 Mutual Independence of Rotation Parameters

In order to be able to fulfil many of the aims set out in Section 5.7.1, one necessary implicit condition is that the individual rotation parameters need to be as mutually independent as possible, and correspond intuitively and uniquely to the x, y and z-components of rotation.

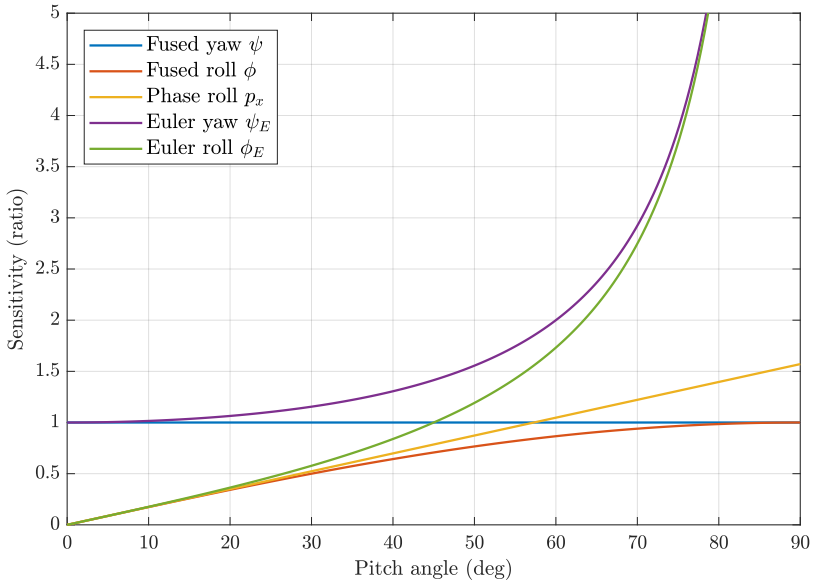


Figure 6.2: Plot of the sensitivities of the Euler angles, fused angles and tilt phase space parameters to infinitesimal local z-rotations at pure pitch orientations. Suppose $\beta \in [0, 90°]$ and we consider locally z-rotating (i.e. about its own z-axis) the pure pitch orientation $R_y(\beta)$. If the z-rotation is by some infinitesimal angle ϵ, then the sensitivity of a parameter X in describing $R_y(\beta)$ is given by $S_X(\beta) = |\frac{dX}{d\epsilon}|$. Clearly, high sensitivities as displayed by the Euler yaw and roll are undesirable, as for example, $S_{\phi_E}(\beta) = 3$ means that a small change in z-rotation causes a threefold change in the Euler roll ϕ_E. We can observe that the fused angles and tilt phase space parameters have reasonable sensitivities everywhere, and in particular that the fused yaw sensitivity is perfectly unit magnitude for all β.

This is *not* the case for Euler angles, as is shown in the following subsections by direct comparison to the fused angles and tilt phase space representations.

6.2.2.1 *Mutual Dependence of Yaw and Roll*

Consider the Euler angles rotation $E = (\psi_E, \theta_E, \phi_E) \in \mathbb{E}$ from a global frame {G} to the body-fixed frame {B}, consisting of a sequence of three rotations, namely about the z, y and x-rotations respectively. Let {E} denote the frame that results when {G} is initially rotated by the Euler yaw ψ_E. Ideally, as per Aim 5, {E} should be the reference frame for a Euler angles-based heading-independent balance state, i.e. the rotation component from {E} to {B}. One can quickly see however that this so-called balance state can actually have large z-rotation components in it, contravening that it is heading-independent in any intuitive sense. One obvious example of this has already been shown in Figure 6.1, where an example robot with zero Euler yaw can be observed to be intuitively facing the exact opposite direction to its

reference identity pose. If we consider the axis of rotation of the Euler roll parameter (the third of the three elemental Euler rotations) in terms of the (supposedly) heading-local frame {E}, we can see that it is given by

$$^{E}\mathbf{x}_{B} = (\cos\theta_{E},\ 0,\ -\sin\theta_{E}). \tag{6.2}$$

This is of course problematic, as the z-component is non-zero, meaning that the Euler roll rotation about this axis contributes to the heading of frame {B}. This effect can most clearly be seen when $|\sin\theta_{E}|$ is large, or in particular when

$$|\sin\theta_{E}| = 1 \iff \theta_{E} = \pm\tfrac{\pi}{2}, \tag{6.3}$$

which corresponds to gimbal lock, where Euler yaw and roll become completely interchangeable as given by Equation (5.19). The Euler pitch does not have this same problem, as relative to {E}, its axis of rotation is given by the pure

$$^{E}\mathbf{y}_{E} = (0, 1, 0). \tag{6.4}$$

Conceptually, what is happening is that part of the total 'yaw' of a rotation is always being quantified by the Euler roll parameter, in addition to the Euler yaw parameter, meaning that neither cleanly represents the component of rotation that they ideally should. The fused angles and tilt phase space representations do not have any such mutual dependencies, thanks in part to the fact that they rely on concurrent and not sequential definitions of pitch and roll. Importantly, this means that the heading-independent balance state defined by each representation (as expected) has no component of rotation about the z-axis, as indicated by Equation (5.36).

6.2.2.2 *Mutual Dependence of Pitch and Roll*

As the Euler pitch elemental rotation sequentially precedes the Euler roll one, the axis of rotation $^{E}\mathbf{x}_{B}$ of the latter is a function of θ_{E}, as explicitly given in Equation (6.2). This creates a dependency of ϕ_{E} on θ_{E}, which results in the Euler roll ϕ_{E} not completely capturing the intuitive sense of 'roll' all by itself. This can be seen in the bottom row of the rotation matrix in Equation (5.21), which corresponds to the z-vector

$$^{B}\mathbf{z}_{G} = (-\sin\theta_{E},\ \cos\theta_{E}\sin\phi_{E},\ \cos\theta_{E}\cos\phi_{E}). \tag{6.5}$$

The unit vector $^{B}\mathbf{z}_{G}$, as discussed in Section 5.1.2, is a heading-independent measure of the global 'up' direction, just like a quasi-static accelerometer would measure the direction of gravity. While the x-component is a pure function of θ_{E}, the y-component is not a function purely of ϕ_{E}, as would naturally be desired. It can be seen for example from Equation (5.68) however, that

$$^{B}\mathbf{z}_{G} = (-\sin\theta,\ \sin\phi,\ \cos\alpha), \tag{6.6}$$

so both of the aforementioned properties hold for fused angles. The situation is slightly more complicated for the tilt phase space, but from Equation (5.72), seeing as the phase roll and pitch parameters are just a rescaling of the sines of the fused angles parameters, an essentially similar result applies.

6.2.2.3 *Purity of the Axis of Rotation*

As discussed in Section 5.2.2, every 3D rotation can be expressed as a single rotation by some angle $\theta_a \in [0, \pi]$ about an axis $\hat{\mathbf{e}} = (e_x, e_y, e_z)$ in 3D space. We start by recalling from Equation (5.91) that the fused yaw is zero if and only if there is no component of rotation about the vertical z-axis, i.e.

$$\psi = 0 \iff e_z = 0, \tag{6.7}$$

and observe from Equations (5.17b) and (5.92) that the fused yaw component of rotation q_f is given by the renormalisation of

$$\tilde{q}_f = \left(\cos\tfrac{\theta_a}{2},\, 0,\, 0,\, e_z \sin\tfrac{\theta_a}{2}\right). \tag{6.8}$$

It can consequently be seen that the fused angles and tilt phase space representations both satisfy Aim 7 (see Section 5.7.1). Conceptually, the interpretation of this result is that the notion of yaw used in both representations is intricately and purely linked to the z-component of the axis of rotation, in such a way that the former cleanly parameterises the latter. This also means that the remaining two pitch and roll parameters in each representation together cleanly parameterise the remaining two x and y-components of the axis of rotation. Neither result is true for the Euler angles representation, where for example the Euler angles rotation given in Example 1 on page 82 demonstrates that rotations free of Euler yaw can still have significant non-zero z-rotation components.

For each representation, we now look more carefully at the purity of the pitch and roll parameters in representing the x and y-components of the axis of rotation. Due to yaw additivity (Aim 3) and the desire for yaw/tilt parameter independence (Aim 1), it only makes sense to look at these components relative to the respective heading-local yaw-rotated intermediate frame, namely

- Frame {A} for the fused angles and tilt phase space representations, given by a yaw rotation of {G} by the fused yaw ψ,
- Frame {E} for the ZYX Euler angles representation, given by a yaw rotation of {G} by the ZYX Euler yaw ψ_E, and,
- Frame {$\tilde{\text{E}}$} for the ZXY Euler angles representation, given by a yaw rotation of {G} by the ZXY Euler yaw $\psi_{\tilde{E}}$.

This conforms with our previously stated expectation (Aim 2) that the pitch and roll values should quantify the amount of rotation in the **xz** and **yz** major planes of exactly these frames, respectively. Thus, we only need to examine and compare, for each representation, the nature of the axis of rotation for rotations that have zero yaw in that representation. Furthermore, as only the signed direction of the axis of rotation matters for the purpose of this discussion, we can consider any non-unit vector $\tilde{\mathbf{e}} = (\tilde{e}_x, \tilde{e}_y, \tilde{e}_z)$ defining that ray.

For fused angles and tilt phase space rotations with zero fused yaw, it can be deduced from Equations (5.36), (5.44) and (5.62) that the axes of rotation are respectively given by

$$\tilde{\mathbf{e}}_F = (\sin\phi, \sin\theta, 0), \tag{6.9a}$$
$$\tilde{\mathbf{e}}_P = (p_x, p_y, 0). \tag{6.9b}$$

On the other hand, from Equation (5.23), for ZYX Euler angles rotations with zero ZYX Euler yaw, and ZXY Euler angles rotations with zero ZXY Euler yaw, the axes of rotation are respectively given by

$$\tilde{\mathbf{e}}_E = (s_{\bar{\phi}_E} c_{\bar{\theta}_E}, c_{\bar{\phi}_E} s_{\bar{\theta}_E}, -s_{\bar{\phi}_E} s_{\bar{\theta}_E}), \tag{6.10a}$$
$$\tilde{\mathbf{e}}_{\tilde{E}} = (s_{\bar{\phi}_{\tilde{E}}} c_{\bar{\theta}_{\tilde{E}}}, c_{\bar{\phi}_{\tilde{E}}} s_{\bar{\theta}_{\tilde{E}}}, s_{\bar{\phi}_{\tilde{E}}} s_{\bar{\theta}_{\tilde{E}}}). \tag{6.10b}$$

For fused angles it can be seen that there is no component of rotation about the x and y-axes exactly when the fused roll ϕ and fused pitch θ are zero, respectively, and for the tilt phase space it can be seen that this occurs exactly when the phase roll p_x and phase pitch p_y are zero, respectively. For ZYX Euler angles however, the y-component is also zero when $\phi_E = \pi$, and for ZXY Euler angles, the x-component is also zero when $\theta_{\tilde{E}} = \pi$. This comes about because the $\tilde{e}_x$ and $\tilde{e}_y$ components are mixed expressions of Euler pitch and roll, instead of clean independent expressions like for fused angles and the tilt phase space, where direct one-to-one associations can be made between

$$\tilde{e}_x \longleftrightarrow \phi, p_x \tag{6.11a}$$
$$\tilde{e}_y \longleftrightarrow \theta, p_y \tag{6.11b}$$

It is also evident from the

$$\tilde{e}_z = \pm s_{\bar{\phi}_E} s_{\bar{\theta}_E} \tag{6.12}$$

components in Equation (6.10) that the Euler pitch and roll together contribute a component of rotation about the z-axis, as previously mentioned, which is unintuitive and indicates an impure contribution to the axis of rotation. In fact, seeing as Equation (6.10a) essentially encodes an arbitrary y-rotation followed by an x-rotation, and Equation (6.10b) encodes the exact reverse, it can be interpreted that the non-commutativity of these operations manifests itself in the opposite sign that results in the z-component $\tilde{e}_z$. Thus, as $\tilde{e}_z = 0$ for fused

angles and the tilt phase space, these two new representations can conceptually be thought of as a concurrent way of combining x and y-rotations in a symmetrical and neutral way, somewhere exactly in between choosing the x-rotation to go first and choosing the y-rotation to go first.

6.2.3 Axisymmetry of Yaw

By convention (and without loss of generality), in this thesis the z-axis is chosen to point in the direction opposite to gravity. This, amongst other things, ensures that the concepts of 'roll', 'pitch' and in particular 'yaw', line up with what one would intuitively expect. The fixed choice of z-axis, however, still leaves one degree of rotational freedom open for the choice of the directions of the global x and y-axes, where we recall that these must obviously lie in the plane perpendicular to the z-axis, be perpendicular to each other, and satisfy the right-hand rule. No one choice is 'right' or 'wrong'—they are all equally valid and mathematically correct definitions of the global reference frame—so one would desire that any analysis of the orientation or balance of a body gives analogous results no matter which one is chosen. In the context of this chapter, the concept of parameter axisymmetry refers to the property that one or more rotation parameters are either:

(a) Invariant to the freedom of choice of x and y-axis, or,

(b) Vary in an intuitive rotational manner proportional to the choice.

In other words, axisymmetry refers to the notion that the rotation parameters, in order to be self-consistent, should be symmetrical about the unambiguously defined z-axis. This is a relatively natural property to desire, as, for example, the amount of yaw a rotation has should clearly transcend any arbitrary choice of which reference frame to use for analysis.

The fused yaw parameter is axisymmetric in the sense that it is invariant to the choice of global x and y-axes, i.e. type (a) axisymmetry. Consider a robot that is upright, and thereby considered to be in its identity orientation relative to the environment. If the robot undergoes any rotation, the above statement of fused yaw axisymmetry asserts that the fused yaw of this rotation is the same no matter what choice of reference global x and y-axis is made to numerically quantify the rotation. This is an important and reassuring property of the fused yaw as, given that the z-axis is unambiguously defined, any concept of yaw about the z-axis should clearly be a property of the actual physical rotation, not a property of some arbitrary choice of reference frame made solely for the purpose of mathematical analysis. This is not the case for Euler yaw however, as can easily be demonstrated as follows. Suppose we have a robot standing upright relative to the

well-defined global z-axis, and suppose we define two different global reference frames:

- Frame $\{G_1\}$, such that relative to the identity pose of the robot the z-axis points upwards, the x-axis points forwards, and the y-axis points leftwards, and,
- Frame $\{G_2\}$, such that relative to the identity pose of the robot the z-axis points upwards, the y-axis points forwards and the x-axis points rightwards.

Both $\{G_1\}$ and $\{G_2\}$ are perfectly valid choices of reference frames that are consistent with the gravity-defined z-direction. If the robot now performs a 180° rotation about the horizontal axis 45° in between forwards and leftwards, the rotation quantified in terms of $\{G_1\}$ has a Euler yaw of $+90°$, but the rotation quantified in terms of $\{G_2\}$ has a Euler yaw of $-90°$, which is completely contradictory. This should not be, however, as the robot in both cases is performing exactly the same rotation relative to its environment. In terms of fused yaw, both quantifications of the rotation, i.e. relative to $\{G_1\}$ and relative to $\{G_2\}$, have a yaw of zero.

6.2.3.1 *Mathematical Model of Yaw Axisymmetry*

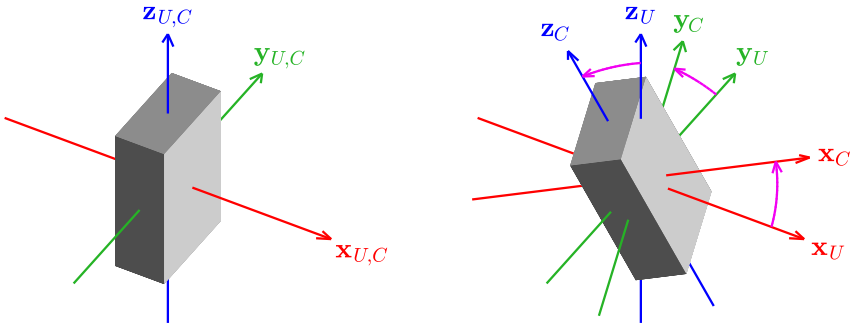
We now develop a mathematical model of yaw axisymmetry and demonstrate that the fused yaw satisfies it, while the Euler yaw does not. Consider once again a robot that is upright relative to its environment, and that is in its identity orientation. Let {U} be a global coordinate frame such that $\mathbf{z}_U$ points in the direction opposite to gravity, as required, and suppose that a rotation is undergone by the robot that is numerically given by ${}^{U}_{C}R$, where {C} is a body-fixed frame that coincides with {U} prior to rotation. This is a fixed physical rotation of the robot relative to its environment, so it should have a unique well-defined fused yaw according to axisymmetry.

As the z-axis is uniquely determined by the direction of gravity, every valid global coordinate system {G} that can be used as a reference frame to quantify ${}^{U}_{C}R$, including {U} itself, is a pure z-rotation of {U}. That is, given any valid global coordinate system {G},

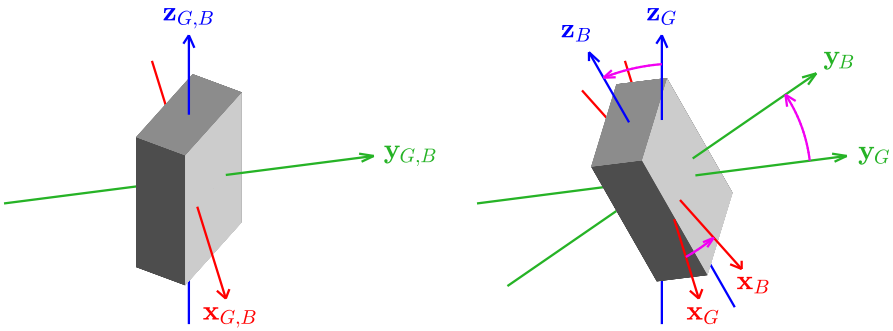
$$ {}^{G}_{U}R = R_z(\beta), \tag{6.13a} $$

$$ {}^{U}_{G}R = R_z(-\beta), \tag{6.13b} $$

for some angle $\beta \in (-\pi, \pi]$. Given a frame {G}, a body-fixed frame {B} is attached to the robot in such a way that it coincides with {G} when the robot is initially upright, but rotates with the robot, as shown in Figure 6.3. Thus, the physical rotation undergone by the robot maps frames {U} to {C}, and {G} to {B}. As such, the rotation matrices ${}^{U}_{C}R$ and ${}^{G}_{B}R$ are simply two different ways of quantifying the exact same rotation, just with a different reference frame.



(a) Left: Prior to rotation the global frame {U} coincides with the local frame {C}, Right: Frame {C} rotates with the body, and quantifies the physical rotation relative to {U}.



(b) Left: Prior to rotation the global frame {G} coincides with the local frame {B}, Right: Frame {B} rotates with the body, and quantifies the physical rotation relative to {G}.

Figure 6.3: Definition of frames for the investigation of parameter axisymmetry. Consider a body (represented here by a box) undergoing any physical rotation. The physical rotation can be modelled (a) as the rotation from {U} to {C}, or equivalently (b) as the rotation from {G} to {B}, where $\mathbf{z}_U \equiv \mathbf{z}_G$. Thus, ${}^{U}_{C}R$ and ${}^{G}_{B}R$ both numerically quantify the same physical rotation, despite being different rotation matrices. The axisymmetry property of fused yaw asserts that irrespective of the choice of {G}, the fused yaws of ${}^{U}_{C}R$ and ${}^{G}_{B}R$ are the same. As a result, every single physical rotation can unambiguously be assigned a fused yaw (irrespective of reference frame), which by contrast is not possible for Euler yaw.

By definition, the rotation ${}^{G}_{B}R$ maps {U} onto {C}, so

$$ {}^{G}_{B}R = {}^{G}_{U}R \, {}^{U}_{C}R \, {}^{U}_{G}R. \tag{6.14} $$

This equation can be understood directly from the theory of *referenced rotations* (Allgeuer, 2020), or from the observation that given any vector relative to {G}, applying the rotation ${}^{G}_{B}R$ is equivalent to transforming the vector to {U} coordinates using ${}^{U}_{G}R$, applying the rotation ${}^{U}_{C}R$ from {U} to {C}, and then transforming the result back to {G} coordinates using ${}^{G}_{U}R$. Consequently, from Equation (6.13),

$$ {}^{G}_{B}R = R_z(\beta) \, {}^{U}_{C}R \, R_z(-\beta). \tag{6.15} $$

Taking the fused yaw $\Psi(\cdot)$ of both sides of Equation (6.15) gives

$$ \begin{aligned} \Psi\left({}^{G}_{B}R\right) &= \Psi\left(R_z(\beta) \, {}^{U}_{C}R \, R_z(-\beta)\right) \\ &= \operatorname{wrap}\left(\beta + \Psi\left({}^{U}_{C}R \, R_z(-\beta)\right)\right) \\ &= \operatorname{wrap}\left(\beta + \Psi\left({}^{U}_{C}R\right) - \beta\right) \\ &= \Psi\left({}^{U}_{C}R\right). \end{aligned} \tag{6.16} $$

We note that $\Psi\left({}^{U}_{C}R\right)$ is clearly independent of β and the choice of {G}, so $\Psi\left({}^{G}_{B}R\right)$ must be independent as required. This demonstrates that the fused yaw of the physical rotation of the robot is invariant to the choice of reference global x and y-axis, as required, and therefore that the fused yaw satisfies type (a) parameter axisymmetry.

At this point, it can be demonstrated once again that Euler yaw violates parameter axisymmetry by considering, for example,

$$ \begin{aligned} {}^{U}_{C}R &= R_x(\tfrac{3\pi}{4}), \\ \beta &= \tfrac{\pi}{2}. \end{aligned} $$

The Euler yaw of ${}^{U}_{C}R$ is clearly zero, as it consists of just a single counterclockwise (CCW) x-rotation by $\frac{3\pi}{4}$ radians, but from Equation (6.15),

$$ \begin{aligned} {}^{G}_{B}R &= R_z(\beta) \, {}^{U}_{C}R \, R_z(-\beta) \\ &= R_z(\tfrac{\pi}{2}) \, R_x(\tfrac{3\pi}{4}) \, R_z(-\tfrac{\pi}{2}) \\ &= \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}. \end{aligned} $$

We therefore conclude that

$$ \begin{aligned} {}^{G}_{B}R &= R_y(\tfrac{3\pi}{4}) \\ &= E_R(\pi, \tfrac{\pi}{4}, \pi), \end{aligned} $$

where $E_R(\cdot, \cdot, \cdot)$ is notation for the rotation matrix corresponding to the enclosed Euler angles parameters. The Euler yaw of ${}^{G}_{B}R$ is thus

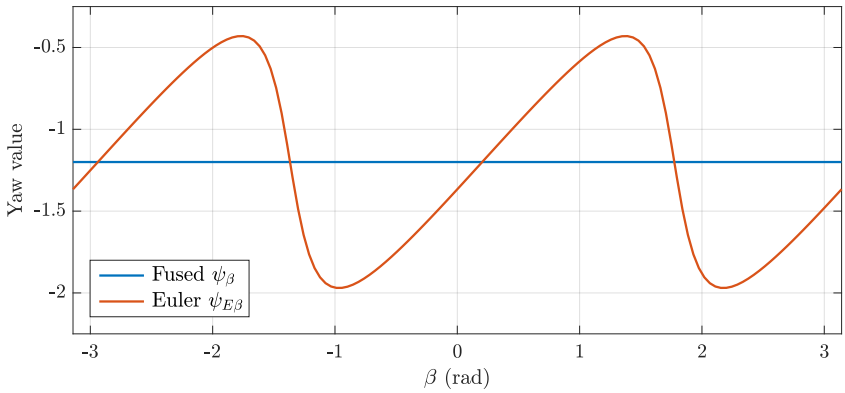


Figure 6.4: Plots of fused yaw and Euler yaw against β for the determination of parameter axisymmetry. A β of zero corresponds to the chosen physical rotation of ${}^{U}_{C}R = F_R(-1.2, 0.2, -1.3, 1)$. In contrast to the irregular nature of the Euler yaw, the constant nature of the fused yaw exemplifies its type (a) axisymmetry. That is, no matter what valid global reference frame is chosen to quantify ${}^{U}_{C}R$, the fused yaw of the rotation is the same.

π radians, which is totally different to the (zero) Euler yaw of ${}^{U}_{C}R$, despite the fact that both rotation matrices numerically quantify the same physical rotation. This proves that the Euler yaw cannot be axisymmetric.

6.2.3.2 *Visualising Yaw Axisymmetry*

As can be observed from Equation (6.13), the set of all possible choices of global reference x and y-axes corresponds directly to the set of all possible choices of β, which is in the range $(-\pi, \pi]$. Thus, given a particular physical rotation of the robot, one can plot the yaw that the rotation has relative to all possible valid global reference frames, by plotting the yaw it has for all possible values of β. This has been done, for example, in Figure 6.4 for the physical rotation ${}^{U}_{C}R = F_R(-1.2, 0.2, -1.3, 1)$, where $F_R(\cdot, \cdot, \cdot, \cdot)$ is notation for the rotation matrix corresponding to the enclosed fused angles parameters. It can clearly be seen that while the fused yaw remains constant for all β, the Euler yaw varies greatly and quite irregularly. In fact, if the same physical rotation but with $h = -1$ had been chosen, the Euler yaw would be seen to take on all possible values from $-\pi$ to π as β varies.

A similar observation can be made in Figure 6.5a, where surface plots of the Euler yaw and fused yaw parameters are provided for the case that an upright robot is rotated away from the vertical z-axis in every possible direction, about the vectors in the **xy** plane. Put into other words, Figure 6.5a plots, as surfaces, the Euler yaw and fused yaw parameters for all tilt rotations up to $\frac{\pi}{2}$ radians in magnitude. The x and y-coordinates of the surfaces are the x and y-components

of the rotation vector representations of the respective tilt rotations. While the fused yaw demonstrates axisymmetric, and in fact constant, behaviour about the gravity-defined z-axis, the Euler yaw behaves differently depending on the direction in which the robot is tilted away from the z-axis. While requiring the surface plots in Figure 6.5a to be rotationally symmetric about the z-axis may at first seem like a different definition of axisymmetry than the one used so far, one can quickly see that plotting the yaw parameters of all tilt rotations (of a particular magnitude) with respect to a single global reference frame is in fact mathematically equivalent to plotting the yaw parameters of a single (physical) tilt rotation with respect to all possible global reference frames. This comes about because both the set of all tilt rotations of a particular magnitude, and the set of all global reference frames, are generated by z-rotational changes of basis of any one single element.

6.2.4 Axisymmetry of Pitch and Roll

To investigate the axisymmetry of pitch and roll for the Euler angles, fused angles and tilt phase space representations, we consider the same situation as described in Section 6.2.3.1, and illustrated in Figure 6.3. That is, suppose an upright robot undergoes a physical rotation ${}^{U}_{C}R$ relative to one particular global reference frame {U}, and that {G} is any other global reference frame such that ${}^{G}_{B}R$ models the exact same rotation. As the z-axes of both {U} and {G} are constrained by convention to point in the same opposite direction to gravity, the two frames are just separated by a single z-rotation $R_z(\beta)$, so one can write

$$ {}^{G}_{U}R = R_z(\beta), \tag{6.17} $$

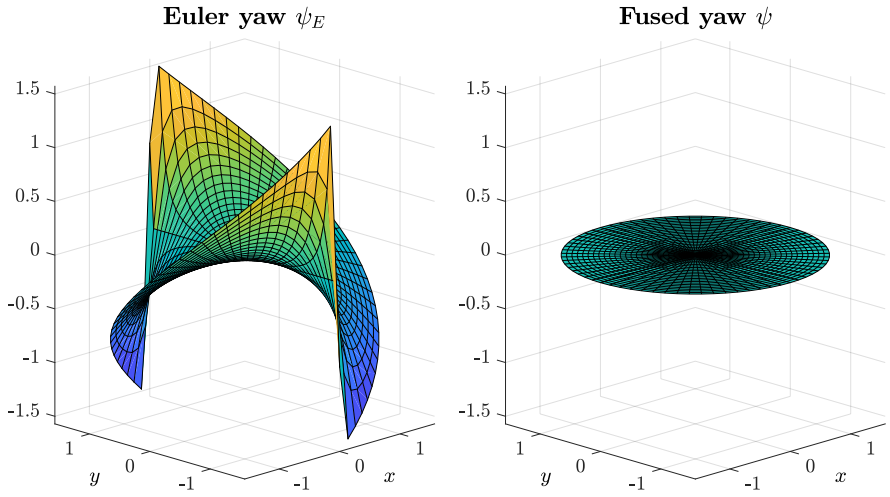
where β is a scalar angle in the range $(-\pi, \pi]$. Given this mathematical framework, the definition of parameter axisymmetry (see page 90) for any arbitrary rotation parameter can be restated as the property that the parameter, when calculated of ${}^{G}_{B}R$, is either:

(a) Completely independent of β, or,

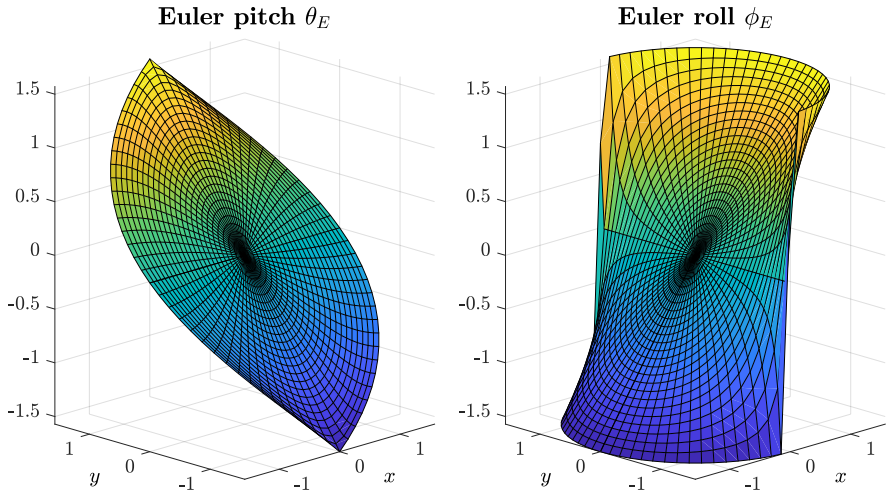
(b) Varies in a rotational manner with respect to β.

We start by introducing the following notation for the Euler angles, tilt angles, fused angles and tilt phase space representations of the fixed physical rotation ${}^{U}_{C}R$:

$$ {}^{U}_{C}E = (\psi_{E0}, \theta_{E0}, \phi_{E0}), \tag{6.18a} $$
$$ {}^{U}_{C}T = (\psi_0, \gamma_0, \alpha_0), \tag{6.18b} $$
$$ {}^{U}_{C}F = (\psi_0, \theta_0, \phi_0, h_0), \tag{6.18c} $$
$$ {}^{U}_{C}P = (p_{x0}, p_{y0}, \psi_0), \tag{6.18d} $$

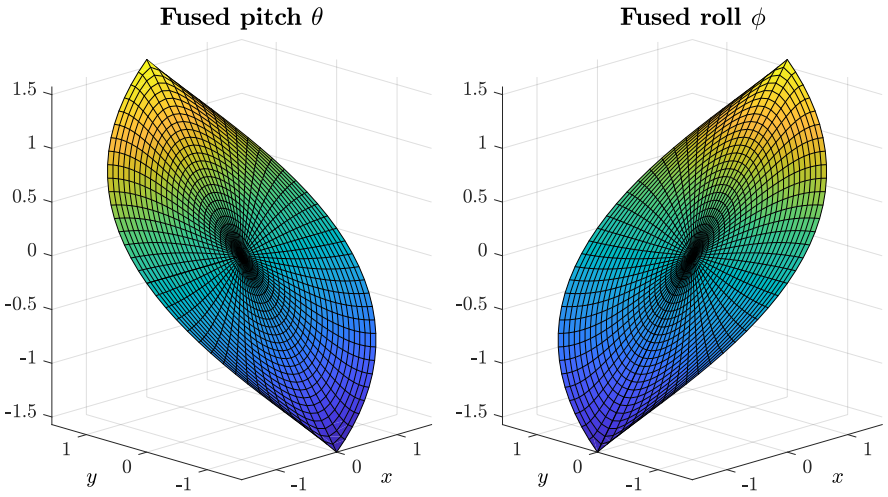


(a) 3D plots of Euler yaw and fused yaw relative to $(x, y) = (\alpha c_\gamma, \alpha s_\gamma)$.

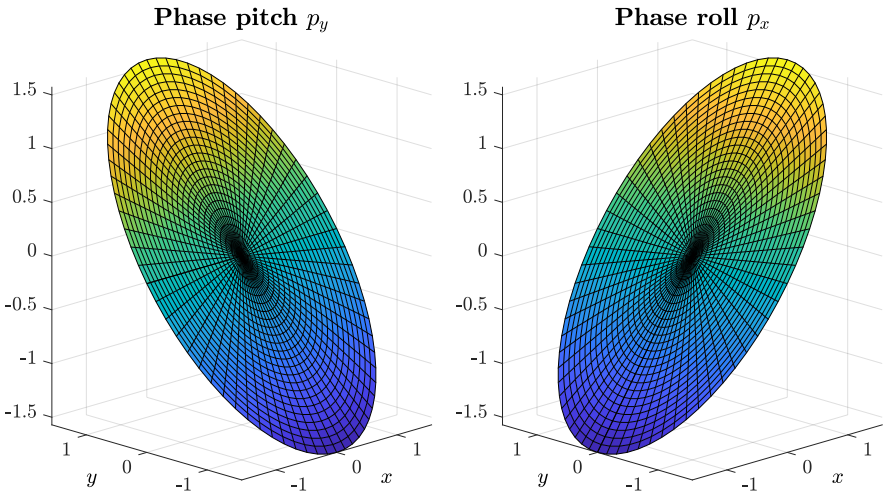


(b) 3D plots of Euler pitch and roll relative to $(x, y) = (\alpha c_\gamma, \alpha s_\gamma)$.

Figure 6.5: 3D plots of the Euler angles, fused angles and tilt phase space parameters of all pure tilt rotations up to $\frac{\pi}{2}$ in magnitude (continued in Figure 6.6). The parameters are plotted relative to the rotation vector coordinates of the associated tilt rotations, namely $(x, y) = (\alpha \cos\gamma,\ \alpha \sin\gamma)$. Although it may at first seem unrelated to parameter axisymmetry, plotting all tilt rotations of a particular magnitude α is equivalent to plotting any one such tilt rotation relative to all different possible global reference frames (i.e. all different β). Thus, for instance the rotational symmetry seen in the fused yaw plot is indicative of its type (a) axisymmetry (see page 90), while the lack of rotational symmetry in the Euler yaw plot is indicative of its lack of axisymmetry. The Euler pitch and Euler roll plots behave similarly in nature for small $\|(x, y)\| = \alpha$, but as the tilt rotations get larger, the Euler pitch and roll start to behave completely differently. For values of α beyond $\frac{\pi}{2}$ (beyond the pictured domain), $|\phi_E|$ even starts to exceed $\frac{\pi}{2}$ and takes on values up to π. This does not happen for the Euler pitch at all.



(a) 3D plots of fused pitch and roll relative to $(x, y) = (\alpha c_\gamma, \alpha s_\gamma)$.



(b) 3D plots of phase pitch and roll relative to $(x, y) = (\alpha c_\gamma, \alpha s_\gamma)$.

Figure 6.6: Continuation of Figure 6.5. In contrast to the lack of correspondence in behaviour between the Euler pitch and roll parameters (see Figure 6.5b), the fused pitch and roll (and phase pitch and roll) parameters do clearly visibly behave identically to each other. The perfect planar circular discs that can be seen in (b) for the tilt phase space are indicative of its type (b) axisymmetry (see page 90). Refer to Visualisation C in Section 6.2.4.5 for further discussion of this figure.

and similarly define the following notation for the same various representations of ${}^{G}_{B}R$:

$$ {}^{G}_{B}E = (\psi_{E\beta}, \theta_{E\beta}, \phi_{E\beta}), \tag{6.19a} $$

$$ {}^{G}_{B}T = (\psi_{\beta}, \gamma_{\beta}, \alpha_{\beta}), \tag{6.19b} $$

$$ {}^{G}_{B}F = (\psi_{\beta}, \theta_{\beta}, \phi_{\beta}, h_{\beta}), \tag{6.19c} $$

$$ {}^{G}_{B}P = (p_{x\beta}, p_{y\beta}, \psi_{\beta}). \tag{6.19d} $$

Given this notation, the result, for example, that the tilt angle parameter α is type (a) axisymmetric can be proven by demonstrating that α_{β} is independent of β, i.e. that it is a function only of the 'zero' variables $*_0$ in Equation (6.18). We already know from Section 6.2.3.1 that the fused yaw is invariant to the choice of β, so simply restating Equation (6.16) in terms of the new notation gives our first result:

$$ \psi_{\beta} = \psi_0. \tag{6.20} $$

This is the mathematical embodiment of the type (a) axisymmetry of the fused yaw parameter. We now continue through all the other representations and rotation parameters, and demonstrate the type (a) or type (b) axisymmetry of each, except for the Euler angles parameters, which are demonstrated to be non-axisymmetric. The previous result, that the Euler yaw is *not* axisymmetric, can be mathematically inferred from the fact that, up to angle wrapping,

$$ \psi_{E\beta} = \psi_{E0} + \operatorname{atan2}(B, A), \tag{6.21} $$

where

$$ A = c^2_{\beta} c_{\theta_E} + s^2_{\beta} c_{\phi_E} - c_{\beta} s_{\beta} s_{\theta_E} s_{\phi_E}, \tag{6.22a} $$

$$ B = c_{\beta} s_{\beta} (c_{\theta_E} - c_{\phi_E}) - s^2_{\beta} s_{\theta_E} s_{\phi_E}. \tag{6.22b} $$

Clearly, the right-hand side of Equation (6.21) is a function of β.

6.2.4.1 *Axisymmetry of Tilt Angles*

From Equation (6.18b) and the conversion equation from tilt angles to rotation matrices given in Equation (5.58), we have that

$$ {}^{U}_{C}R = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ -s_{\alpha_0} s_{\gamma_0} & s_{\alpha_0} c_{\gamma_0} & c_{\alpha_0} \end{bmatrix}, \tag{6.23} $$

where the dotted entries have been omitted for brevity. Similarly, from Equation (6.19b), we have that

$$ {}^{G}_{B}R = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ -s_{\alpha_{\beta}} s_{\gamma_{\beta}} & s_{\alpha_{\beta}} c_{\gamma_{\beta}} & c_{\alpha_{\beta}} \end{bmatrix}. \tag{6.24} $$

Based on Equation (6.15) however, we also know that

$$\begin{aligned}
{}^{G}_{B}R &= R_z(\beta)\,{}^{U}_{C}R\,R_z(-\beta) \\
&= \begin{bmatrix} c_\beta & -s_\beta & 0 \\ s_\beta & c_\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ -s_{\alpha_0}s_{\gamma_0} & s_{\alpha_0}c_{\gamma_0} & c_{\alpha_0} \end{bmatrix} \begin{bmatrix} c_\beta & s_\beta & 0 \\ -s_\beta & c_\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} & \cdot & & \cdot & \cdot \\ & \cdot & & \cdot & \cdot \\ -c_\beta s_{\alpha_0} s_{\gamma_0} - s_\beta s_{\alpha_0} c_{\gamma_0} & & c_\beta s_{\alpha_0} c_{\gamma_0} - s_\beta s_{\alpha_0} s_{\gamma_0} & & c_{\alpha_0} \end{bmatrix}.
\end{aligned} \tag{6.25}$$

So by comparing entries in Equations (6.24) and (6.25), we know that

$$s_{\alpha_\beta} s_{\gamma_\beta} = s_{\alpha_0}(c_\beta s_{\gamma_0} + s_\beta c_{\gamma_0}), \tag{6.26a}$$

$$s_{\alpha_\beta} c_{\gamma_\beta} = s_{\alpha_0}(c_\beta c_{\gamma_0} - s_\beta s_{\gamma_0}), \tag{6.26b}$$

$$c_{\alpha_\beta} = c_{\alpha_0}. \tag{6.26c}$$

As the $\cos(\cdot)$ function is one-to-one on the domain $[0, \pi]$, by direct consequence of Equation (6.26c),

$$\alpha_\beta = \alpha_0, \tag{6.27}$$

which demonstrates that the tilt angle α is type (a) axisymmetric. This insight also allows Equations (6.26a) and (6.26b) to be simplified to

$$\begin{bmatrix} \cos\gamma_\beta \\ \sin\gamma_\beta \end{bmatrix} = \begin{bmatrix} c_\beta & -s_\beta \\ s_\beta & c_\beta \end{bmatrix} \begin{bmatrix} \cos\gamma_0 \\ \sin\gamma_0 \end{bmatrix}. \tag{6.28}$$

By identifying the middle matrix as a 2D rotation matrix that rotates a vector CCW by an angle of β, one can deduce that, up to angle wrapping,

$$\gamma_\beta = \gamma_0 + \beta. \tag{6.29}$$

This expression, in particular in combination with Equation (6.28), is the mathematical embodiment of the type (b) axisymmetry of the tilt axis angle parameter γ, as it can clearly be seen that γ varies in an intuitive rotational manner with respect to β. Thus, it can be concluded that all three tilt angles parameters are axisymmetric.

6.2.4.2 *Axisymmetry of Fused Angles*

The fused pitch and roll are axisymmetric in the sense that their sine ratios, $\sin\theta$ and $\sin\phi$, circumscribe a uniform circle as a function of the choice of global reference x and y-axes, i.e. as a function of β. That is, the locus of $(\sin\phi, \sin\theta)$ over all possible choices of β is a circle, and this circle is traversed uniformly as the choice varies. This corresponds to type (b) axisymmetry, and as demonstrated later, Euler angles do not satisfy this property.

Similar to Equations (6.23) and (6.24), the rotation matrices ${}^{U}_{C}R$ and ${}^{G}_{B}R$ can, from Equation (5.67), be written as

$$ {}^{U}_{C}R = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ -s_{\theta_0} & s_{\phi_0} & c_{\alpha_0} \end{bmatrix}, \tag{6.30a} $$

$$ {}^{G}_{B}R = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ -s_{\theta_\beta} & s_{\phi_\beta} & c_{\alpha_\beta} \end{bmatrix}. \tag{6.30b} $$

From Equation (6.15) however, ${}^{G}_{B}R$ can alternatively be written as

$$ \begin{aligned} {}^{G}_{B}R &= R_z(\beta)\, {}^{U}_{C}R\, R_z(-\beta) \\ &= \begin{bmatrix} c_\beta & -s_\beta & 0 \\ s_\beta & c_\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ -s_{\theta_0} & s_{\phi_0} & c_{\alpha_0} \end{bmatrix} \begin{bmatrix} c_\beta & s_\beta & 0 \\ -s_\beta & c_\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ -c_\beta s_{\theta_0} - s_\beta s_{\phi_0} & c_\beta s_{\phi_0} - s_\beta s_{\theta_0} & c_{\alpha_0} \end{bmatrix}. \end{aligned} \tag{6.31} $$

Comparing Equations (6.30b) and (6.31) yields

$$ s_{\phi_\beta} = c_\beta s_{\phi_0} - s_\beta s_{\theta_0}, \tag{6.32a} $$

$$ s_{\theta_\beta} = c_\beta s_{\theta_0} + s_\beta s_{\phi_0}, \tag{6.32b} $$

which leads to the matrix equation

$$ \begin{bmatrix} \sin\phi_\beta \\ \sin\theta_\beta \end{bmatrix} = \begin{bmatrix} c_\beta & -s_\beta \\ s_\beta & c_\beta \end{bmatrix} \begin{bmatrix} \sin\phi_0 \\ \sin\theta_0 \end{bmatrix}. \tag{6.33} $$

By once again identifying the middle matrix as a 2D rotation matrix that rotates a vector CCW by an angle of β, one can see that this equation is *exactly* the mathematical statement of the type (b) axisymmetry of fused pitch and roll. Furthermore, seeing as the fused hemisphere parameter h is just the sign of $\cos\alpha$, from Equation (6.26c) we can deduce that

$$ h_\beta = h_0, \tag{6.34} $$

i.e. that the fused hemisphere parameter is type (a) axisymmetric. Thus, it can be concluded that all four fused angles parameters, including of course the fused yaw, are axisymmetric.

The axisymmetry of the fused pitch and roll can be interpreted and visualised by viewing the respective sine ratios as quadrature sinusoid components of the associated rotation. The fused pitch θ and fused roll ϕ come together with the fused hemisphere h to define the tilt rotation component of a rotation. The magnitude of this tilt rotation is given by the tilt angle α, and the relative direction is given

by the tilt axis angle γ. The fused pitch and roll can be thought of as a way of 'splitting up' the action of α into orthogonal components, much like a vector can be resolved into components relative to a coordinate frame. More precisely, the sine ratios $\sin\phi$ and $\sin\theta$ are in fact a decomposition of $\sin\alpha$ into quadrature sinusoid components, i.e. sinusoid components that are exactly 90° out of phase, and trace out a uniform circle parametrically. This quadrature nature is exemplified by the equations

$$\sin\phi = \sin\alpha\cos\gamma, \tag{6.35a}$$
$$\sin\theta = \sin\alpha\sin\gamma, \tag{6.35b}$$

and

$$\sin\alpha = \sqrt{\sin^2\phi + \sin^2\theta}, \tag{6.36a}$$
$$\gamma = \operatorname{atan2}(\sin\theta, \sin\phi), \tag{6.36b}$$

which are strongly resemblant of the standard equations that relate Cartesian coordinates to polar coordinates. The property of axisymmetry of the fused pitch and roll is equivalent to stating that the choice of global reference x and y-axis simply results in a fixed phase shift to the quadrature components, i.e. as suggested by Equation (6.29). This suggests that the nature of fused pitch and roll in expressing a rotation is a property of the actual physical rotation, not whatever arbitrary reference frame is chosen to numerically quantify it.

The uniform circular nature of the fused pitch and roll sine ratios, and how these vary axisymmetrically in a purely rotational manner with respect to β, can be seen in Figure 6.7. The locus of the sine ratio pair $(\sin\phi_\beta, \sin\theta_\beta)$ is plotted for a particular physical rotation ${}^{U}_{C}F$ as β varies, and the quadrature decomposition of $\sin\alpha$ into $\sin\phi$ and $\sin\theta$ components is illustrated graphically by the orange triangle. The result that the value of β directly additively affects the tilt axis angle γ, as stated in Equation (6.29), can also be observed.

6.2.4.3 *Axisymmetry of the Tilt Phase Space*

The phase roll p_x and phase pitch p_y are type (b) axisymmetric rotation parameters. This can most easily be seen from the axisymmetry of the tilt angles parameters, where from Equations (6.27) and (6.29) we can see that

$$p_{x\beta} = \alpha_\beta\cos\gamma_\beta = \alpha_0\cos(\gamma_0+\beta) = \alpha_0(c_\beta c_{\gamma_0} - s_\beta s_{\gamma_0}), \tag{6.37a}$$
$$p_{y\beta} = \alpha_\beta\sin\gamma_\beta = \alpha_0\sin(\gamma_0+\beta) = \alpha_0(s_\beta c_{\gamma_0} + c_\beta s_{\gamma_0}). \tag{6.37b}$$

This can be simplified and factored into the matrix equation

$$\begin{bmatrix} p_{x\beta} \\ p_{y\beta} \end{bmatrix} = \begin{bmatrix} c_\beta & -s_\beta \\ s_\beta & c_\beta \end{bmatrix}\begin{bmatrix} p_{x0} \\ p_{y0} \end{bmatrix}. \tag{6.38}$$

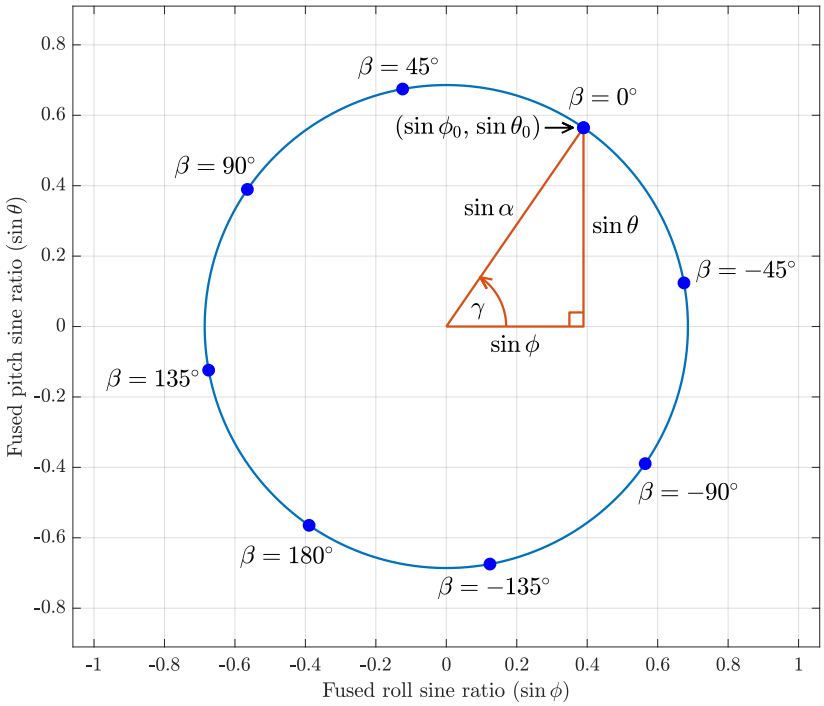


Figure 6.7: Locus of the fused roll and pitch sine ratios $(\sin\phi_\beta, \sin\theta_\beta)$ as β varies, i.e. for all possible choices of global reference x and y-axes, for the physical rotation ${}^{U}_{C}F = F(2.0, 0.6, 0.4, 1)$. Points in β-increments of 45° are labelled, and illustrate how β can be seen to be a positive offset to the tilt axis angle γ, as in Equation (6.29). The orange triangle demonstrates the decomposition of $\sin\alpha$ into the quadrature sinusoid components $\sin\phi$ and $\sin\theta$.

Similar to what Equation (6.33) did for the fused pitch and roll, this matrix equation epitomises the type (b) axisymmetry of the tilt phase space pitch and roll parameters. The middle matrix can clearly be identified as a 2D rotation matrix that encodes a CCW rotation by β, meaning that $(p_{x\beta}, p_{y\beta})$ varies in an intuitive rotational manner with respect to β, as required. As a result, it can be concluded that all three tilt phase space parameters are axisymmetric, as the third parameter, p_z, is just the fused yaw.

Like for fused angles, the tilt phase space pitch and roll parameters can also be interpreted to be quadrature sinusoid components that together describe the tilt rotation component. The phase pitch and roll parameters can be thought of as the most direct way of 'splitting up' the magnitude α of a tilt rotation into orthogonal components. These components correspond to quadrature sinusoids that are exactly 90° out of phase, and trace out a uniform circle parametrically. Similar to

Equations (6.35) and (6.36) for fused angles, the quadrature nature of the phase pitch and roll is highlighted by the equations

$$p_x = \alpha \cos \gamma, \tag{6.39a}$$
$$p_y = \alpha \sin \gamma, \tag{6.39b}$$

and their inverse relations

$$\alpha = \sqrt{p_x^2 + p_y^2}, \tag{6.40a}$$
$$\gamma = \text{atan2}(p_y, p_x). \tag{6.40b}$$

Due to these relations, as well as Equation (6.29), the property of axisymmetry for the phase roll and pitch can be seen to be equivalent to the observation that the choice of global reference x and y-axis simply results in a fixed phase shift to the quadrature components. This observation is illustrated more clearly later in Figure 6.10, as well as in Figure 6.8, which plots the locus of the tilt phase parameters (p_x, p_y) for a particular physical rotation as β varies. In the latter figure, the quadrature decomposition of α into p_x and p_y parameters is also depicted by means of an orange triangle, and the additivity of β to the tilt axis angle γ can be surmised from the β labels.

It is evident by now that fused angles and the tilt phase space share many similar positive properties. One property that the tilt phase space has, but the fused angles representation does not, is magnitude axisymmetry. This relates to the fact that equal angle magnitude rotations in any tilt direction (i.e. away from the z-axis) have equal norms in the 2D tilt phase space, and that this norm is directly proportional to the magnitude of tilt rotation. While the first of these properties holds for the fused angles sine ratios $(\sin \phi, \sin \theta)$, the latter does not, as the norm in the sine ratio space starts changing more slowly the closer a tilt rotation gets to a magnitude of 90°. Magnitude axisymmetry is an important property, for example in cases of rotations or orientations being used for the purposes of feedback control. In such cases, for control theoretical reasons it is often desired that rotations that are, for instance, twice as large provide exactly twice as much contribution to the feedback error term.

6.2.4.4 *Non-axisymmetry of Euler Angles*

To demonstrate that the Euler pitch and roll parameters are non-axisymmetric, we first turn to a simple example. Suppose we have a robot standing upright relative to the well-defined global z-axis, and two different global reference frames, $\{G_1\}$ and $\{G_2\}$, as defined in the bullet point list on page 91. The x-axis of $\{G_1\}$ and y-axis of $\{G_2\}$ both point forwards relative to the robot, and the y-axis of $\{G_1\}$ and x-axis of $\{G_2\}$ point in antiparallel, i.e. opposite, directions—leftwards and rightwards relative to the robot, respectively. If the robot performs a

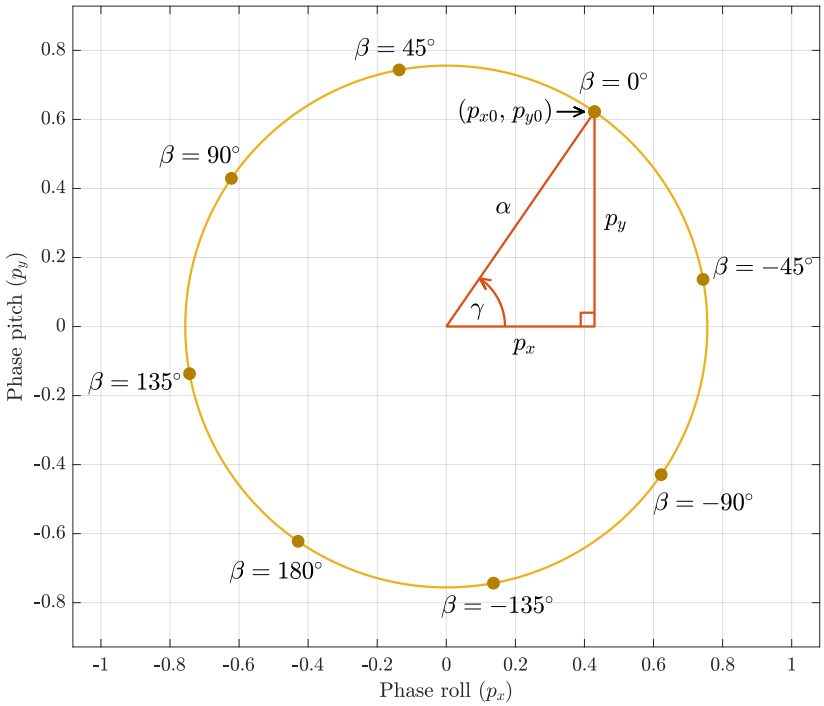


Figure 6.8: Locus of the phase roll and pitch $(p_{x\beta}, p_{y\beta})$ as β varies, i.e. for all possible choices of global reference x and y-axes, for the same physical rotation as in Figure 6.7. Points in β-increments of 45° are labelled, and illustrate how β can be seen to be a positive offset to the tilt axis angle γ, as in Equation (6.29). The orange triangle demonstrates the decomposition of α into the quadrature sinusoid components p_x and p_y.

90° CCW rotation about the horizontal axis 60° from being forwards and 30° from being leftwards, then the rotation quantified in terms of Euler angles relative to $\{G_1\}$ is given by

$$E_1 = (1.047, 1.047, 1.571), \tag{6.41}$$

and in terms of Euler angles relative to $\{G_2\}$ is given by

$$E_2 = (-0.524, 0.524, -1.571). \tag{6.42}$$

As $\mathbf{x}_{G_1}$ and $\mathbf{y}_{G_2}$ are by definition both the same physical vector, and as E_1 and E_2 actually both represent the same physical rotation, logically (from the concept of pitch/roll axisymmetry) one would expect that the parameter of E_1 that quantifies the amount of rotation about the x-axis, i.e. the Euler roll ϕ_{E1}, is numerically the same as the parameter of E_2 that quantifies the amount of rotation about the y-axis, i.e. the Euler pitch θ_{E2}. This is not the case however, as from above, $\phi_{E1} = 1.571$ and $\theta_{E2} = 0.524$. Similarly, the vectors $\mathbf{y}_{G_1}$ and $\mathbf{x}_{G_2}$ are negatives of each other, so logically from axisymmetry one would expect that θ_{E1} is numerically the negative of ϕ_{E2}, but this is also not the case, as

$\theta_{E1} = 1.047$ and $\phi_{E2} = -1.571$. Thus, it can be concluded that the Euler angles representation does *not* satisfy pitch/roll axisymmetry. Performing the same calculations for fused angles and the tilt phase space yields

$$\begin{aligned} F_1 &= (0,\, 1.047,\, 0.524,\, 1), \\ F_2 &= (0,\, 0.524,\, -1.047,\, 1), \\ P_1 &= (0.785,\, 1.360,\, 0), \\ P_2 &= (-1.360,\, 0.785,\, 0). \end{aligned} \tag{6.43}$$

It is immediately clear by inspection that pitch/roll axisymmetry is satisfied in this example, for both representations, as

$$\begin{aligned} \phi_1 &= \;\;\theta_2 \;\; = 0.524, \\ \theta_1 &= -\phi_2 \;\; = 1.047, \\ p_{x1} &= \;\;p_{y2} = 0.785, \\ p_{y1} &= -p_{x2} = 1.360. \end{aligned} \tag{6.44}$$

Fused yaw axisymmetry can also be observed in Equation (6.43), as

$$\begin{aligned} \psi_1 = \psi_2 \;\; &= 0, \\ p_{z1} = p_{z2} &= 0, \end{aligned} \tag{6.45}$$

but the same cannot be said about the Euler yaw, as

$$\psi_{E1} = 1.047 \neq -0.524 = \psi_{E2}. \tag{6.46}$$

Similar to Equation (6.21) for Euler yaw, the non-axisymmetry of the Euler pitch and roll can also be demonstrated mathematically. Using the notation from Equations (6.18) and (6.19) (see also Section 6.2.3.1 for the definition of β), the Euler pitch and roll of ${}^{G}_{B}R$ can be derived to be

$$\theta_{E\beta} = \operatorname{asin}\left(c_\beta s_{\theta_{E0}} + s_\beta c_{\theta_{E0}} s_{\phi_{E0}}\right), \tag{6.47a}$$

$$\phi_{E\beta} = \operatorname{atan2}\left(c_\beta c_{\theta_{E0}} s_{\phi_{E0}} - s_\beta s_{\theta_{E0}},\, c_{\theta_{E0}} c_{\phi_{E0}}\right). \tag{6.47b}$$

Clearly these expressions are neither independent of β, as would be required for type (a) axisymmetry, nor do they vary in an intuitive rotational manner with respect to β like Equations (6.33) and (6.38) do, as would be required for type (b) axisymmetry. The non-axisymmetry of these expressions can be better visualised by plotting them against each other as a locus of β. This has been done in Figure 6.9, where the loci of the Euler angles sine ratios $(\sin\phi_E, \sin\theta_E)$, fused angles sine ratios $(\sin\phi, \sin\theta)$, and tilt phase components (p_x, p_y) have been plotted for all $\beta \in (-\pi, \pi]$, for a physical rotation of

$${}^{U}_{C}R = F_R(-1.2,\, 0.2,\, -1.3,\, 1).$$

The relationships between corresponding points on the three loci are shown in the figure using dotted lines. The uniform circular nature

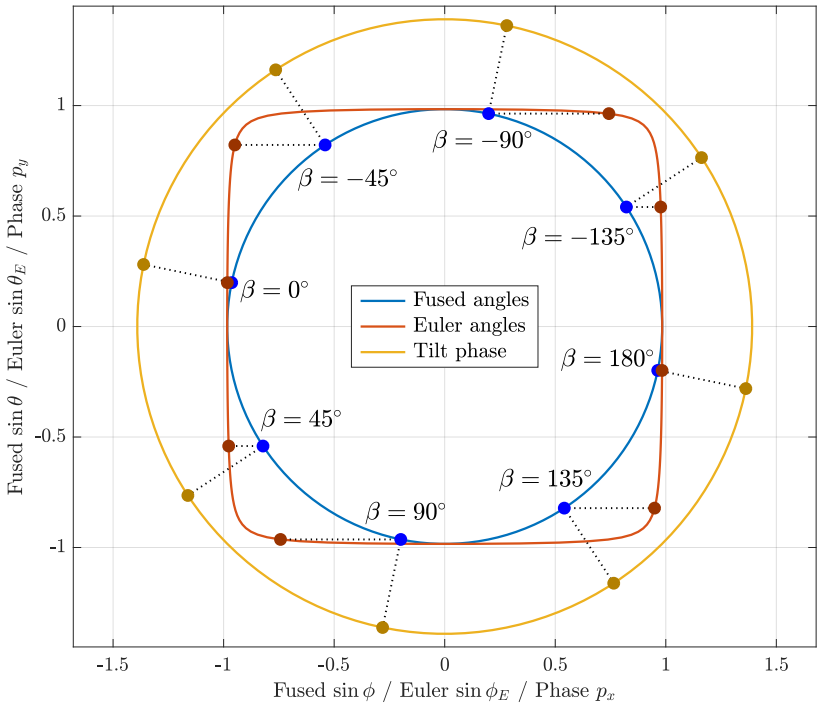


Figure 6.9: The fused angles $(\sin\phi_\beta, \sin\theta_\beta)$, Euler angles $(\sin\phi_{E\beta}, \sin\theta_{E\beta})$, and tilt phase space $(p_{x\beta}, p_{y\beta})$ loci of pitch vs. roll as β varies, for the physical rotation ${}^U_C R = F_R(-1.2, 0.2, -1.3, 1)$. For each locus, points in β-increments of 45° are labelled, and joined to each other between the three loci using dotted lines. For $\beta = 0$ we have that ${}^U_C R = {}^G_B R$, and for instance that $(p_{x\beta}, p_{y\beta}) = (p_{x0}, p_{y0})$. The non-circularity of the Euler angles locus, as well as the non-uniform spacing of the associated keypoints, demonstrates the violation of axisymmetry for Euler pitch and roll.

of the fused angles and tilt phase space loci are clear indications of the type (b) axisymmetry of their associated pitch and roll parameters, while the irregular non-uniform shape of the Euler angles locus clearly demonstrates the inherent non-axisymmetry of the Euler pitch and roll parameters. Conceptually, the problem of Euler angles is the fundamental requirement of a sequential order of rotations, leading to definitions of pitch and roll that do not correspond to each other in behaviour, as one then implicitly depends on the value of the other.

6.2.4.5 *Visualising Pitch/Roll Axisymmetry*

Although the nature of pitch/roll axisymmetry has been somewhat visualised in previous figures, e.g. Figures 6.7 and 6.8, we now complete the picture with some further depictions of the nature of the various pitch and roll parameters.

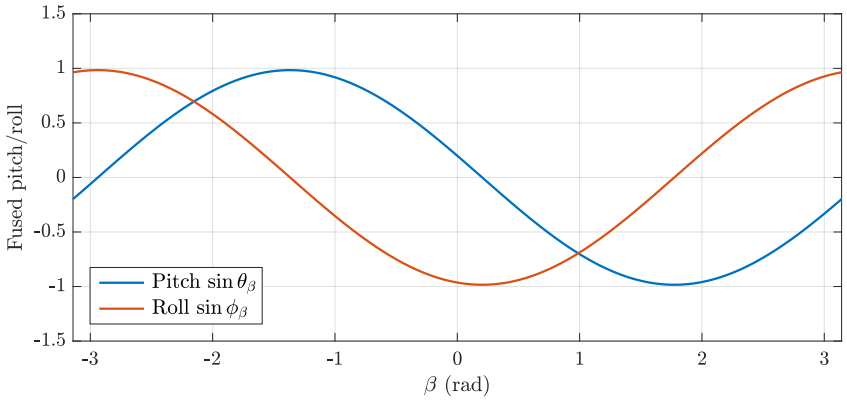
VISUALISATION A While the variation of fused and Euler yaw with respect to β has already been plotted in Figure 6.4 for one particular physical rotation, we now do the same for the various different notions of pitch and roll that have been compared in this chapter. Figure 6.10 shows the variations of the fused, phase and Euler pitch and roll as β ranges from $-\pi$ to π, for the exact same physical rotation as in Figure 6.9. While 90° out-of-phase quadrature sinusoid waveforms can be observed in the fused angles and tilt phase space plots, as discussed in Sections 6.2.4.2 and 6.2.4.3, the Euler angles plot reveals in particular the irregularity and non-axisymmetry of the Euler roll parameter.

VISUALISATION B We know from the type (a) axisymmetry of the tilt angle α, i.e. Equation (6.27), and the type (b) axisymmetry of the tilt axis angle γ, i.e. Equation (6.29), that all possible representations ${}^{G}_{B}R$ of a particular physical rotation ${}^{U}_{C}R$ have the same value of α, and take on every possible value of γ. Thus, any single locus as per Figure 6.9 can be generated by plotting the respective pitch and roll parameters, for all rotations that have a tilt angle of α_0, i.e. the same tilt angle as ${}^{U}_{C}R$. Consequently, all possible loci of pitch and roll (as β varies) can be examined at once by plotting the level sets of constant α in the pitch/roll plane. This has been done for the phase pitch and roll in Figure 6.11. For fused angles and Euler angles, it turns out that it is equivalent and more correct to plot the level sets of constant $\sin\alpha$ in the pitch/roll plane, as suggested by relations like Equation (6.36a). The resulting plots are shown in Figure 6.12. While the fused angles and tilt phase space plots demonstrate clear axisymmetry and uniformity about the origin, the Euler angles level set contours get stretched apart into a more rounded square-like form, and for $\sin\alpha = 1$, the top and bottom edges of the square can even be considered to completely 'open up' due to the gimbal lock singularities.

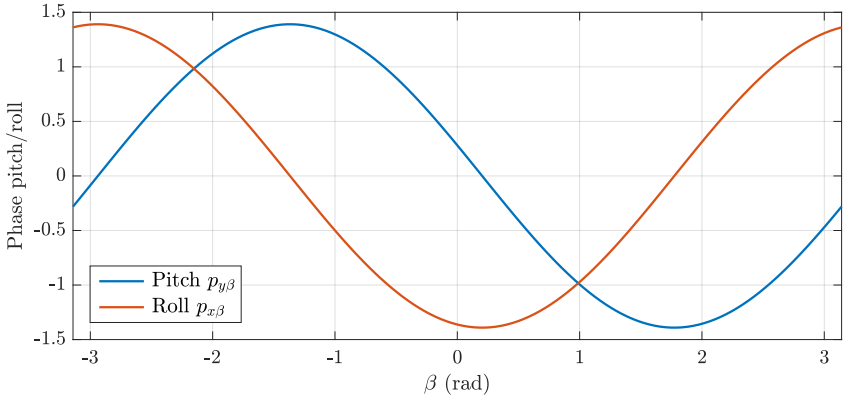
For the purposes of understanding what is going on, one should note that the fused angles and Euler angles $\sin\alpha$ level set plots in Figure 6.12 are actually 'double covers' of the equivalent α level sets, as

$$\sin\alpha \equiv \sin(\pi - \alpha). \tag{6.48}$$

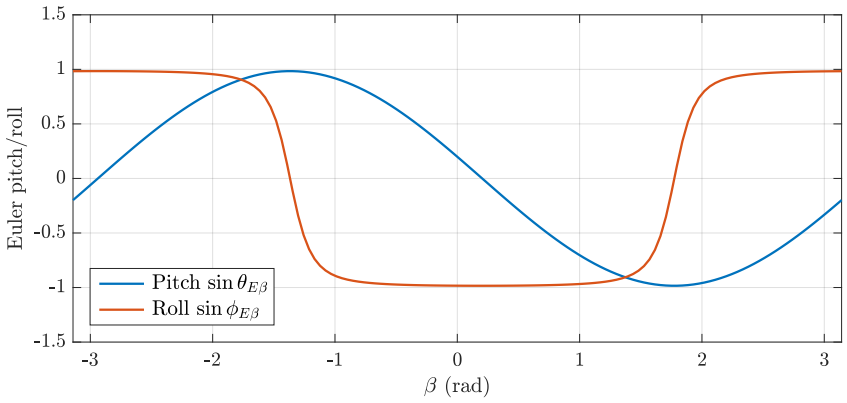
In fact, each cover corresponds exactly to one fused hemisphere h. Being slightly lenient with regards to gimbal lock, the top h-hemisphere corresponds to a Euler pitch/roll in the domain $[-\frac{\pi}{2}, \frac{\pi}{2}] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$, while the bottom h-hemisphere corresponds to the Euler roll being in the remainder of its domain, i.e. larger than $\frac{\pi}{2}$ in magnitude. For fused angles, as α moves from 0 to π, the level sets start as a single point at the origin, expand uniformly to a circle of radius 1 when $\alpha = \frac{\pi}{2}$, and then shrink back down uniformly again to a point as α approaches π. If one were to plot the raw fused angles (ϕ, θ) instead of their sine ratios $(\sin\phi, \sin\theta)$, then a similar behaviour



(a) Plot of the fused roll and pitch sine ratios $(\sin\phi_\beta, \sin\theta_\beta)$ against β.



(b) Plot of the phase roll and pitch $(p_{x\beta}, p_{y\beta})$ against β.



(c) Plot of the Euler roll and pitch sine ratios $(\sin\phi_{E\beta}, \sin\theta_{E\beta})$ against β.

Figure 6.10: Plots of fused, Euler and phase pitch and roll against β for the determination of parameter axisymmetry (see Figure 6.4 for the yaw plot). Exactly as in Figure 6.9, a β of zero corresponds to the chosen physical rotation of ${}^{U}_{C}R = F_R(-1.2, 0.2, -1.3, 1)$. While exact quadrature sinusoid pitch/roll waveforms can be identified in the (a) fused angles and (b) tilt phase space plots, in particular the irregularity and non-axisymmetry of the Euler roll parameter can be identified in the (c) Euler angles plot.

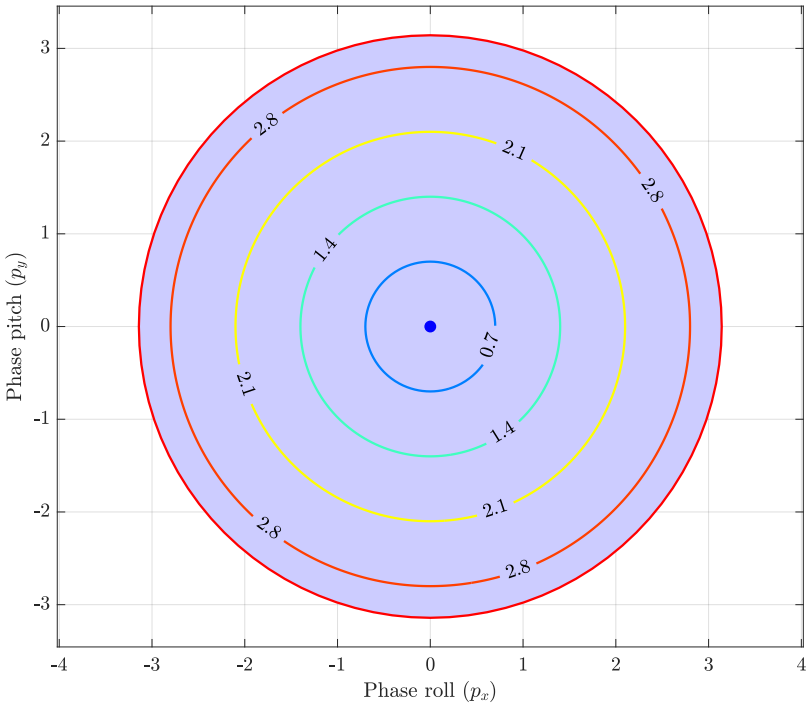
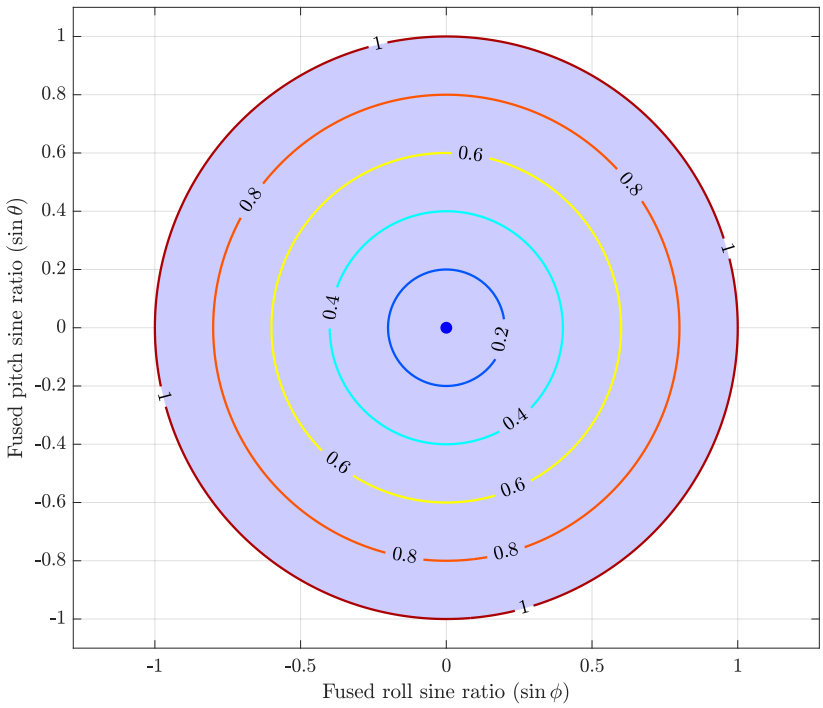


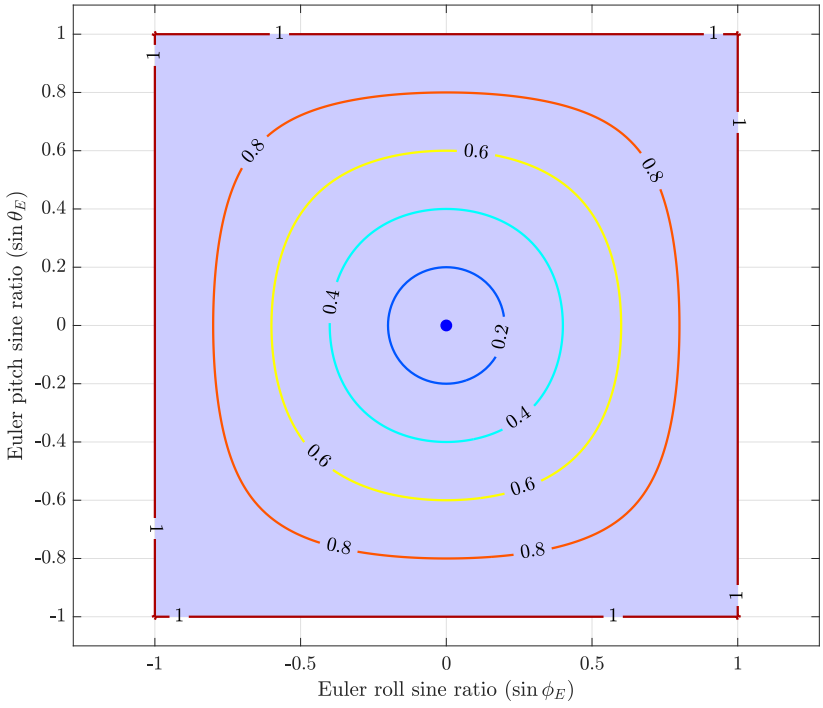
Figure 6.11: Level sets of constant α in the (p_x, p_y) tilt phase space plane. The shaded region is the valid (bounded) 2D tilt phase space domain. The uniformly circular nature of the plot visually illustrates the axisymmetry of the phase pitch and roll.

occurs, only the level sets around $\alpha = \frac{\pi}{2}$ grow to a diamond shape and back. For Euler angles, as α moves from 0 to π, the level sets shown in Figure 6.12b start expanding more or less circularly, but quickly stretch out into square form, where the top and bottom edges of the square become increasingly stretched and less densely populated with β points. At $\alpha = \frac{\pi}{2}$, the level set essentially corresponds to two separate vertical lines, at $\sin\phi_E = \pm 1$, although it is somewhat a matter of definition how the Euler angles parameters exactly behave at gimbal lock. Beyond that, the exact reverse process of shrinking back down to the origin occurs. If one were to plot the raw Euler angles (ϕ_E, θ_E) instead of their sine ratios, the level sets would initially behave somewhat qualitatively similarly, in that a dot grows approximately circularly before becoming more square-like and ending up at $\alpha = \frac{\pi}{2}$ as two vertical lines at $\phi_E = \pm\frac{\pi}{2}$. Beyond this however, the raw level sets flip outside of these lines and shrink back down to a point at $\phi_E = \pm\pi$ in a somewhat symmetrical reverse process.

VISUALISATION C As discussed in Section 6.2.3.2, and shown in Figures 6.5 and 6.6 (see page 96), the axisymmetry of rotation parameters can be examined by constructing surface plots of their values for all tilt rotations up to $\frac{\pi}{2}$ radians in magnitude. This is



(a) Level sets of constant $\sin\alpha$ in the $(\sin\phi, \sin\theta)$ fused angles plane.



(b) Level sets of constant $\sin\alpha$ in the $(\sin\phi_E, \sin\theta_E)$ Euler angles plane.

Figure 6.12: Level sets of constant $\sin\alpha$ in the fused angles and Euler angles pitch/roll planes. The shaded region in each case corresponds to the valid pitch/roll domain. The uniformly circular nature of (a) the fused angles plot, in contrast to (b) the Euler angles plot, visually illustrates the axisymmetry of the fused pitch and roll.

because the action of tilting an upright robot away from the z-axis in all different directions is fundamentally equivalent to representing a single tilt rotation of 0 to $\frac{\pi}{2}$ radians relative to all different possible global reference frames. The interpretation of Figure 6.5a, and how it demonstrates the type (a) axisymmetry of the fused yaw and the non-axisymmetry of the Euler yaw, has already been discussed in Section 6.2.3.2. In Figures 6.6a and 6.6b, we can see the surface plots for fused pitch and roll, and phase pitch and roll, respectively. It can clearly be identified that the respective pitch and roll components are mutually symmetric in each case, and mirror each other in behaviour throughout their entire domains. Specifically, a simple 90° rotation about the z-axis maps both roll surface plots directly onto their respective pitch surface plots. This rotation is equivalent to simply reinterpreting all of the plotted tilt rotations with a β of $\frac{\pi}{2}$, as per Equation (6.13), so this is one example of pitch/roll axisymmetry that can be directly interpreted from the figure. Looking at the Euler angles plots in Figure 6.5b, the same cannot be said for the Euler pitch and roll. The Euler roll plot shows clear distortion effects due to the gimbal lock singularities, has a much larger range of values (from $-\pi$ to π, although this cannot be seen in the figure as only $\alpha \leq \frac{\pi}{2}$ is plotted), and does not correspond at all to the pitch plot via any rotation. This illustrates why it can be said that the Euler pitch and roll parameters are *not* axisymmetric, and do *not* correspond to each other in domain and/or behaviour (refer to the statement of Problem D on page 84).

6.3 CONCLUSION

This chapter has demonstrated four fundamental reasons why Euler angles are not a suitable representation for representing orientations in balance-related applications, and why the fused angles and tilt phase space representations do better in all four of these areas. Euler angles may still be suitable for the analysis of physical gimbals or a collocated series of joints, but for almost all other applications, the proximity of the gimbal lock singularities to normal working ranges (greatly affecting even the non-yaw components), and the lack of parameter axisymmetry (leading to illogical and non-self-consistent behaviour of the various parameters), make Euler angles unsuitable for the task. The core problem of Euler angles is the inferior reliance on a sequential order of rotations, in particular forcing an order on the pitch and roll parameters instead of allowing them to be concurrent. This leads to mutual parameter dependencies that corrupt the 'purity' of the first and third rotation parameters, as can for example clearly be seen for the Euler yaw and roll in Figure 6.5. None of these problems are the case for the fused angles and tilt phase space representations, where for example it can be observed that the fused angles representation combines the 'uncorrupted' second

rotation parameters of the ZYX and ZXY Euler angles representations respectively, with a novel and self-consistent definition of yaw, to form a genuinely useful and geometrically meaningful representation of orientation for balance.

7 ATTITUDE ESTIMATION

In Chapter 4, we discussed in detail how the acquired sensor data is calibrated and filtered to ensure that it is maximally useful in measuring the state of the robot. The acquired data included elements like the gyroscope and accelerometer values, with magnetometer values being treated separately in Allgeuer (2020). In this chapter, we discuss how the calibrated and corrected sensor data is fused together into higher level estimates of the state of the robot. In particular, we discuss how the orientation (also referred to as the attitude) of the robot is estimated from the Inertial Measurement Unit (IMU) data. The result of this estimation is used by the various implemented gaits for the purpose of future state prediction and feedback.

Attitude estimation is the task of constructing an estimate of the full 3D orientation of a body relative to a global fixed frame, based on a finite history of sensor measurements, e.g. gyroscope, accelerometer and magnetometer measurements. The body in question is often a robot, but in principle it can correspond to any object that is equipped with the sensors necessary for the estimation task. With the advent of low-cost inertial sensors—particularly those based on Micro-Electro-Mechanical Systems (MEMS)—the field of application for attitude estimation techniques has greatly widened, extending into the field of low-cost robotics. With low cost sensors and processors however, it is crucial that any estimation algorithms are able to run computationally efficiently, and are able to function with high noise inputs without excessively sacrificing estimator response. In addition to low estimator latency, orientation-independent mathematical and numerical stability are also desirable. For balance-related applications like bipedal walking, it is also important that the magnetometer does not influence the non-heading components of the estimated result, as these are the components that are most relevant for balance feedback, and the magnetometer is generally a less robust sensor than the gyroscope and accelerometer due to the high prevalence of magnetic inconsistencies, disturbances and distortions.

An attitude estimator that fulfils the aforementioned criteria is presented in this chapter. The estimator is available as part of the Humanoid Open Platform ROS Software[1], or completely separately as a generic portable C++ library (Allgeuer, 2016). All of the algorithms and discussed cases are implemented in the release(s), and have been tested both in simulation and on all of the real humanoid platforms

1 https://github.com/AIS-Bonn/humanoid_op_ros/blob/master/src/nimbro_robotcontrol/util/rc_utils/include/rc_utils/attitude_estimator.h

used throughout this thesis, including in particular for many years under the strenuous conditions of the RoboCup competition.

7.1 RELATED WORK

Much effort has been made in the past to develop algorithms for the reconstruction of attitude in aeronautical environments. This work was largely in relation to the Attitude and Heading Reference Systems (AHRS) required for aeronautical applications, with examples being the works of Gebre-Egziabher et al. (2004) and Munguía and Grau (2014). Other classic works in the area of attitude estimation, such as Vaganay et al. (1993) and Balaram (2000), have focused more on robotics and control applications, but do not specifically address the issues encountered with low-cost IMU systems. A comprehensive survey of modern nonlinear filtering methods for attitude estimation was undertaken by Crassidis et al. (2007). Almost all of the surveyed advanced filtering techniques relied on some form of the Extended Kalman Filter (EKF), with various modifications being used to improve particular characteristics of the filter, often convergence. Such EKF filters can be seen to be computationally expensive however, when considering implementation on embedded targets such as microcontrollers. It is often also difficult to provide a guarantee of filter robustness (Euston et al., 2008), and difficult to ensure that the magnetometer measurements do not significantly affect the estimated non-heading components of orientation.

Alternative to the general stream of development of EKF filtering is the concept of complementary filtering. This builds on the well-known linear single-input single-output (SISO) complementary filters, and extends these in a nonlinear fashion to the full 3D orientation space. Such filters have favourable frequency response characteristics, and seek to fuse low frequency attitude information with high frequency attitude rate data. Prominent examples of generalised complementary filters include the works of Jensen (2011) and Mahony et al. (2008). A different approach again is taken by Madgwick et al. (2011), who formulate a numerical filter directly in the quaternion space, and numerically integrate quaternion velocities $\dot{q}$ with the use of renormalisation to estimate the required orientation. The method, developed in the context of wearable devices for rehabilitation robotics, does not respect magnetometer independence, and uses a non-ideal gradient descent numerical approximation for orientation reconstruction that converges over multiple cycles. An empirical comparison of the Madgwick filter, a complementary filter and an EKF-based approach is performed in Cavallo et al. (2014).

The orientation estimation problem addressed in this thesis relates specifically to the design of an attitude estimator that can function with noisy low-cost sensors, and that is simple and efficient enough to

be implemented at high loop rates on low-power embedded targets, such as microcontrollers. To this end, the work presented by Mahony et al. (2008) was used as a basis for the developed attitude estimator. A central problem in applying this work however, is that a method is required for reconstructing an instantaneous 3D orientation 'measurement' directly from the sensor measurements (see Section 7.5). This is a complex optimisation problem that generally requires a suboptimal solution algorithm for computational feasibility reasons (Mahony et al., 2008). Literature does not elucidate a clear solution to this problem—in particular not in an explicit form—and not in a way that can function robustly in all cases. One of the main contributions of the work presented here lies in the development of an algorithm for the robust calculation of such instantaneous orientation measurements. Other contributions include the novel use of fused yaw (see Section 5.3.1) in an estimator, the integration of a *quick learning* scheme (Section 7.6.1), and the explicit extension of the attitude estimator to cases of reduced sensory information (Section 7.6).

7.2 PROBLEM DEFINITION AND NOTATION

The goal of the attitude estimation process is to calculate an estimate of the rotation of a body relative to a global reference frame, based on observations acquired through sensory perception. Such sensory perception can include accelerometers, gyroscopes, magnetometers, Global Positioning System (GPS), visual perception and/or Light Detection and Ranging (LIDAR). The types of sensors considered for the task in this thesis are the ones that are typically found in IMU systems, and are typically available in low-cost variants for mobile robotic systems, i.e. gyroscope, accelerometer and magnetometer sensors. No matter which sensors are used however, it is always a stringent requirement that the estimator is globally stable, and is able to function equally well throughout the entire orientation space.

We define {G} to be the global reference frame relative to which the orientation of the body is estimated, and {B} to be the body-fixed frame that rotates with the body, and therefore with the sensors that provide the observational input to the attitude estimator. It is assumed that {B} is defined in such a way that its z-axis points 'upwards' (and x-axis points 'forwards') relative to the body, and correspondingly, that {G} is defined in such a way that its z-axis points 'upwards' relative to the world, i.e. in the direction opposite to gravity. Importantly, this means that the gravity vector can be written as

$$^{G}\mathbf{g} = (0, 0, -g), \tag{7.1}$$

where $g = 9.81\,\mathrm{m/s^2}$. These are the standard definitions of {G} and {B} that were used throughout the rotations chapters (Chapters 5 and 6). All further notation that was introduced and used there is also

considered to carry across to this chapter, including for instance the rotation basis notation

$$ {}^{G}_{B}R = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ {}^{G}\mathbf{x}_{B} & {}^{G}\mathbf{y}_{B} & {}^{G}\mathbf{z}_{B} \\ \downarrow & \downarrow & \downarrow \end{bmatrix} = \begin{bmatrix} \leftarrow & {}^{B}\mathbf{x}_{G} & \rightarrow \\ \leftarrow & {}^{B}\mathbf{y}_{G} & \rightarrow \\ \leftarrow & {}^{B}\mathbf{z}_{G} & \rightarrow \end{bmatrix}, \tag{7.2} $$

where for example ${}^{B}\mathbf{z}_{G}$ is the unit z-vector of frame {G}, expressed in the coordinates of frame {B}. Note that throughout this chapter we refer to two vectors as

- **Parallel** if they are linearly dependent and a positive multiple of each other,
- **Antiparallel** if they are linearly dependent and a negative multiple of each other, and
- **Collinear** if they are either parallel or antiparallel.

As a final note, the concepts of *fused yaw* and *tilt* are used in many places within this chapter, and should be reviewed, if necessary, from Sections 5.3.1 and 5.3.2.

7.3 SENSOR INPUTS

Using the notation and definitions introduced in the previous section, the problem considered in this paper can be more precisely reformulated as being the task of robustly calculating an online estimate for ${}^{G}_{B}q$ (or ${}^{G}_{B}R$), given arbitrary time sequences of 3-axis gyroscope, accelerometer and magnetometer data. Keeping in mind the sensor and calibration models introduced in Chapter 4, the format and type of data provided by each of these sensors is assumed to be modelled as follows.

GYROSCOPE This sensor is assumed to provide a measure ${}^{B}\boldsymbol{\Omega}_{y} \in \mathbb{R}^3$ of the angular velocity of the body, in the coordinates of frame {B}. The conversion of the raw data from the IMU frame {I} to the body-fixed frame {B} is performed using the calibrated IMU orientation offset ${}^{B}_{I}R$ (see Section 4.1.1). As given by Equation (4.3), after scale correction and temperature compensation, the gyroscope measurement is assumed to be affected by a largely time-invariant gyroscope bias $\mathbf{b}_{\Omega}$, along with zero mean sensor noise $\mathbf{w}_{\Omega}$. The bias $\mathbf{b}_{\Omega}$ is managed using the gyroscope bias calibration and online gyroscope bias autocalibration schemes (see Sections 4.1.3 and 4.1.4), and the sensor noise $\mathbf{w}_{\Omega}$ is not explicitly abated using low-pass filtering, as the gyroscope data is directly integrated as part of the estimation process, and it is the nature of integration to dampen out high frequency components of a signal. The final corrected and calibrated angular velocity measurement

$$ {}^{B}\boldsymbol{\Omega}_{y} \in \mathbb{R}^3, \tag{7.3} $$

is used as the gyroscope input to the attitude estimator, and is expressed in the units of rad/s.

ACCELEROMETER This sensor is assumed to provide a measure ${}^{B}\tilde{\mathbf{a}} \in \mathbb{R}^3$ of the proper acceleration of the body. This is the inertial coordinate acceleration being experienced by the body, together with the effect of gravitational acceleration. The latter term is assumed to dominate the measured proper acceleration. Cases where this assumption is violated, like for example in collisions, are implicitly filtered out by the low-pass dynamics of the estimator. Grouping the inertial acceleration components with a zero mean noise term $\mathbf{w}_a$, yields the measurement model

$$ {}^{B}\tilde{\mathbf{a}} = -{}^{B}_{G}R\,{}^{G}\mathbf{g} + \mathbf{w}_a. \tag{7.4} $$

The minus sign comes from the definition of proper acceleration, as an object at rest on the Earth's surface, for instance, has a proper acceleration of $g = 9.81\,\mathrm{m/s^2}$ upwards, but the gravity vector ${}^{G}\mathbf{g}$ nominally points downwards, as in Equation (7.1). The raw accelerometer measurements are brought from the IMU frame {I} to the body-fixed frame {B} using the IMU orientation offset ${}^{B}_{I}R$ (see Section 4.1.1), and are subsequently low-pass filtered using a mean filter to combat the sensor noise. The resulting measured proper acceleration is then

$$ {}^{B}\tilde{\mathbf{a}} \approx -{}^{B}_{G}R\,{}^{G}\mathbf{g} = {}^{B}_{G}R\,(0,0,g) \tag{7.5a} $$

$$ = g\,{}^{B}\mathbf{z}_G. \tag{7.5b} $$

Thus, for each filtered accelerometer measurement ${}^{B}\tilde{\mathbf{a}}$, we can construct an estimate

$$ {}^{B}\tilde{\mathbf{z}}_G = \frac{{}^{B}\tilde{\mathbf{a}}}{\|{}^{B}\tilde{\mathbf{a}}\|} \in \mathcal{S}^2, \tag{7.6} $$

of the positive global z-vector ${}^{B}\mathbf{z}_G$, and use this as the accelerometer-based input to the attitude estimator.

MAGNETOMETER The measurement model and calibration process of the magnetometer sensor is relatively complex, and is described in detail in Allgeuer (2020). The described magnetometer correction pipeline involves spike and mean filtering of the data, as well as the application of hard and/or soft iron corrections, and/or cyclic angle warping for final fine-tuning. It is assumed that the final extracted magnetometer measurement ${}^{B}\tilde{\mathbf{m}}$ is expressed relative to the body-fixed frame {B}, and is a direct measure of the strength and direction of the Earth's magnetic field ${}^{G}\mathbf{m}_e = (m_{ex}, m_{ey}, m_{ez})$, i.e.

$$ {}^{B}\tilde{\mathbf{m}} \approx {}^{B}_{G}R\,{}^{G}\mathbf{m}_e \in \mathbb{R}^3. \tag{7.7} $$

7.4 COMPLEMENTARY FILTERING

The method of attitude estimation presented in this chapter is based on the concept of complementary filtering.

7.4.1 1D Linear Complementary Filter

A simple preliminary approach to the attitude estimation problem is to separate the problem into each of its independent axes of rotation. This can work for body rotations close to the upright identity pose, but does not extend well to the whole orientation space. Nevertheless, the 1D filtering approach demonstrates well the concept of linear complementary filtering. Taking for example the pitch direction of rotation, one can express the filter equations as

$$\dot{\hat{\theta}} = \omega_y - \hat{c}_\omega + k_p(\theta_y - \hat{\theta}) \tag{7.8a}$$

$$\dot{\hat{c}}_\omega = -k_i(\theta_y - \hat{\theta}), \tag{7.8b}$$

where

- $\hat{\theta}$ $\Rightarrow$ Estimated pitch angle of the body (output of the filter)
- ω_y $\Rightarrow$ Angle rate measurement in the pitch direction, based on the gyroscope
- $\hat{c}_\omega$ $\Rightarrow$ Estimated integral term as an offset to ω_y
- θ_y $\Rightarrow$ An instantaneous measurement of the pitch angle, based solely on the accelerometer (and possibly magnetometer)
- k_p $\Rightarrow$ Proportional gain indicating the influence of θ_y on $\hat{\theta}$
- k_i $\Rightarrow$ Integral gain indicating the influence of θ_y on $\hat{c}_\omega$

Proportional-Integral (PI) filter equations similar to Equation (7.8) can also be formulated for the roll and yaw directions, where, if no magnetometer is used, the $\theta_y - \hat{\theta}$ error term for the yaw case needs to be left as zero, as the accelerometer alone cannot yield information about the yaw of the body.

Effectively, the PI compensation closes the loop on the type I system, forming a linear second-order system with zero theoretical steady state error to step inputs. The linear complementary filter combines the high-pass rate data with the low-pass position data to form a high bandwidth estimate of the system state. However, despite possessing positive filter attributes, the assumption that each axis behaves independently places a severe limitation on the usability of the filter for attitude estimation. A core issue is that the angular velocity about one axis generally affects the rotation about all axes, and to differing amounts depending on the orientation of the body. The three independent 1D filters also do not clearly indicate how

the three estimated outputs can be unambiguously and meaningfully combined into a total estimation of the 3D orientation of the body.

7.4.2 Extension to 3D Nonlinear Filtering

In light of the limitations of the 1D complementary filter, it is desirable to formulate a complementary filter that operates on the full 3D rotation space, ideally retaining the positive frequency attributes of the linear filter. Mahony et al. (2008) introduced three such nonlinear filters, the *direct*, *passive* and *explicit complementary filters*. The main difference between the three filters is that while the direct complementary filter uses the instantaneous inertial sensor data to transform the gyroscope measurements in the update equation, the passive complementary filter uses the current filter estimate, and the explicit complementary filter uses an update technique that operates directly on the sensor measurement vectors.

A key design decision of the attitude estimator presented here is that the magnetometer measurements should not have any direct influence on the attitude estimate, other than to resolve the yaw, i.e. heading. The reason for this is to reduce instabilities in the output pitch and roll components, and to alleviate the requirement of performing a magnetometer calibration in order for these components of the estimate to function correctly. This is impossible to achieve with the explicit complementary filter, and so the only filter in Mahony et al. (2008) to provide a solution to the problem of constructing an instantaneous orientation measurement from sensor data was found to be unsuitable. Comparison of the direct and passive filters also led to the conclusion that the feed-forward nature of the direct formulation was unsuitable due to high frequency noise considerations. Consequently, the attitude estimator presented in this chapter is built around the core of the nonlinear passive complementary filter.

7.4.3 3D Nonlinear Passive Complementary Filter

As indicated in Figure 7.1, we define frame {E} as the frame corresponding to the current estimate

$$\hat{q}_E^G \equiv \hat{q} \in \mathbb{Q} \tag{7.9}$$

of the orientation of the body-fixed frame {B}, relative to the global frame {G}.[2] In every update cycle, given the current accelerometer and magnetometer-based sensor measurements ${}^B\tilde{\mathbf{z}}_G$ and ${}^B\tilde{\mathbf{m}}$ (and

2 To aid understanding, if the output quaternion $\hat{q}$ of the attitude estimator is perfectly accurate, i.e. $\hat{q} = {}_B^G q$, then clearly {E} = {B}, so {E} is the attitude estimator's estimate of the frame {B}.

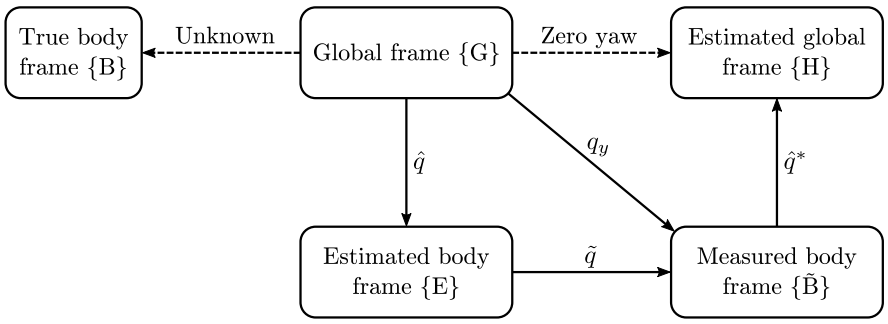


Figure 7.1: Overview of the coordinate frame definitions for the 3D nonlinear passive complementary filter, and measured quaternion orientation resolution methods.

if needed also $\hat{q}$), the first task is to construct a full 3D measured quaternion orientation

$$\overset{G}{\tilde{B}}q_y \equiv q_y \in \mathbb{Q}, \tag{7.10}$$

that is consistent with these measurements. The frame that results from this reconstruction of an orientation from the sensor measurements is labelled frame {$\tilde{B}$}, and within the accuracy of the sensors and respective calibrations should correspond to the frame {B}. The deviation of the measured quaternion q_y from the current estimated quaternion $\hat{q}$ can be calculated as

$$\tilde{q} = \hat{q}^* q_y, \tag{7.11}$$

and is referred to as the *error quaternion*

$$\overset{E}{\tilde{B}}\tilde{q} \equiv \tilde{q} \equiv (\tilde{q}_0, \tilde{\mathbf{q}}) \in \mathbb{Q}. \tag{7.12}$$

The rotation axis of this error quaternion rotates the current orientation estimate towards the orientation given by the sensor measurements, and leads to the corrective error feedback term

$$\boldsymbol{\Omega}_e = 2\tilde{q}_0\tilde{\mathbf{q}}. \tag{7.13}$$

This term is an angular velocity in $\mathbb{R}^3$ that rotates $\hat{q}$ towards q_y, with an angular speed of exactly the sine of the angle between these two orientations. Using the corrective error feedback term, the nonlinear 3D passive complementary filter equations can be written as

$$\dot{\hat{q}} = \tfrac{1}{2}\hat{q}(\boldsymbol{\Omega}_y - \hat{\mathbf{c}}_\Omega + k_p\boldsymbol{\Omega}_e), \tag{7.14a}$$

$$\dot{\hat{\mathbf{c}}}_\Omega = -k_i\boldsymbol{\Omega}_e, \tag{7.14b}$$

where we recall that quaternions and vectors can be directly multiplied with each other by making the vectors into quaternions themselves by adding a zero scalar part (w-component). The variables involved in the filter equations are given by

$\hat{q}$ $\Rightarrow$ Estimated quaternion orientation of the body (output of the filter)

$\boldsymbol{\Omega}_y$ $\Rightarrow$ Measured angular velocity of the body based on the gyroscope, as given by Equation (7.3)

$\hat{\mathbf{c}}_\Omega$ $\Rightarrow$ Estimated integral term as an offset to $\boldsymbol{\Omega}_y$

$\boldsymbol{\Omega}_e$ $\Rightarrow$ Corrective error feedback angular velocity term, as given by Equation (7.13)

k_p $\Rightarrow$ Proportional gain indicating the influence of $\boldsymbol{\Omega}_e$ on $\hat{q}$

k_i $\Rightarrow$ Integral gain indicating the influence of $\boldsymbol{\Omega}_e$ on $\hat{\mathbf{c}}_\Omega$

The PI gains k_p and k_i of the filter should be tuned to provide non-oscillatory yet responsive transients, as limited by sensor noise. Note that mathematically, Equation (7.14a) simply computes the quaternion velocity $\dot{\hat{q}}$ that is equivalent to the effect of the angular velocity

$$\boldsymbol{\Omega}_y - \hat{\mathbf{c}}_\Omega + k_p \boldsymbol{\Omega}_e \tag{7.15}$$

being applied to the quaternion $\hat{q}$. The direct parallel of the 3D filter update equations to the case for the 1D complementary filter in Equation (7.8) is clear to see by equivalating

$$\hat{q} \leftrightarrow \hat{\theta}, \qquad \boldsymbol{\Omega}_y \leftrightarrow \omega_y, \tag{7.16a}$$

$$\hat{\mathbf{c}}_\Omega \leftrightarrow \hat{c}_\omega, \qquad \boldsymbol{\Omega}_e \leftrightarrow \theta_y - \hat{\theta}. \tag{7.16b}$$

Given the filter equations in Equation (7.14), in each time step the variables $\hat{q}$ and $\hat{\mathbf{c}}_\Omega$ are updated based on numerical trapezoidal integration as a function of the measured elapsed time Δt since the previous time step. Given the nominal update cycle time Δt_n of the estimator, it is recommended for robustness purposes to coerce Δt to some interval scaled by Δt_n, in order to avoid large jumps in the estimator states when lags occur, and ensure greater correctness of the gyroscope angular velocity integration in general. In the implementation of the attitude estimator as part of the Humanoid Open Platform ROS Software, the measured elapsed time in each cycle is coerced to the range

$$\Delta t \in \left[0.8\Delta t_n,\, 2.2\Delta t_n\right], \tag{7.17}$$

as these are known to be realistic limits for the true cycle time in all normal cases.

The stability of the passive complementary filter is discussed in detail in Mahony et al. (2008). Theoretical analysis demonstrates that there is a measure zero set in the space of all possible measured rotation and bias errors such that equilibrium exists despite lack of convergence. The equilibrium is unstable however, and the error is locally exponentially stable in all other cases. This set consists of all error states such that $\hat{\mathbf{c}}_\Omega$ is constant and equal to $\boldsymbol{\Omega}_y$, and $\tilde{q}$ is a rotation

by π radians. This pathological set is of no concern however, as it is never reached in any practical situation. Even intentional initialisation of the filter to such an equilibrium state in simulated experiments did not prove to be a problem, as mere arithmetic floating point errors were enough for the divergent dynamics near the pathological set to take over, and force the estimator away from this unstable equilibrium.

7.5 MEASURED ORIENTATION RESOLUTION METHODS

Given the 3D nonlinear passive complementary filter presented in the previous section, the core task that remains is to specify how the measured quaternion orientation q_y is to be constructed from the current accelerometer and magnetometer-based measurements ${}^{B}\tilde{\mathbf{z}}_G$ and ${}^{B}\tilde{\mathbf{m}}$ respectively. The various approaches to doing this are referred to as the measured quaternion orientation resolution methods, and can, if needed, also use the current estimated orientation $\hat{q}$ as an input—for instance to ensure in the case of a lack of information that the generated q_y is as close as possible to $\hat{q}$. The two main implemented resolution methods are described in this section. Details of the additional *ZYX yaw resolution method* can be found in Allgeuer (2020). All three resolution methods are available in the released attitude estimator code (Allgeuer, 2016).

7.5.1 Magnetometer Resolution Method

As can be seen from Equation (7.11), the calculation of the error quaternion $\tilde{q}$, requires knowledge of q_y, the instantaneous measured orientation best fitting the sensor measurements ${}^{B}\tilde{\mathbf{z}}_G$ and ${}^{B}\tilde{\mathbf{m}}$. In general, these two measurements suffice to construct a unique rotation q_y that best fits the given data. If not, $\hat{q}$ is taken as a further input in the place of ${}^{B}\tilde{\mathbf{m}}$, and one of the other resolution methods is used instead.[3] No matter what resolution method is used however, it is a design decision that the quaternion $q_y \equiv {}^{G}_{B}q_y$ must *always* be calculated in such a way that it satisfies

$$ {}^{\tilde{B}}\mathbf{z}_G = {}^{B}\tilde{\mathbf{z}}_G, \tag{7.18} $$

i.e. q_y must always be consistent with the direction of gravity measured by the accelerometer, with only the heading component being decided by either ${}^{B}\tilde{\mathbf{m}}$ or $\hat{q}$.

For the *magnetometer resolution method*, we use ${}^{B}\tilde{\mathbf{m}}$ to resolve the heading of q_y. In order to do this, we need a calibrated value of ${}^{G}\mathbf{m}_e$—the magnetic field vector of the Earth relative to the nominated global reference frame {G}. The calibration of this vector, known as the

3 The *fused yaw resolution method* is strongly recommended over the *ZYX yaw resolution method* in all situations.

reference field vector, is discussed in Allgeuer (2020). For the purposes of the magnetometer resolution method, only the x and y-components of the reference field vector are required, i.e. m_{ex} and m_{ey}, and these are used only to resolve the heading of q_y.

The quaternion q_y is calculated by first constructing the rotation matrix

$$\overset{G}{\tilde{B}}R_y = \begin{bmatrix} \leftarrow & {}^{\tilde{B}}\mathbf{x}_G & \rightarrow \\ \leftarrow & {}^{\tilde{B}}\mathbf{y}_G & \rightarrow \\ \leftarrow & {}^{\tilde{B}}\mathbf{z}_G & \rightarrow \end{bmatrix}, \tag{7.19}$$

and then converting this to quaternion form. The required value of ${}^{\tilde{B}}\mathbf{z}_G$ is already known from Equation (7.18), so it only remains to calculate suitable mutually orthogonal axis vectors ${}^{\tilde{B}}\mathbf{x}_G$ and ${}^{\tilde{B}}\mathbf{y}_G$. From Equation (7.7), ideally we would wish to be able to find ${}^{\tilde{B}}\mathbf{x}_G$ and ${}^{\tilde{B}}\mathbf{y}_G$ such that

$${}^{B}\tilde{\mathbf{m}} = \overset{G}{\tilde{B}}R_y^T \, {}^{G}\mathbf{m}_e, \tag{7.20}$$

but this is not necessarily possible, so instead we minimise the angular difference between the left-hand side and right-hand side vectors of this equation. The angular difference can be seen to be minimised when the respective projections of the two vectors onto the plane perpendicular to ${}^{\tilde{B}}\mathbf{z}_G$ are parallel. The projection of ${}^{B}\tilde{\mathbf{m}}$ can be seen to be

$${}^{\tilde{B}}\hat{\mathbf{m}} = {}^{B}\tilde{\mathbf{m}} - \left({}^{B}\tilde{\mathbf{m}} \cdot {}^{\tilde{B}}\mathbf{z}_G\right){}^{\tilde{B}}\mathbf{z}_G, \tag{7.21}$$

and observing that

$$\overset{G}{\tilde{B}}R_y^T \, {}^{G}\mathbf{m}_e = m_{ex} \, {}^{\tilde{B}}\mathbf{x}_G + m_{ey} \, {}^{\tilde{B}}\mathbf{y}_G + m_{ez} \, {}^{\tilde{B}}\mathbf{z}_G, \tag{7.22}$$

the required projection of $\overset{G}{\tilde{B}}R_y^T \, {}^{G}\mathbf{m}_e$ can be seen to be

$${}^{\tilde{B}}\hat{\mathbf{m}}_e = m_{ex} \, {}^{\tilde{B}}\mathbf{x}_G + m_{ey} \, {}^{\tilde{B}}\mathbf{y}_G. \tag{7.23}$$

Solving these equations for the condition that ${}^{\tilde{B}}\hat{\mathbf{m}}$ and ${}^{\tilde{B}}\hat{\mathbf{m}}_e$ are parallel (i.e. positive multiples of one another) yields

$${}^{\tilde{B}}\mathbf{x}_G = \frac{{}^{\tilde{B}}\tilde{\mathbf{x}}_G}{\|{}^{\tilde{B}}\tilde{\mathbf{x}}_G\|}, \qquad {}^{\tilde{B}}\mathbf{y}_G = \frac{{}^{\tilde{B}}\tilde{\mathbf{y}}_G}{\|{}^{\tilde{B}}\tilde{\mathbf{y}}_G\|}, \tag{7.24}$$

where

$${}^{\tilde{B}}\hat{\mathbf{u}} = {}^{\tilde{B}}\hat{\mathbf{m}} \times {}^{\tilde{B}}\mathbf{z}_G, \tag{7.25a}$$

$${}^{\tilde{B}}\tilde{\mathbf{x}}_G = m_{ex} \, {}^{\tilde{B}}\hat{\mathbf{m}} + m_{ey} \, {}^{\tilde{B}}\hat{\mathbf{u}}, \tag{7.25b}$$

$${}^{\tilde{B}}\tilde{\mathbf{y}}_G = m_{ey} \, {}^{\tilde{B}}\hat{\mathbf{m}} - m_{ex} \, {}^{\tilde{B}}\hat{\mathbf{u}}. \tag{7.25c}$$

Equation (7.19) is then used as previously described to calculate $\overset{G}{\tilde{B}}R_y$ and subsequently q_y. Note that the z-component m_{ez} of the reference

field vector is not required at any point in the calculations. The magnetometer resolution method fails due to a division by zero if $m_{ex} = m_{ey} = 0$, or if ${}^{\tilde{B}}\mathbf{z}_G$ and ${}^{B}\tilde{\mathbf{m}}$ are collinear. Both of these situations should never reasonably occur (away from the Earth's north and south poles), as it corresponds to the Earth's magnetic field being vertical in the global fixed frame—a generally unexpected case.

7.5.2 Fused Yaw Resolution Method

As discussed in Section 6.2, the concept of Euler yaw is not particularly advantageous when it comes to applications relating to balance and heading. The fused yaw on the other hand is, so a resolution method based on this notion of yaw has been developed.

As indicated in Figure 7.1, we define the frame {H} to be the frame {$\tilde{\text{B}}$} rotated by the inverse of $\hat{q}$. That is, {H} corresponds to the current estimated orientation of the global fixed frame. Note that this will not be identical to {G} in general, as $\hat{q}$ and q_y generally differ, even if only slightly. The *fused yaw resolution method* works by zeroing the fused yaw of {H} with respect to {G}, much like the ZYX Euler method (described in Allgeuer, 2020) zeroes the corresponding ZYX Euler yaw. One notable distinction to the ZYX Euler method is that having zero relative fused yaw is in fact a mutual relationship, as the inverse of a rotation has the exact negative of its fused yaw. A second notable distinction is that the notion of fused yaw is more closely related to quaternions than ZYX Euler yaw, and so a convenient direct quaternion formulation exists.

The z-vector ${}^{H}\mathbf{z}_G = (z_{Gx}, z_{Gy}, z_{Gz})$ of the true global frame {G} with respect to the estimated global frame {H} can be calculated to be

$$\begin{aligned} {}^{H}\mathbf{z}_G &= L_{{}^{H}_{\tilde{B}}q}\big({}^{\tilde{B}}\mathbf{z}_G\big) \\ &= L_{\hat{q}}\big({}^{\tilde{B}}\mathbf{z}_G\big) \\ &= \hat{q}\,{}^{\tilde{B}}\mathbf{z}_G\,\hat{q}^*. \end{aligned} \tag{7.26}$$

Temporarily treating quaternions as column vectors in $\mathbb{R}^4$, the fused yaw resolution method can then be summarised mathematically as

$$q_y = \frac{\tilde{q}_y}{\|\tilde{q}_y\|}, \tag{7.27}$$

where

$$\tilde{q}_y = \begin{bmatrix} 1+z_{Gz} & -z_{Gy} & z_{Gx} & 0 \\ z_{Gy} & 1+z_{Gz} & 0 & -z_{Gx} \\ -z_{Gx} & 0 & 1+z_{Gz} & -z_{Gy} \\ 0 & z_{Gx} & z_{Gy} & 1+z_{Gz} \end{bmatrix} \hat{q}. \tag{7.28}$$

From inspection it can be seen that the only case in which the algorithm fails is if ${}^{H}\mathbf{z}_G = (0, 0, -1)$. This occurs if the rotation from

{G} to {H} (or equivalently, vice versa) is at the fused yaw singularity, in which case it can be derived from Equation (7.11) that the error quaternion $\tilde{q} \equiv {}^{E}_{B}q$ must be a rotation by π radians. We recall however, from page 121, that this is the exact condition of the only unstable equilibrium point, i.e. problem case, of the passive complementary filter itself. Thus, we conclude that unlike the ZYX yaw method, the use of the fused yaw resolution method ensures that there is only a single error condition for which any part of the total estimation process yields suboptimal results. Furthermore, this one error condition is when $\tilde{q}$ is at an exact antipode of the identity rotation—a case that in practical situations is never reached. Nevertheless, for reasons of completeness and robustness, the above algorithm falls back to zeroing the ZYX yaw if it fails. This is guaranteed not to fail if the fused yaw algorithm failed. It is important to note that the fused yaw resolution method is equally stable in all global orientations of the body, as its action depends only on the deviation between the two global frames {G} and {H}, and *not* on where in the rotation space the current attitude estimate $\hat{q}$ actually lies.

7.6 EXTENSIONS TO THE ESTIMATOR

Given the complete attitude estimator as described in the chapter thus far, some extensions can be made to improve performance, or allow estimation with less than the full required 9 axes of data.

7.6.1 Quick Learning

It is desired for the attitude estimator to settle quickly from large estimation errors, yet simultaneously provide adequate general noise rejection. To this end, *quick learning*[4] is proposed as a method to help achieve this. Quick learning allows two sets of PI gains to be tuned—one set that provides suitably fast transient response for quick convergence, and one set that provides good tracking and noise rejection for the long term. Given a desired quick learning time, a parameter $\lambda \in [0, 1]$ is then used to fade linearly between these two sets of gains over this time, where the final faded gains are the ones that provide good tracking and noise rejection. The gain fading scheme is mathematically given by

$$\left(k_p, k_i\right) = \lambda\left(k_p^{nom}, k_i^{nom}\right) + (1-\lambda)\left(k_p^{quick}, k_i^{quick}\right). \tag{7.29}$$

Quick learning can be triggered at any time, including automatically when the estimator starts, and is disabled when λ subsequently returns to 1.0 via a linear ramp.

4 This extension to the estimator is referred to as 'quick learning' mostly for historical reasons, and would more accurately be summarised as a 'gain scheduling scheme'.

7.6.2 Estimation without Magnetometer Data

If no magnetometer data is available in a system, the attitude estimator can still be used without any degradation in the estimation quality of the pitch and roll dimensions by setting ${}^{B}\tilde{\mathbf{m}}$ and ${}^{G}\mathbf{m}_e$ to zero. In this case, the magnetometer resolution method directly falls through, and the estimation relies solely on the selected yaw-based orientation resolution method. Due to the yaw-zeroing approach that is used, the open-loop yaw produced by the estimator remains stable with each update of q_y. The component of $\boldsymbol{\Omega}_y$ in the instantaneous direction of $\mathbf{z}_G$ however, does not have any feedback via the corrective error feedback term $\boldsymbol{\Omega}_e$ in this case, so small constant global yaw velocities in $\hat{q}$ can result. Essentially, this is because the integration in Equation (7.14) of the component of $\boldsymbol{\Omega}_y - \hat{\mathbf{c}}_\Omega$ in the direction of $\mathbf{z}_G$ is then 'unconstrained', leading to yaw drifts. Such yaw drifts are unavoidable without magnetometer data however, as no sensory information is then available that can be used to prevent it.

The effect of the yaw drifts on the output quaternion $\hat{q}$ can be eliminated for convenience by removing the fused yaw component of $\hat{q}$ (to give $\hat{q}_t$), using an equation like Equation (5.93). Note that the fused yaw is the one and only notion of yaw for which it is perfectly valid to perform this operation, as when the fused yaw component of a rotation is removed, the tilt rotation component remains, and tilt rotations have the unique property of having a direct one-to-one correspondence to the set of possible accelerometer-measured gravity directions, as discussed in Sections 5.1.2 and 5.7.1. In addition to not being a strictly valid operation, removing the ZYX Euler yaw has sensitivity issues (see Section 6.2.1), and leads to unexpected behaviour near the not uncommon scenario of pitch rotations by $\frac{\pi}{2}$ radians.

7.7 EXPERIMENTAL RESULTS

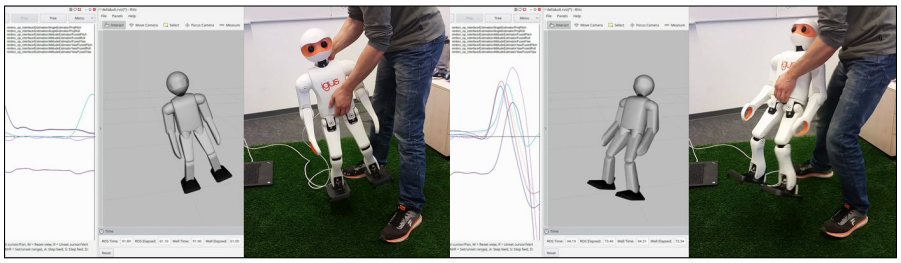
Over the years, thorough experimentation and testing of the proposed attitude estimator (and corresponding C++ implementation) has been performed on multiple different platforms, as well as in simulation. In the context of this thesis, we provide results of the attitude estimator running on a real igus Humanoid Open Platform robot, using specifically the fused yaw resolution method, for mathematical, algorithmic and performance reasons.

Video 7.1 shows the visual recording of an experiment in which an igus Humanoid Open Platform robot was manually rotated in the pitch, roll and yaw directions, followed by various combinations of the three, including laying the robot face down on the floor and picking it up again. Two attitude estimators—one allowed to use magnetometer data and one not—were run in parallel to a single instance of the angle estimator that was used by Missura (2015) for the purpose of capture

steps. The output of the attitude estimator with magnetometer data is shown in the video by means of a 3D robot model in RViz. The model rotates live relative to the visualised grid frame, to show the current estimated robot orientation at all times. Adjoining live data plots can also be seen, indicating the fused angles output of all three orientation estimation methods at once. It can generally be observed from the video that the estimated RViz orientation is virtually indistinguishable from the true motion of the robot, keeping in mind that the former naturally does not show the effect of any translations.

The data that was captured during Video 7.1 is shown in Figure 7.2. As an initial observation, it can be identified that the fused pitch and roll waveforms are nearly identical for the two instances of the attitude estimator. This is expected, as it was one of the core aims of the estimator that the magnetometer *strictly* only affects the fused yaw component of the estimation, and not the balance-critical fused pitch and roll components. Note that a similar statement for the Euler pitch and roll would be invalid (and not expected), as the ZYX Euler roll parameter contains components of 'yaw', as discussed in Section 6.2.2.1. In contrast to the near-inseparable pitch and roll waveforms, the two estimated yaw waveforms can be observed to have a near-constant offset to each other. This is expected, as the estimator that is not using magnetometer data has no sensor data that allows it to discern absolute heading, so it effectively expresses its estimated yaws relative to the pose the robot was in when it was switched on and the estimator started running. Nonetheless, importantly, it can be seen that the final estimated fused yaws for both attitude estimators are very close to the ones that were estimated at the beginning of the experiment. This is expected, as the robot at the end of the experiment was facing in the approximate same direction as it was facing at the beginning of the experiment.

Initially, as the robot is rotated independently about its local y, x and z-axes, it can be seen that the overwhelming estimation responses come in the fused pitch, roll and yaw parameters, respectively. After this, a complicated rotation sequence ensues that effectively rotates the robot by $\approx 90°$ clockwise (CW). It can be extracted from the yaw estimation data that the two attitude estimators quantified this rotation as $-89.6°$ and $-90.6°$, respectively. Given the general circular non-uniformity of the magnetometer measurements and the somewhat approximate nature of the 90° rotation, these two values are quite close to their expected value. The experiment continues by indirectly rotating the robot back to facing forward, and then tipping it forwards onto the ground. It can be observed in the video that the foot of the robot is inadvertently pulled by the right foot of the experimenter as the robot is being placed on the ground. This can be seen to directly explain the disturbances that appear at time $t = 35.5\,\text{s}$.



Video 7.1: Recording of the attitude estimation experiment, comparing the estimated orientation of the robot in RViz with the real orientation of the robot. The captured data is plotted in Figure 7.2.
https://youtu.be/LEkEiFzAVrE
Attitude Estimation Experiment (RViz vs Real)

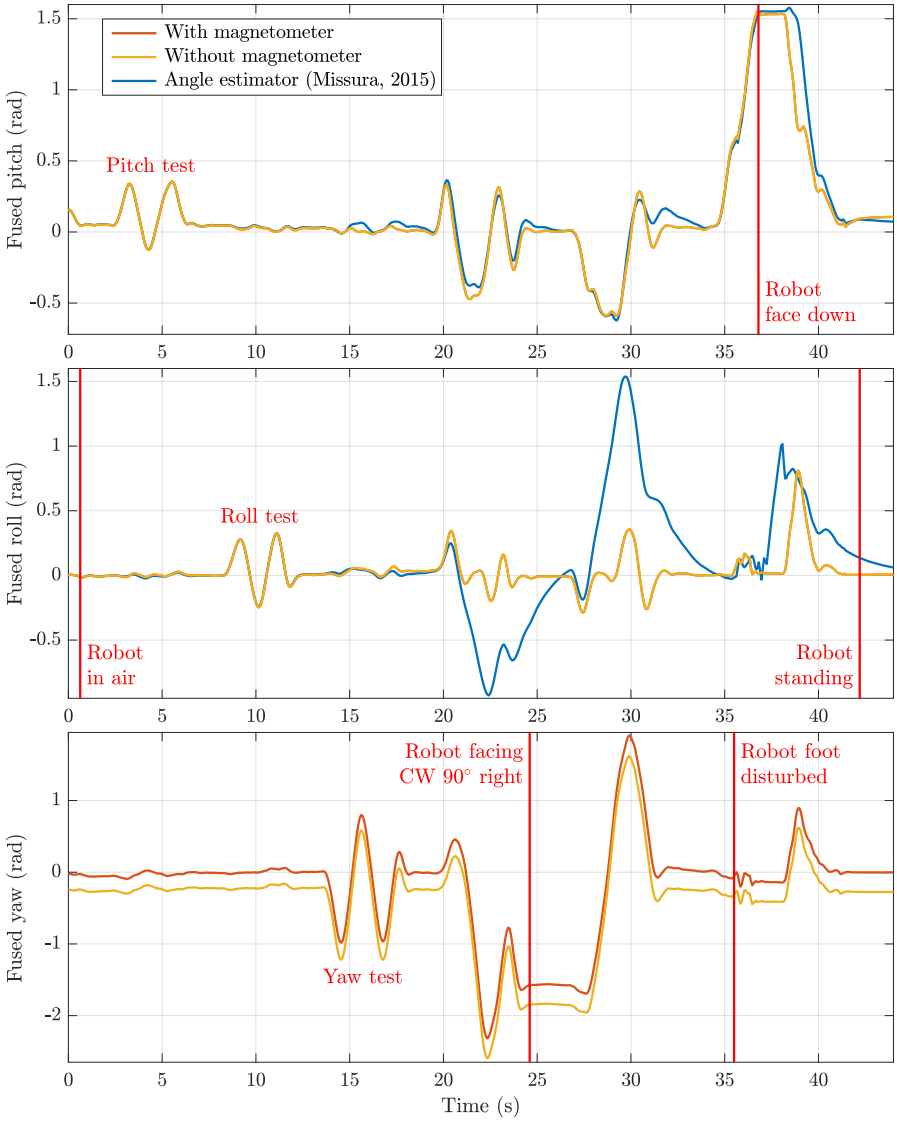


Figure 7.2: Plots of the attitude estimation data captured in Video 7.1. The outputs of the attitude estimation with and without magnetometer measurements are compared to the angle estimator used in Missura (2015) for the purpose of capture steps.

While the pitch and roll values[5] that were estimated by the angle estimator (Missura, 2015) corresponded quite closely to the values estimated by the attitude estimators for very basic motions (i.e. the pitch and roll tests), as soon as these basic motions were combined into more complex ones, the angle estimator severely struggled. While Video 7.1 visually proves that the pitch and roll values calculated by the attitude estimators are quite accurate, it can be seen that as of $t = 20\,\text{s}$, the values calculated by the angle estimator severely diverge from these. At $t = 22.4\,\text{s}$ for example, the estimated angle estimator roll is $-53.2°$, and at $t = 29.7\,\text{s}$ the estimated roll even reaches $88.1°$. Looking at the video, these two estimated rolls are clearly nonsensical, especially seeing as they occur *prior* to the robot being laid down on the ground. A different problem can also be identified around the time $t = 37\,\text{s}$, where the estimated roll becomes somewhat unstable, as evidenced by the sharp jagged edges in the waveform there and the subsequent increase of the estimated roll to $58.2°$ despite the fact that no significant rotation is occurring during that time (see Video 7.1). After the robot is lifted upright again and placed back into a standing position, it can be seen that the angle estimator needs many seconds to recover, as was also the case the previous times the roll had a problem during the experiment. A core problem of the angle estimator is that it fails to capture local z-rotations. This is a non-ignorable issue, as they occur all the time during real walking.

Figure 7.3 shows the results of a further experiment where the robot was rotated $360°$ about the global z-axis, and placed back down on the ground.[6] Once again, a constant offset between the resulting estimated fused yaws was observed, as expected, and the final fused yaws correlated very closely to the initial ones. The sudden jumps in the middle of the plot are attributed to the fused yaws rolling over from $+\pi$ to $-\pi$. Comparing the initial and final estimated fused yaws reveals that the attitude estimator with magnetometer data saw a total change of $-1.2°$, while the estimator without magnetometer data saw a total change of $-0.9°$. These two results are in relative close correlation to each other, and it is thinkable that the true rotation that was undergone by the robot was actually only accurate to $\pm 1°$, seeing as ultimately the performed rotation was judged visually (using a fixed straight edge as a guide) by the experimenter.

While Figure 7.3 demonstrates that the short- to medium-term performance of the estimated fused yaw is reliable and stable, even without magnetometer data, the long-term performance still needs to be ascertained. This has been done in Figure 7.4, which shows the results of a long-term yaw drift test for a robot that was left untouched

5 Note that the angle estimator only estimates pitch and roll, and does not provide any estimate of yaw.

6 Note that the robot was in fact overrotated (around time $t = 13\,\text{s}$), and brought back to the required pose after that. The final adjustments to the robot happened slowly in the time period from $t = 16\,\text{s}$ to $20\,\text{s}$.

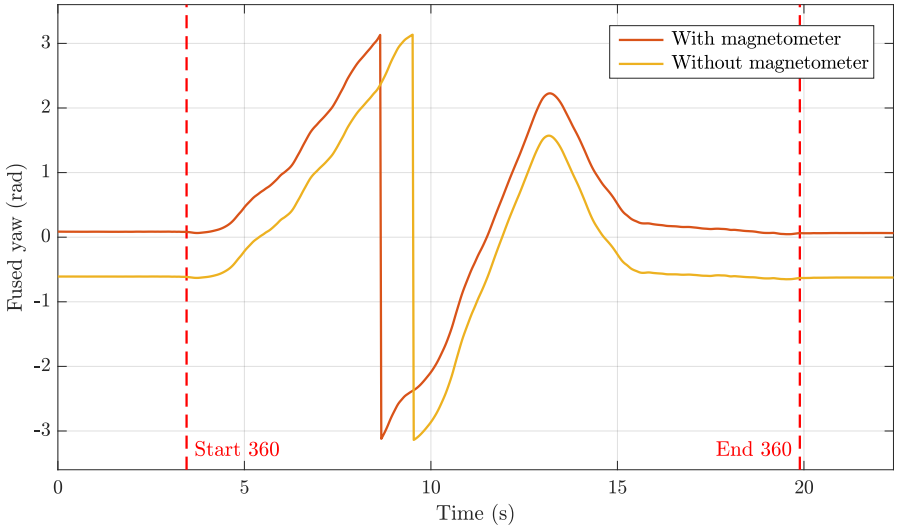


Figure 7.3: Estimated fused yaw with and without magnetometer data for a 360° yaw rotation of the robot. The rotation was performed visually by the experimenter with the aid of a fixed straight edge, and the resulting changes in yaw were measured to be −361.2° and −360.9°, respectively. Given the level of agreement between these results, it is thinkable that the true rotation performed by the experimenter was only accurate to ±1°.

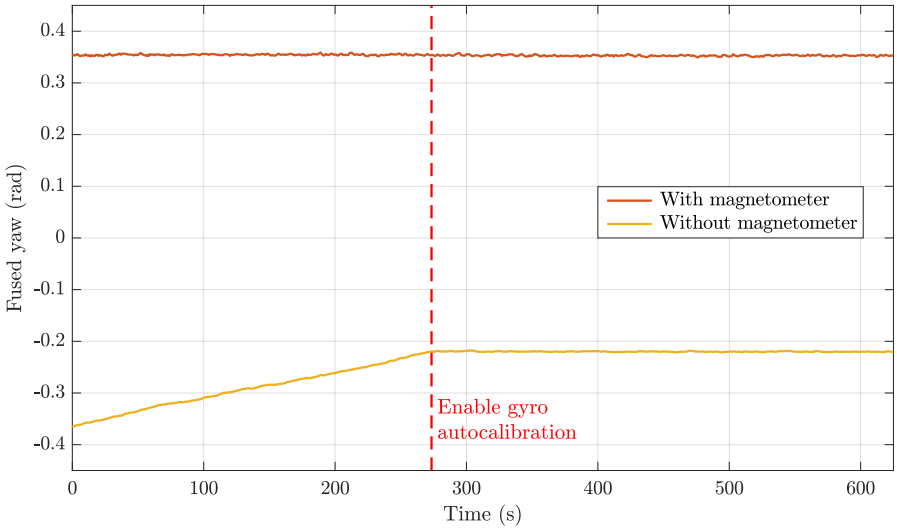


Figure 7.4: Plot of the long-term yaw estimation accuracy of the attitude estimator, demonstrating the possibility of yaw drift in the case of magnetometer-free estimation. The robot was kept stationary in a standing position for the whole experiment, and at $t = 273.4\,\mathrm{s}$ online gyroscope bias autocalibration (see Section 4.1.4) was enabled. While the yaw drift prior to autocalibration was ≈0.031°/s, afterwards the yaw was stable at <0.0001°/s drift.

in a standing position for more than 10 minutes. It can be seen that the attitude estimator with magnetometer data is not prone to yaw drift, as a properly calibrated magnetometer provides an absolute reference of heading that can be used to maintain a consistent estimate of fused yaw. The attitude estimator without magnetometer data however, relies solely on 3D open-loop integration of the gyroscope to provide a reasonable estimate of fused yaw, and is thus vulnerable to drift.[7] Normally, online gyroscope bias autocalibration (see Section 4.1.4) is always enabled, but initially in Figure 7.4 when it was disabled, a yaw drift of 0.031°/s (32.3 s per degree) was observed. This is because the gyroscope bias is not truly constant over time, so a one-time gyroscope bias calibration always eventually leads to residual errors that cause yaw drift.[8] When the online gyroscope bias autocalibration scheme was enabled at time $t = 273.4$ s however, the yaw drift immediately halted, and reduced to below the measurable threshold of 0.0001°/s (0.36°/h). It is important to note that while the autocalibration scheme only activates when the robot is stationary, it nevertheless ensures that up to the very instant the robot starts moving again, the estimated gyroscope bias is completely correct.

It can be observed from Figure 7.4 that the fused yaw estimated using magnetometer data is noisier than the fused yaw estimated without. This is because the magnetometer is an inherently noisy sensor, that when the robot is moving actually gets even noisier due to the magnetic and electromagnetic disturbances caused by the servos. Deviations of the local magnetic field from the Earth's magnetic field are also frequent occurrences in buildings, where electrical mains and the presence of larger metal objects and structures are commonplace. Ultimately, the use of magnetometer data in the attitude estimator allows absolute estimates of heading to be made, at the cost of increased noise and decreased short-term stability of the fused yaw.

The effect of quick learning (refer to Section 7.6.1) is shown in Figure 7.5. While it is nominally only active in the first few seconds after initialisation, it can also be relevant if the attitude estimator is set to a particular orientation, or reset. In the experiment in the figure, a running attitude estimator was reset twice to the identity rotation, once with quick learning enabled, and once with it disabled. The difference in time that it took for the estimator to converge to the orientation of the robot with quick learning disabled is very apparent, at around 8.8 s as compared to 0.36 s. The effect of the increased gains

7 The estimated fused pitch and roll cannot drift in any situation, as the accelerometer data provides an absolute reference of tilt. The accelerometer data however cannot be used to resolve heading, as a change in heading leaves the measured gravitational acceleration constant.

8 Note that it would be possible to simply 'ignore' gyroscope measurements smaller than a given threshold, and this would be an easy way to 'cheat' an eternally stable fused yaw, but this only helps as long as the robot is completely stationary, and does not do anything to solve the problem that the gyroscope bias is still wrong when the robot subsequently moves.

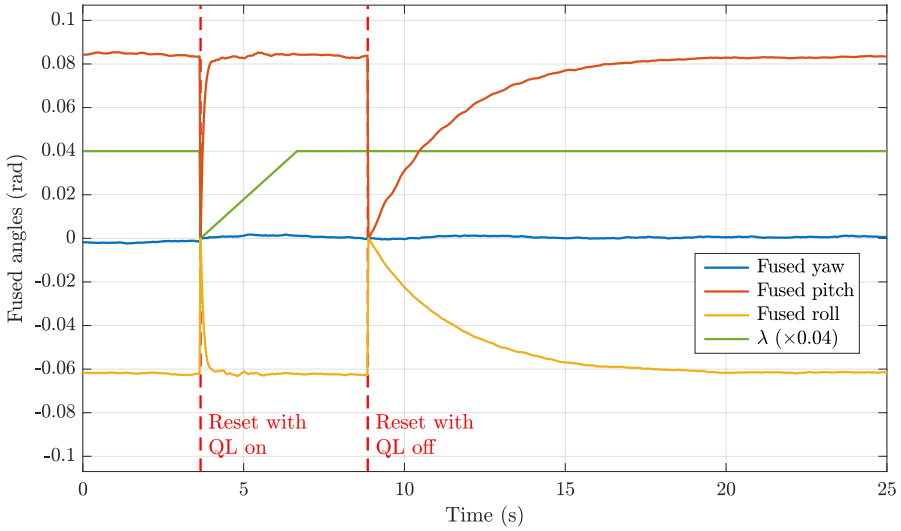


Figure 7.5: The effect of quick learning on the settling time of the attitude estimator. At times $t = 3.66\,\text{s}$ and $t = 8.86\,\text{s}$, the attitude estimator is reset to the identity rotation, and thereby forced to converge again to the true orientation of the robot. Quick learning is enabled ($\lambda = 0$) for the first reset, but disabled for the second. The resulting difference in settling time (0.36 s vs. 8.8 s) is apparent.

during quick learning can be seen to temporarily increase the amount of noise in the estimated fused pitch and roll around the $t = 5{-}6\,\text{s}$ mark, but as λ returns to 1 and the gains return to normal, this can be seen to smoothen out again. Note that the robot was being held in the air during the experiment, so it is not expected that the robot's orientation is perfectly constant throughout. With quick learning, the knowledge that the current orientation estimate (after reset) is likely not accurate allows larger gains to be reasonably employed to reduce the convergence time of the estimator, while ensuring all the while that the gains during normal operation can remain small enough that the excessive influence of accelerometer and magnetometer noise can be avoided.

The attitude estimator was designed to be able to run at high loop rates on embedded hardware, so as to minimise estimation and possible control feedback latencies. The C++ attitude estimator library code was tested on a Personal Computer (PC) with a 2.40 GHz Intel i5-2430M processor. On a single Central Processing Unit (CPU) core, the average execution time of the estimator over 100 million cycles was found to be 127.6 ns for the magnetometer method, 144.3 ns for the ZYX yaw method (see Allgeuer, 2020), and 112.3 ns for the fused yaw method. It is to be expected that the fused yaw method takes comparatively less time, as it does not in general require a rotation matrix to quaternion conversion, unlike the other two methods. From these results it is confidently anticipated that the algorithm is efficient enough to be implemented at high execution rates on a low-cost

microcontroller, where floating and/or fixed point operations are comparatively more expensive than on a PC.

7.8 DISCUSSION

The presented attitude estimator has been used, completely unchanged, on a wide variety of different robots for many years, including in the demanding environment of the RoboCup competition. In the early years, when magnetometers were still allowed in the competition, the magnetometer resolution method was the main method in use. As of 2016 however, when magnetometers were disallowed, the fused yaw method took its place, and proved to work very well in estimating the pitch and roll of the trunk of the robot at all times. Paired up with the gyroscope scale calibration (see Section 4.1.2) and online gyroscope bias autocalibration scheme (see Section 4.1.4), it was also possible to derive a relatively reliable and low-drift estimate of the yaw, despite it essentially being dependent on open-loop integration. The scale calibration of the gyroscope also contributed in allowing lower values of the proportional gain k_p to be chosen, leading to reduced delay times when tracking sharp changes in orientation, and generally superior transient response characteristics.

One of the most important features of the attitude estimator is that no matter which measured orientation resolution method is used, the filter is equally stable in all global body orientations, and only demonstrates potential non-convergent behaviour on a pathological set that is of no practical concern. Extensions to the filter also allow for reliable attitude estimation in situations of reduced sensory data, and the quick learning feature allows for shorter settling times from large estimation errors when required.

The output of the estimator can be used flexibly depending on the situation, in the form of either

- $\hat{q} \in \mathbb{Q}$, the full quaternion orientation estimate of the robot,
- $\hat{q}_t \in \mathbb{Q}$, the (pitch/roll) tilt rotation component of the robot, expressed as a zero z-component quaternion,
- $\hat{F} = (\psi, \theta, \phi, h) \in \mathbb{F}$, the individual fused angles parameters of $\hat{q}$ (see Section 5.3.4), or,
- $\hat{P} = (p_x, p_y, p_z) \in \mathbb{P}^3$, the individual tilt phase space parameters of $\hat{q}$ (see Section 5.3.5).

In these various forms, the estimated orientation output of the attitude estimator can be used for all kinds of feedback purposes, including in particular the analysis and control of balancing biped robots.

8

A CENTRAL PATTERN GENERATOR FOR WALKING

For most bipedal robot platforms, completely open-loop walking—for at least short periods of time—is reasonably possible with a moderate level of stability. This is because it is usually possible to design walking motions that take advantage of the small basin of passive stability provided by the foot stiffnesses and support polygons. The Central Pattern Generator (CPG) is one such walking motion, and is sufficiently generic and tuneable that it has proven to work, and be effective on, a wide range of different robots. The general approach to walking used by the bipedal gaits in this thesis is to start with an open-loop gait core, and build around it stabilising feedback mechanisms that adjust the gait step sizes, timing, and other specifics of the generated waveforms. The CPG that is introduced in this chapter is used as the basis of the direct fused angle feedback controller gait in Chapter 9, and has been implemented in the `cap_gait` gait engine of the Humanoid Open Platform ROS Software.[1]

The CPG is based on the general ideas that were present in the CPG used by Missura (2015), but with many modifications and improvements that were mainly intended to allow the generator to be more flexible and work on a wider variety of robots. The main differences to the older CPG, also described in Missura and Behnke (2013), include:

- Changes to the leg retraction profiles to transition more smoothly between swing and support phases,
- The addition of a double support phase for greater walking stability and passive oscillation damping,
- The addition of a trim factor for the angle relative to the ground at which the feet are lifted during stepping,
- The integration of a dynamic pose blending algorithm to enable smoother transitions to and from walking,
- The incorporation of support coefficient waveforms, for use with the actuator control scheme (see Chapter 3),
- The introduction of a leaning strategy based on the rate of change of the commanded gait velocity, and
- The use of hip motions instead of leg angle motions for the gait command velocity-based leaning strategies.

1 https://github.com/AIS-Bonn/humanoid_op_ros/blob/master/src/nimbro/motion/gait_engines/cap_gait/src/cap_gait.cpp

8.1 CPG GAIT INTERFACES

The Central Pattern Generator (CPG) in its purest form can simply be considered to be a complex high-dimensional function with certain inputs and outputs, i.e. a function like

$$f_{CPG} : \{\mathbf{v}_g, \mu_i\} \mapsto \{\mathbf{q}_o, \boldsymbol{\xi}, \kappa_l, \kappa_r\} \tag{8.1}$$

where the individual input and output variables are as described in the following subsections.

8.1.1 CPG Gait Inputs

One of the two inputs to the open-loop CPG gait is the dimensionless Gait Command Velocity (GCV) vector

$$\mathbf{v}_g = (v_{gx}, v_{gy}, v_{gz}) \in [-1, 1]^3. \tag{8.2}$$

The components of the input GCV vector $\mathbf{v}_g$ specify the desired walking speed of the robot in the sagittal, lateral and yaw rotation directions respectively, as a ratio of the corresponding maximum allowed speeds. For instance, full speed forwards walking corresponds to a GCV of $\mathbf{v}_g = (1, 0, 0)$. The raw input GCV vector is first scaled, if required, to ensure that its p-norm satisfies

$$\|\mathbf{v}_g\|_p = \left(|v_{gx}|^p + |v_{gy}|^p + |v_{gz}|^p\right)^{\frac{1}{p}} \leq 1, \tag{8.3}$$

and the resulting vector is slope-limited to ensure that no sharp changes in walking speed can occur, yielding the internal GCV vector

$$\mathbf{v}_i = (v_{ix}, v_{iy}, v_{iz}). \tag{8.4}$$

In general, a value of p in the range $[2, 2.5]$ was found to be suitable for the tested robots. The p-norm limit ensures, for stability reasons, that the robot cannot for instance try to walk full speed forwards and sidewards at the same time, and the slope limit ensures that no jumps or step changes can occur in the CPG outputs.

The second input to the CPG gait is the gait phase variable

$$\mu_i \in (-\pi, \pi]. \tag{8.5}$$

This variable parameterises the gait cycle that is generated by the CPG in a direct and deterministic way, and for regular open-loop walking starts at 0 or π at the beginning of walking, and is updated after each time step Δt using

$$\mu_i \leftarrow \text{wrap}(\mu_i + \pi f_g \Delta t), \tag{8.6}$$

where f_g is the configured nominal gait frequency in units of steps per second. The wrap$(\cdot)$ function ensures that the gait phase stays strictly in the range $(-\pi, \pi]$ at all times. Each value of μ_i corresponds to a fixed instant of the gait cycle. For example,

$$\mu_i = 0 \;\Rightarrow\; \text{Foot strike of the right leg, beginning of double support}$$
$$\mu_i = \pi \;\Rightarrow\; \text{Foot strike of the left leg, beginning of double support}$$

It should be noted that by convention, the swing phase of the left leg and support phase of the right leg occur in the phase range $\mu_i \in (0, \pi]$, and the *support phase* of the left leg and *swing phase* of the right leg occur in the phase range $\mu_i \in (-\pi, 0]$. As the arms generally swing opposite to the legs during walking, the swing and support phase definitions for the arms are exactly opposite to those for the legs.

8.1.2 CPG Gait Outputs

The outputs of the CPG gait are exactly those required by the actuator control scheme (see Chapter 3). In each execution cycle this corresponds to

- The vector of joint position commands $\mathbf{q}_o \in \mathbb{R}^N$, specifying the desired angular position of each joint (for N joints),
- The vector of joint effort commands $\boldsymbol{\xi} \in [0, 1]^N$, specifying how stiff each joint should be, and,
- The required support coefficients κ_l and κ_r of the left and right legs respectively, specifying the proportion of the weight of the robot that is expected to be supported by each leg.

The vector $\mathbf{q}_o$ can be seen to correspond directly to the desired joint space pose of the robot, and like the support coefficients κ_l and κ_r, it is calculated dynamically based on the GCV and gait phase inputs. The joint effort command vector $\boldsymbol{\xi}$, on the other hand, is generally assigned a manually configured constant value, and is thus trivial to compute in each cycle.

8.1.3 Provisions for Closed-loop Feedback

As discussed at the beginning of this chapter, we wish to construct the CPG in such a way that it is possible for higher level balance controllers to add stabilising feedback to the generated walking motions, with the aim of making them fundamentally more robust. The CPG thus makes provisions in particular for the inclusion of timing and step size feedback.

8.1.3.1 *Timing Feedback*

A balance feedback controller can apply timing feedback to the CPG through the specification of a gait frequency offset f_{go}, time to step t_s, and corresponding support leg sign $\delta_t \in \{-1, 1\}$.[2] The t_s parameter specifies the amount of time that should be taken by the CPG until the foot opposite to δ_t should next be commanded to strike the ground. The current support foot according to the current gait phase μ_i is given by

$$\delta_c = \begin{cases} -1 & \text{if } \mu_i \in (0, \pi], \\ 1 & \text{if } \mu_i \in (-\pi, 0]. \end{cases} \tag{8.7}$$

Thus, the amount of gait phase $\mu_r \in [0, 2\pi)$ that needs to be covered in the specified duration of t_s seconds is

$$\mu_r = \begin{cases} -\mu_i & \text{if } \delta_t = \delta_c = 1, \\ \pi - \mu_i & \text{if } \delta_t = -1, \\ 2\pi - \mu_i & \text{if } \delta_t = 1 \text{ and } \delta_c = -1. \end{cases} \tag{8.8}$$

This corresponds to an instantaneous gait frequency, in steps per second, of

$$\tilde{f}_g = \frac{\mu_r}{\pi t_s} + f_{go}, \tag{8.9}$$

where the effect of the gait frequency offset parameter f_{go} has also been included. After application of a maximum gait frequency bound $f_{g,max}$, i.e.

$$f_g = \text{coerce}\big(\tilde{f}_g, 0, f_{g,max}\big), \tag{8.10}$$

Equation (8.6) can be used as normal to update the gait phase in each cycle. The final instantaneous predicted time to step is

$$\hat{t}_s = \frac{\mu_r}{\pi f_g}. \tag{8.11}$$

Note that a feedback controller may return a different t_s in each cycle, so the amount that μ_i is incremented by with each update may vary continuously, and differ at each instant throughout a single step.

8.1.3.2 *Step Size Feedback*

In the framework of the CPG, step size feedback is realised via modification of the internal GCV vector $\mathbf{v}_i$. Balance feedback controllers can either simply apply additive offsets to the value of $\mathbf{v}_i$, or intercept its calculation prior to slope limiting (but after p-norm limiting), and calculate an arbitrary desired value $\tilde{\mathbf{v}}_i$ based on that intermediate value and the current balance state. In the latter case, the slope limiting is

2 $\delta_t = 1$ corresponds to the left leg, $\delta_t = -1$ corresponds to the right leg.

skipped because otherwise the ability of the controller to make quick balance recovery steps is inhibited. To prevent discontinuities in the gait outputs from occurring as a result, the internal GCV vector $\mathbf{v}_i$ is made persistent and updated in each cycle using the equation

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \frac{\Delta t}{\hat{t}_s}(\tilde{\mathbf{v}}_i - \mathbf{v}_i), \tag{8.12}$$

where $\hat{t}_s \geq \Delta t$ is the final predicted time to step as in Equation (8.11), and Δt is the nominal cycle time. This equation effectively adjusts the current internal GCV so that it reaches $\tilde{\mathbf{v}}_i$ exactly at the end of the current step.

No matter which method of step size adjustment is used, the final value of $\mathbf{v}_i$ is the one that is passed on to the internals of the CPG, and used to generate the required joint waveforms.

8.2 CPG MOTION GENERATION

For mathematical and algorithmic details of the generation of the CPG walking motions, including the leg lifting, leg swing, arm swing, hip swing and leaning components, refer to Allgeuer (2020).

8.3 EXPERIMENTAL RESULTS

The Central Pattern Generator (CPG) has been successfully implemented on all of the robots used throughout this thesis, including in particular the igus Humanoid Open Platform, NimbRo-OP2 and NimbRo-OP2X. The level of stability that the open-loop gait has varies with the mechanical quality of the robot and whether serial or parallel leg kinematics are present, but all tested robots were able to walk with the CPG with a moderate to high quality. Clearly, without any balance feedback there is a limit to the level of disturbances (and self-disturbances) that can be passively rejected by the gait, but the self-stability of the CPG was found to be sufficient in all robots for moderate durations of walking over flat terrains. For conservative input GCV vectors, the most common mode of failure is if a timing offset is induced in the balance of the robot, and the robot ends up trying to lift a leg that it is currently standing on.

Due to the moderately large number of parameters of the CPG, it may at first seem like it is difficult to tune, but seeing as most of the parameters were intentionally designed to be dimensionless and highly independent in terms of their effect, this is not truly the case. The process of tuning starts with the tuning of the halt pose, i.e. the base pose of the gait. The halt pose should be a stable and symmetric standing pose of the robot, with a small yet non-zero amount of leg retraction. In the halt pose, the balance of the robot should be suitably centred above the support polygon defined by its feet, and in the

case of serial kinematics robots, should incorporate a slight forwards tilt of the torso. The leg lifting and hip swing parameters are first tuned to achieve stable walking on the spot, before experiments with pure steady-state sagittal, lateral and rotational walking motions are used to tune the leg swing and arm swing parameters. The leaning parameters come last as they depend on the behaviour of the robot when it is accelerating in the sagittal direction, which requires tests with changing GCVs. Once the parameters of the CPG have been tuned for one robot, carrying them across to another robot is usually a simple task, and requires only very minimal changes to the parameters, if at all, for the gait to work. Usually, even between robots of different shapes and sizes, only the halt pose needs to be adjusted, and at most the configured step height parameters as well.

Figure 8.1 shows plots of the generated joint space waveforms for the left and right limbs of an igus Humanoid Open Platform robot. When the CPG gait is triggered at time $t = 0.65\,\text{s}$, no pre-gait blending is required as the robot is already in the halt pose. After π radians (i.e. 417 ms) of pose blending, and an overlapping 1.2 s of forced zero internal GCV time, a step change in $\mathbf{v}_g$ occurs to the commanded value of $(0.7, 0.4, 0.4)$. When this happens, the slope limiting of the internal GCV $\mathbf{v}_i$ ensures that it remains continuous in its approach of $\mathbf{v}_g$, as can be seen from the resulting linear ramps in the top plot of the figure. After a few seconds of steady-state limit cycle walking, at $t = 6.3\,\text{s}$, the trigger comes that the robot should stop walking. It cannot do this immediately, as it is in the middle of taking a large step and is currently walking with a significantly non-zero GCV, but over the next 2–3 steps, the internal GCV is faded linearly back down to zero. Once $\mathbf{v}_i$ arrives at zero, at the next culmination of a step ($\mu_i = 0$ or π) the CPG waveforms are stopped, and after 417 ms of further pose blending back to the halt pose, the CPG gait releases control of the robot.

A video of the CPG gait running on various kinematic simulations, physical simulations and real robots is provided in Video 8.1. The quality of the gait without feedback is not perfect, and the robots are prone to falling if disturbed, but the CPG serves as a reliable foundation on which robust gaits like the capture step gait (see Allgeuer, 2020 for use of this CPG as part of a capture step gait) and direct fused angle feedback controller (see Chapter 9) can be built.

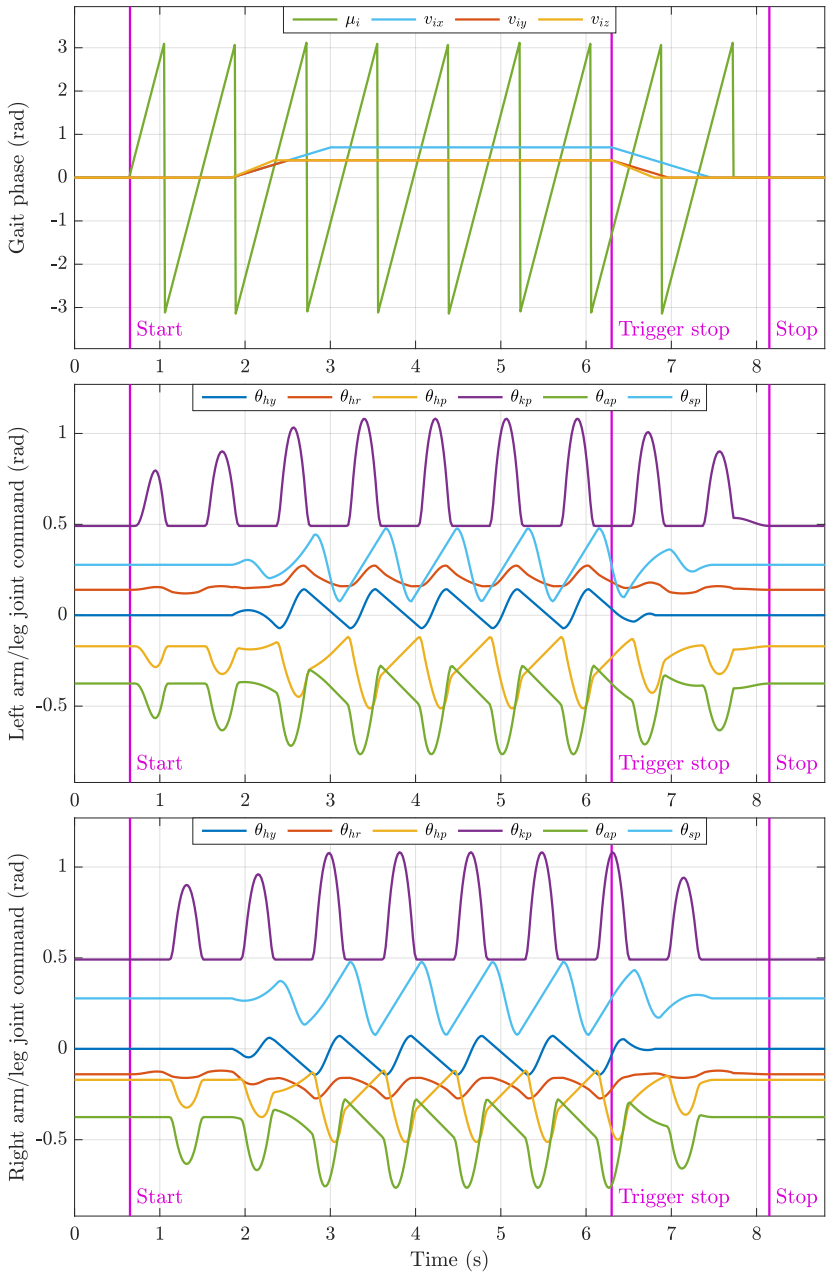
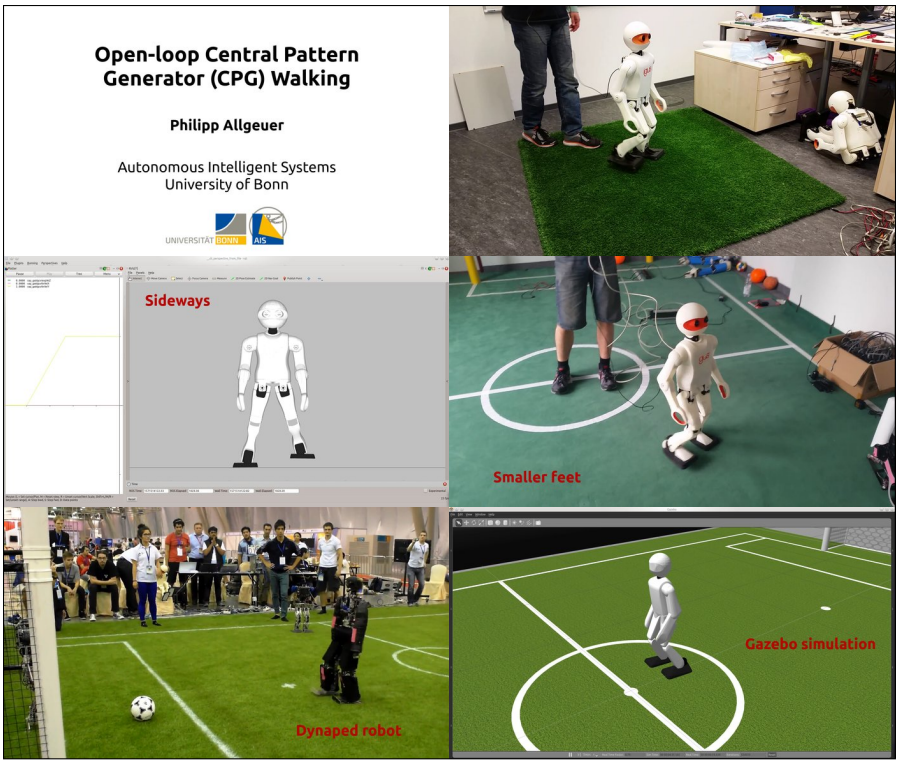


Figure 8.1: Sample output waveforms of the Central Pattern Generator (CPG) for an input GCV of $\mathbf{v}_g = (0.7, 0.4, 0.4)$ and a nominal gait frequency of $f_g = 2.4\,\text{Hz}$. The CPG gait is activated at time $t = 0.65\,\text{s}$, and receives the trigger to stop walking at time $t = 6.3\,\text{s}$. After a further two steps, during which the robot slows down to an internal GCV of zero, the robot stops walking and fades back to the halt pose. Note how the waveforms for the left limbs (middle plot) are in general in exact antiphase to those for the right limbs (bottom plot). This can most clearly be identified in the plots of θ_{kp}, the knee pitch angles. The top plot shows the values of the gait phase μ_i and internal GCV vector $\mathbf{v}_i$ used for the generation of the waveforms. The effect of the initial GCV zero time, in addition to the effect of the GCV slope limiting, can be seen in the plot.



Video 8.1: Demonstration of the open-loop Central Pattern Generator (CPG) gait on the igus Humanoid Open Platform, Dynaped and NimbRo-OP2 robots. The gait is demonstrated in kinematic simulation, physical simulation, and on the real robot hardware.
https://youtu.be/ksJRwGlSuTM
Open-loop Central Pattern Generator (CPG) Walking

9 DIRECT FUSED ANGLE FEEDBACK CONTROLLER

This chapter presents a novel balance controller, referred to as the direct fused angle feedback controller. It does not make any notable adjustments to the step sizes, but nonetheless has a strong stabilising effect on the open-loop Central Pattern Generator (CPG) walking motions of the robot, through the use of timing adjustments and so-called corrective actions, as indicated in Figure 9.1. Corrective actions are inline modifications to the waveforms generated by the CPG, and are computed as a function of the trunk orientation feedback received from the attitude estimation in the form of *fused angles* (see Section 5.3.4). As the direct fused angle controller works by mechanisms other than step size adjustment, aside from the choice of which method of timing adjustment to use, it can coexist entirely and be used in conjunction with the capture step controller if desired (see Allgeuer, 2020). This option for coexistence is demonstrated in the released implementation of both controllers (and the CPG gait), as all three of them are implemented in the same gait engine.[1]

It is demonstrated by the direct fused angle feedback controller in this chapter that genuinely good walking results can be achieved using relatively simple feedback mechanisms, with only minimal modelling of the robot, and without measuring or controlling forces or torques. Only joint positions and a 6-axis Inertial Measurement Unit (IMU) are required for this method to work, making it flexible, portable and reliable for robots of all sizes, including in particular those with low-cost actuators and sensors. The main take-away message of the direct fused angle controller is that simple robot-agnostic feedback mechanisms are enough to make a robot walk if the sensor and feedback chains are carefully constructed, and that the feedback mechanisms can easily be run in parallel with more complicated step size adjustment schemes if desired, for greater overall walking robustness.

9.1 GAIT STRUCTURE

The direct fused angle controller takes as input the estimated fused pitch θ_B and fused roll ϕ_B (see Section 5.3.4) of the trunk of the robot, as estimated by the attitude estimator (see Chapter 7). Based on the deviations of these two angles from their nominal expected waveforms, the controller calculates a gait frequency offset f_{go} in

1 https://github.com/AIS-Bonn/humanoid_op_ros/tree/master/src/nimbro/motion/gait_engines/cap_gait

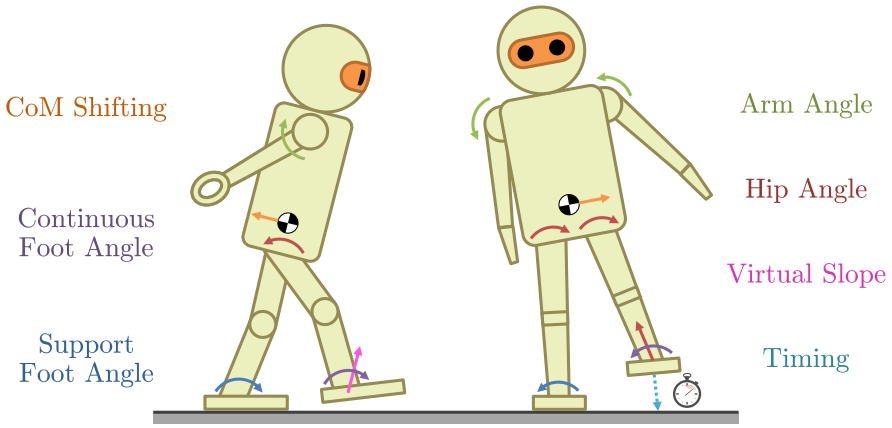


Figure 9.1: Summary of the timing feedback and corrective actions implemented as part of the direct fused angle feedback controller. For more details on each individual corrective action, refer to Section 9.2 and Figure 9.2.

Hertz for the purpose of CPG timing feedback as per Equation (8.9) (see Section 8.1.3.1), and implements the required corrective actions using additive components to the CPG-generated waveforms. The final calculated CPG joint commands are then passed to the actuator control scheme as normal.

9.2 CORRECTIVE ACTIONS

Numerous different corrective actions have been implemented in the direct fused angle controller. Corrective actions can be thought of as motion primitives that are dynamically weighted and superimposed on top of the open-loop CPG gait waveforms, so as to affect or bias the balance of the robot in some way. Six different corrective actions (mostly illustrated in Figure 9.2) have been implemented in a mix of the abstract and inverse pose spaces (Allgeuer, 2020). The implemented corrective actions are as follows:

1. **Arm angle:** The abstract arm angles ϕ_{ax}, ϕ_{ay} are adjusted to bias the balance of the robot and produce reaction moments that help counterbalance transient instabilities. As an example, if both arms are moved backwards, the balance of the robot is tendentially shifted backwards as a result.

2. **Hip angle:** The torso of the robot is tilted within the lateral and sagittal planes to induce lean in a particular direction. As for the lateral leaning component of the CPG, the leg retraction parameter of one of the legs is trimmed to ensure that no disparity in foot height occurs as a result.

3. **Continuous foot angle:** Continuous offsets are applied to the abstract foot angles ϕ_{fx}, ϕ_{fy} to bias the tilt of the entire robot

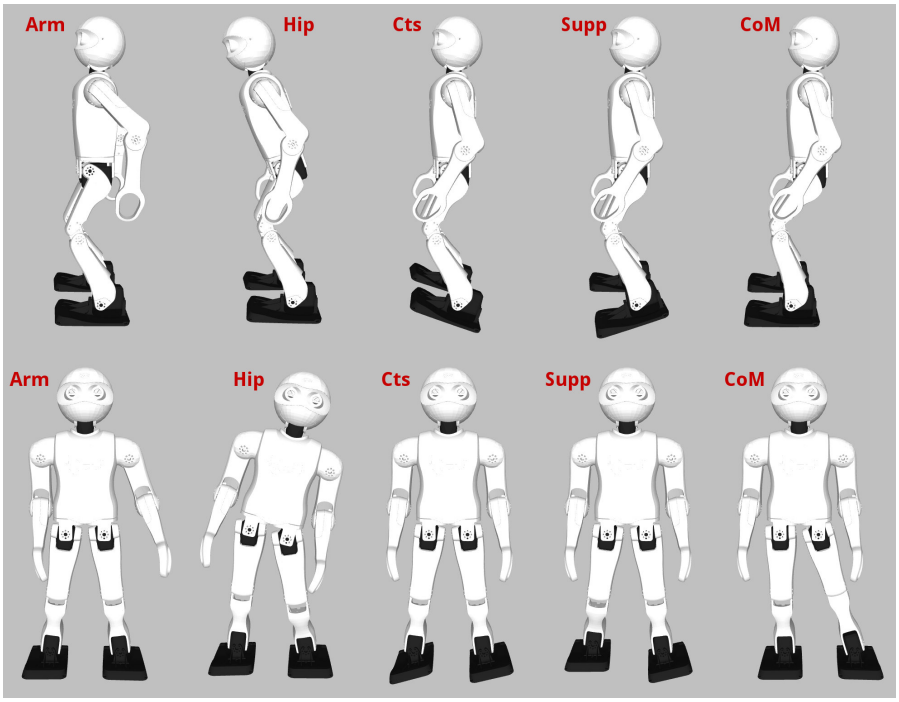


Figure 9.2: The implemented corrective actions in both the sagittal (top row) and lateral (bottom row) planes—from left to right in both cases, the arm angle, hip angle, continuous foot angle, support foot angle, and CoM shifting corrective actions. The actions in this figure have been exaggerated for the purposes of clearer illustration.

from the feet up. For example, if the feet are both offset so that the toe area points further downwards, the balance of the robot is biased in the backwards direction.

4. **Support foot angle:** Gait phase-dependent offsets are applied to the abstract foot angles ϕ_{fx}, ϕ_{fy} of the support foot, in order to induce ground reaction forces that push the balance of the robot in the opposite direction. The offsets are faded in linearly starting at the instant of foot strike, and are faded out linearly just in time for leg lift-off. As such, the support foot angle offsets are only applied to each foot during their single and double support phases, and not during their swing phase.

5. **CoM shifting:** The inverse kinematic positions of the feet relative to the torso are adjusted in the horizontal plane to shift the position of the Centre of Mass (CoM). The centring of the robot's mass above its support polygon is thereby adjusted, influencing its balance. As an example of such influence, shifting the CoM to the right for instance increases the proportion of time spent on the right foot during walking.

6. **Virtual slope:** The inverse kinematic positions of the feet relative to the torso are adjusted in the vertical direction in a gait phase-

dependent manner to lift the feet more at one swing extremity. This can be thought of as what the robot would need to do in order to walk up or down a slope.

9.3 FUSED ANGLE FEEDBACK MECHANISMS

Each of the aforementioned corrective actions (and timing feedback) are activated and driven based on an array of feedback mechanisms that use the estimated fused pitch and roll of the torso as their source. Each feedback mechanism calculates its contribution to the activations of the corrective actions as a function of the deviations of the fused pitch θ_B and fused roll ϕ_B angles from their nominal limit cycle values. The nominal limit cycles of the fused angles are modelled as parameterised sine functions of the gait phase μ_i, and the fused angle deviations d_θ, d_ϕ are defined as the difference between θ_B, ϕ_B and their current expected values θ_{exp}, ϕ_{exp}. That is,

$$d_\theta = \theta_B - \theta_{exp}, \tag{9.1a}$$

$$d_\phi = \phi_B - \phi_{exp}, \tag{9.1b}$$

where

$$\theta_{exp} = k_{eoy} + k_{emy} \sin(\mu_i - k_{epy}), \tag{9.2a}$$

$$\phi_{exp} = k_{eox} + k_{emx} \sin(\mu_i - k_{epx}), \tag{9.2b}$$

where k_* are constants that are tuned based on CPG walking experiments to define the expected fused angle waveforms.

An overview of the complete feedback mechanism calculation pipeline, from the fused angle deviations d_θ, d_ϕ through to the resulting corrective actions and timing adjustment, is shown in Figure 9.3. The algorithmic details of the various feedback mechanisms are discussed in the following subsections. While the timing adjustment ends with the calculation of a gait frequency offset f_{go}, and the virtual slope ends with the calculation of the required inverse kinematic adjustment of the foot locations, the remaining three feedback mechanisms—namely the proportional, derivative and integral feedback mechanisms—are responsible for the activation of the remaining five corrective actions (i.e. the ones in Figure 9.2). This is done in each execution cycle of the direct fused angle controller through the calculation of a fused feedback vector

$$\mathbf{e} = \begin{bmatrix} e_{Px} & e_{Py} & e_{Dx} & e_{Dy} & e_{Ix} & e_{Iy} \end{bmatrix}^T, \tag{9.3}$$

which summarises the amount of proportional, derivative and integral feedback there should be in the lateral (x) and sagittal (y) directions at that instant of time. The calculated fused feedback vector $\mathbf{e}$ is

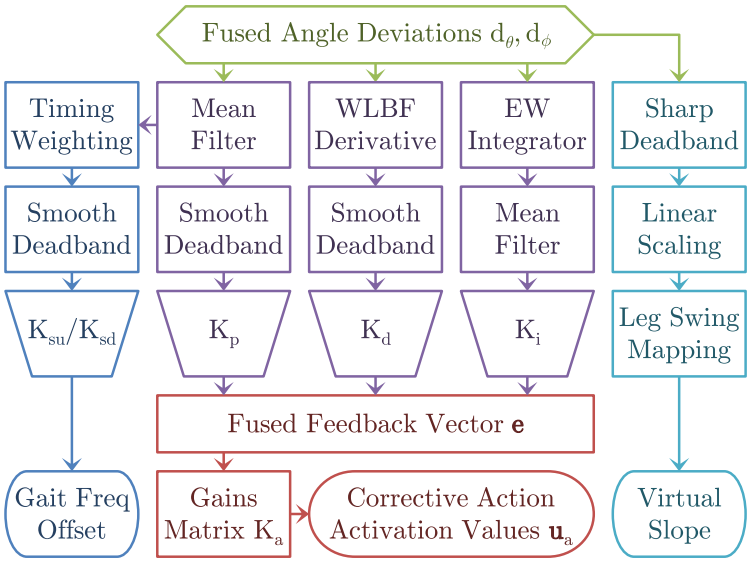


Figure 9.3: Overview of the fused angle feedback calculation pipeline.

converted to a vector $\mathbf{u}_a$ of corrective action activation values using the equation

$$\mathbf{u}_a = K_a \mathbf{e}, \tag{9.4}$$

where K_a is the configured (constant) corrective action gains matrix. Note that a single 5×6 gains matrix K_a is indicated here for generality and mathematical brevity, but in practice a full tune involves only 10–14 non-zero gains in the matrix. These intended feedback paths are explicitly listed throughout the following subsections. The entries of $\mathbf{u}_a$ give the magnitude and sign with which each of the available corrective actions—excluding the virtual slope, which is calculated and added separately (see Section 9.3.5)—are applied to the abstract and inverse poses generated by the CPG gait (Allgeuer, 2020). This is the means by which the robot acts to keep its balance.

To ensure that the pose of the robot stays within suitable joint limits, soft coercion (a form of smooth saturation, see Allgeuer, 2020) is applied to the final calculated abstract arm angles, leg angles, foot angles, and inverse kinematic adjustments to the CoM. Soft coercion is used instead of standard hard coercion (i.e. instead of simply applying hard limiting bounds), because soft coercion is continuously differentiable (of class $\mathcal{C}^1$) and results in smoother saturation behaviour of the limbs, with less self-disturbances of the robot.

9.3.1 Fused Angle Deviation Proportional Feedback

The most fundamental and direct form of fused angle feedback is the proportional corrective action feedback. As with nearly all of the implemented feedback mechanisms, the proportional feedback operates in both the lateral and sagittal planes. The fused angle

deviation values d_θ, d_ϕ are first passed through a mean filter to mitigate the effects of noise. Smooth deadband (Allgeuer, 2020) is then applied to the output to inhibit corrective actions when the robot is close to its intended trajectory, and thereby avoids oscillations due to hunting. Smooth deadband was chosen for this application (as opposed to standard sharp deadband) to soften the transitions between action and inaction, and to avoid any unnecessary self-disturbances and/or unexpected oscillatory activation-deactivation cycles. These can occur if sharp deadband is used, due to its strongly asymmetrical behaviour around its threshold points. The proportional fused angle feedback values e_{Px} and e_{Py} from Equation (9.3) are calculated by scaling the results of the smooth deadband by the proportional fused angle deviation gain K_p.

The proportional fused angle feedback mechanism is intended for activating the *arm angle*, *support foot angle*, and *hip angle* corrective actions, as these have the most direct effect on the transient behaviour of the robot.

9.3.2 Fused Angle Deviation Derivative Feedback

The proportional feedback works well in preventing falls when the robot is disturbed, but if used alone has the tendency to produce potentially unstable low frequency oscillations of the robot, in particular in the sagittal direction due to compliance effects in the ankles. To enhance the transient disturbance rejection performance of the robot, corresponding derivative feedback terms are incorporated to add damping to the system. If the robot has a non-zero angular velocity, this component of the feedback reacts 'earlier' to cancel out the velocity before a large fused angle deviation ensues, and the proportional feedback has to combat the disturbance instead.

The derivative fused angle feedback values e_{Dx} and e_{Dy} from Equation (9.3) are computed by passing the fused angle deviations d_ϕ and d_θ through a Weighted Line of Best Fit (WLBF) derivative filter (Allgeuer, 2020), applying smooth deadband, and then scaling the results by the derivative fused angle deviation gain K_d. The smooth deadband, like for the proportional case, is to ensure that no corrective actions are taken if the robot is within a certain threshold of its normal walking limit cycle, and ensures that the transition from inaction to action and back is smooth.

A Weighted Line of Best Fit (WLBF) filter observes the last N data points of a signal, in addition to assigning confidence weights to each of the data points, and performs weighted linear least squares regression to fit a line to the data, with the data measurement timestamps being used as the independent variable. The linear fit evaluated at the current time gives a smoothed data estimate, and the fitted linear slope gives a smoothed estimate of the derivative of the data stream. WLBF

derivative filters have a number of advantages, in particular when compared to the alternative of computing the numerical derivative of a signal and applying a low pass filter. WLBF filters are more robust to high frequency noise and outliers in the data for the same level of responsiveness to input transients, and have the advantage of inherently and easily supporting non-constant time separations between data points in a stable manner, even if two data points are very close in time. The use of weights in WLBF filters also allows some data points to be given higher confidences than others in a probabilistic manner. For example, in regular parts of the gait cycle where noise and errors in the measurements are expected, lower weights can be used to help reject noise and disturbances in the sensor measurements. To the knowledge of the author, the use of weighted linear least squares regression in this online fashion for the purposes of data smoothing and derivative estimation, in particular in the context of walking gaits, has not previously been published, so no references can be provided.

The derivative fused angle feedback values e_{Dx} and e_{Dy}, are intended for activating the *arm angle*, *support foot angle* and *hip angle* corrective actions, and serve to allow better loop shaping for the transient response to disturbances.

9.3.3 Fused Angle Deviation Integral Feedback

The proportional and derivative fused angle feedback mechanisms are able to produce significant improvements in walking stability, but situations may arise where continued corrective actions are required to stabilise the robot, due to for example asymmetries in the robot. The continual control effort and resulting periodic steady state errors are undesired. The purpose of the integral feedback is to slowly converge to offsets to the gait halt pose (see Section 8.3) that minimise the need for control effort during general walking.

The fused angle deviations d_ϕ and d_θ are first integrated over time using an Exponentially Weighted (EW) integrator, a type of 'leaky integrator'. This kind of integrator incrementally computes the sum

$$I[n] = x[n] + \alpha x[n-1] + \alpha^2 x[n-2] + \cdots \tag{9.5}$$

where $\alpha \in [0, 1]$ is the so-called history time constant, and $x[\cdot]$ is the data to integrate. The sum is most conveniently computed using the difference equation

$$I[n] = x[n] + \alpha I[n-1]. \tag{9.6}$$

If $\alpha = 0$, the integrator simply returns the last data value, but if $\alpha = 1$ the output is the same as that of a classical integrator, so the value of α effectively trims the amount of memory that the integrator has.

A suitable value for α is computed from the desired half-life time T_h, which is a measure of the decay time of the integrator, using

$$\alpha = 0.5^{\frac{\Delta t}{T_h}}, \tag{9.7}$$

where Δt is the nominal time step. An EW integrator is used instead of a standard integrator for flexibility and stability reasons, to combat integral windup, and because old data eventually needs to be 'forgotten' while walking, because situations can change. The alternative of keeping the last N fused angle deviations in a buffer and integrating over the buffer is less efficient, less continuous in the output and less flexible, in addition to being vulnerable to aliasing effects.

As indicated in Figure 9.3, the outputs of the two fused angle deviation EW integrators (one for d_θ and one for d_ϕ) are passed through respective mean filters to allow the trade-off between settling time and level of noise rejection to be trimmed. The smoothed outputs of the mean filters are then scaled by the integral fused angle deviation gain K_i to give the integral fused angle feedback values e_{Ix} and e_{Iy}, as required for the calculation of the fused feedback vector $\mathbf{e}$ in Equation (9.3). The integral fused angle deviation feedback is intended for activating a mix of the *arm angle*, *hip angle*, *continuous foot angle*, and CoM *shifting* corrective actions, although to preserve greater independence between the feedback mechanisms, primarily the last two of these four are recommended.

9.3.4 Fused Angle Deviation Timing Feedback

The Proportional-Integral-Derivative (PID) feedback mechanisms act to return the robot to the expected fused angle limit cycles during walking. For larger disturbances this is not always possible, practicable and/or desired from a timing perspective. For instance, if a robot is pushed so far laterally that it is only resting on the outer edge of its support foot, then no matter what fused angle feedback mechanisms are applied, it is infeasible to return the robot to its centred and upright position in the same time as during normal walking. This generally results in the robot attempting to perform its next step anyway, despite being tilted, and falling as a result. To deal with this mode of failure, timing feedback has been implemented. This form of feedback adjusts the rate of progression of the gait phase as a function of the fused roll deviation d_ϕ. Steps taken by the robot can thereby be sped up or slowed down based on the lateral balance state of the robot.

The required sign of the timing feedback depends on the current gait phase-dependent support leg sign, and only minimal timing adjustments are desired during the double support phase due to the sensor noise associated with foot strike and the transferral of the robot's weight from one foot to the other. As a result, the timing

feedback is constructed by taking the mean filtered fused roll deviation $\bar{d}_\phi$ from the proportional feedback pipeline, and weighting it by a saturated oscillatory gait phase-dependent expression to give

$$\tilde{e}_{Tx} = \bar{d}_\phi \,\text{coerce}\big(-K_{tw}\sin(\mu_i - \tfrac{1}{2}D),\, -1,\, 1\big), \tag{9.8}$$

where $\text{coerce}(\cdot)$ is standard hard coercion, μ_i is the gait phase, D is the double support phase length, and $K_{tw} \in [1, \infty)$ is a gain that adjusts the shape of the oscillatory weighting curve. To avoid unnecessary disturbances to the gait frequency during normal walking, smooth deadband is applied to the calculated weighted deviation $\tilde{e}_{Tx}$ to give the final fused angle timing feedback value e_{Tx}. The required gait frequency offset f_{go} (see Section 8.1.3.1) is then given by

$$f_{go} = \begin{cases} K_{su} e_{Tx} & \text{if } e_{Tx} \geq 0, \\ K_{sd} e_{Tx} & \text{otherwise,} \end{cases} \tag{9.9}$$

where K_{su} and K_{sd} are configured speed-up and slow-down gains. The calculated gait frequency offset f_{go} is then used in the standard CPG gait frequency update equation, given by Equation (8.9).

9.3.5 Virtual Slope Feedback

If the robot is tilted sagittally in the direction that it is walking, it can happen that a foot prematurely strikes the ground during its swing phase. This is undesirable and a cause for destabilisation of the robot. The virtual slope corrective action adjusts the inverse kinematic height of the feet in a gait phase-dependent manner as a function of the estimated fused pitch θ_B and the sagittal component v_{ix} of the internal Gait Command Velocity (GCV). This is used to effectively simulate as if the robot was walking up or down a slope, hence the name of the corrective action. For example, if the robot is walking forwards and tilted forwards, the robot with virtual slope active will lift its feet higher at the front of its swing phases, ensuring that the legs do indeed reach full forward swing before establishing ground contact.

The required sagittal virtual slope V_s is calculated at each point in time by first applying sharp deadband to the measured fused pitch θ_B of the trunk, then linearly scaling the result by one of two gains, depending on whether the robot is tilted in the same direction that it is walking or not. The sagittal virtual slope feedback value e_{Vy} is then calculated using the formula

$$e_{Vy} = V_s v_{ix}, \tag{9.10}$$

where v_{ix} is the sagittal component of the current internal GCV vector $\mathbf{v}_i$. The calculated value of e_{Vy} is taken as the maximum required magnitude of inverse kinematic height adjustment of the feet during

the current step. If ζ_s is the leg-dependent *swing angle* used in the generation of the CPG waveforms (Allgeuer, 2020), the exact required inverse kinematic height adjustment at every instant in time is given independently for both legs by the formula

$$p_{lz} \leftarrow p_{lz} + \tfrac{1}{2}\big(\zeta_s + \operatorname{sign}(e_{Vy})\big), \tag{9.11}$$

where p_{lz} is the z-component of the inverse space ankle position vector $\mathbf{p}_l$, and $\operatorname{sign}(\cdot)$ is a sign function such that $\operatorname{sign}(0) = 1$.

9.3.6 Tuning of the Feedback Mechanisms

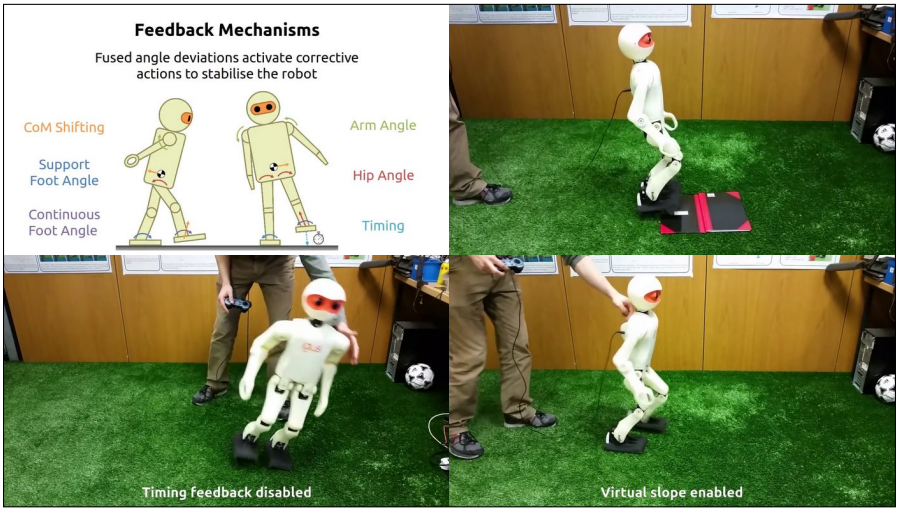
The process of tuning the feedback mechanisms is greatly simplified by their considerable independence. The individual feedback mechanisms have clearly observable and direct effects that can be precisely isolated and tested, so arguably the process of tuning the proposed gait is quicker and easier than it would be for most model-based approaches that do not somehow work out of the box. For instance, in experiments it was always observed that the direct fused angle controller is *significantly* easier to tune for a robot than the capture step controller.

The process of tuning starts with the Proportional-Derivative (PD) gains, as these are most responsible for the transient disturbance response characteristics of the robot. The most effective corrective actions in each of the sagittal and lateral planes of motion are identified for each robot, and the gain ranges that produce noticeable effects without risking oscillations or instabilities are also experimentally established. The choice of corrective actions may be influenced by external objectives, such as for example the desire to minimise trunk angle deviations from upright. In this case, the hip angle feedback components may be omitted for example.

Once the transient response has been tuned, a suitable integral feedback half-life time is chosen based on the rate at which the robot should adapt to changes in its environment, and gains are chosen that bring the activations of the associated integral corrective actions to the desired range. Timing is then considered, with the speed-up and slow-down gains being selected to avoid premature stepping, and to instantaneously halt the gait phase when the robot reaches a certain nominal lateral angular deviation. The virtual slope mechanism is mostly parameter-free, and the sole pair of linear scaling factors are chosen to provide the desired margin of clearance of the foot from the ground during maximum forwards walking.

9.4 EXPERIMENTAL RESULTS

The proposed gait has been implemented and evaluated experimentally on all of the robots used throughout this thesis, including in



Video 9.1: Walking experiments demonstrating the feedback mechanisms implemented as part of the direct fused angle feedback controller. These are the exact experiments (for the most part) for which the plots are provided in Figure 9.4.
https://youtu.be/xnzJi2hTfAo
Omnidirectional Bipedal Walking with Direct Fused Angle Feedback Mechanisms

particular all seven igus Humanoid Open Platform robots that were constructed over the years, as well as the Dynaped robot. As a matter of cross-validation, the direct fused angle controller was observed to work particularly well on Dynaped, despite the fact that essentially the same controller parameters and gains matrix K_a were used as on the igus Humanoid Open Platform. This demonstrates the independence and portability of the feedback controller, which is a great strength of the gait that derives from its essentially model-free nature.

Overall, the feedback mechanisms were observed to make a significant difference to the walking ability of the tested robots—as demonstrated in detail in Video 9.1 for each feedback mechanism individually—with walking often not even being possible for extended periods of time without them. The feedback mechanisms also equipped the robots with disturbance rejection capabilities that were not present otherwise. Reliable omnidirectional walking speeds of 21 cm/s were achieved for the igus Humanoid Open Platform, and 36 cm/s was achieved for Dynaped. Both of these walking speeds were measured on an artificial grass surface of blade length 32 mm.

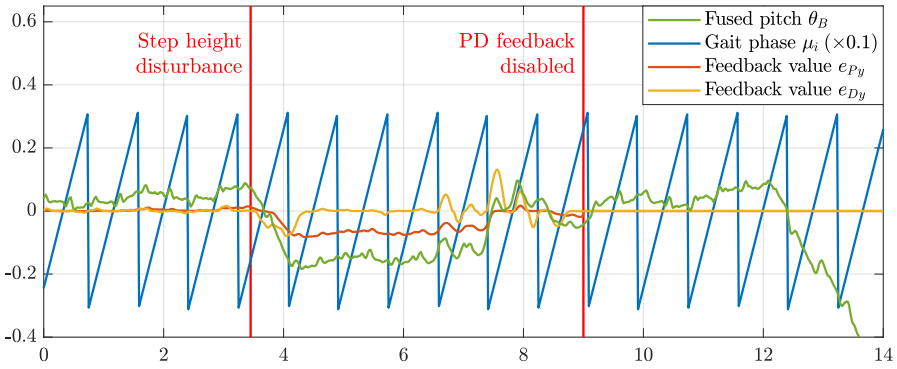
Plots of experimental results demonstrating the efficacy of the feedback mechanisms are shown in Figure 9.4. In Figure 9.4a, the robot walked twice onto an unexpected 1.5 cm step change in floor height, the first time with proportional and derivative feedback enabled, and the second time with them disabled. It can be seen that with the feedback enabled the robot is able to avoid falling—albeit with a large steady state error in the fused pitch—while without feedback

the robot falls immediately. Adding in integral feedback, configured to activate both the CoM shift and continuous foot angle corrective actions, produces the results in Figure 9.4b. It can be seen that the relatively large initial error in the fused pitch is rejected within about 5 s, with the robot converging to upright walking despite being on an uneven floor. Note that the residual fused pitch limit cycles towards the end of the plot are an effect of the robot's feet not being equally positioned on the step change in floor height.

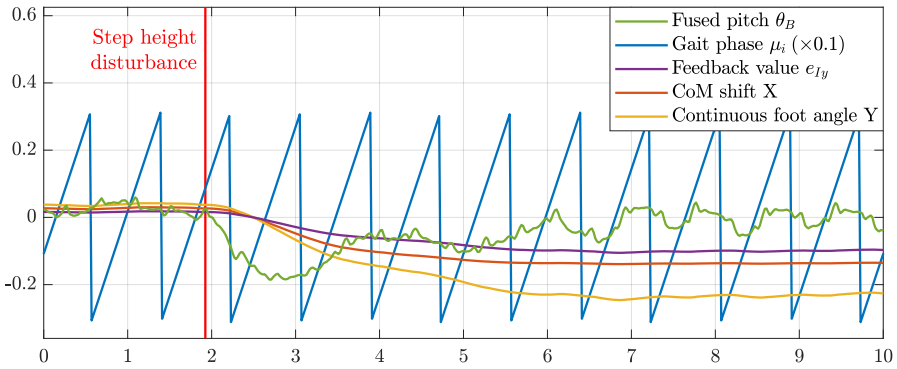
Figure 9.4c shows the effect of timing feedback. Timing feedback activates when there are significant deviations in the lateral direction between the measured and expected orientations of the robot. A plot comparing the measured and expected fused pitch and roll values for stable balanced walking is shown in Figure 9.5. Despite initial appearances, the deviations from expected are actually quite small ($<1.1°$), and only larger deviations should trigger changes to the nominal step timing. In Figure 9.4c, the robot was subjected to two lateral pushes while walking in place, the first with timing feedback enabled, and the second without. It can be seen at $t = 3.1$ s that the gait phase slows down in response to the push, allowing the fused roll to return to its expected limit cycle within the following 2.5 s. An identical push at $t = 8.2$ s, without timing feedback enabled, causes the robot to fall.

Figure 9.4d illustrates the effect of the virtual slope corrective action. The robot was continuously pushed forwards while it was walking forwards, first with the virtual slope feedback enabled, and then without. In the first case, the adjustments to the inverse kinematic height of the feet ensured that the robot could continue to walk forwards and regain balance once it was no longer being pushed. In the latter case however, the feet started to collide with the ground in front of the robot, preventing the robot from taking its intended step sizes, and causing significant self-destabilisation. The robot was not able to get its feet underneath its CoM again, and subsequently fell shortly after.

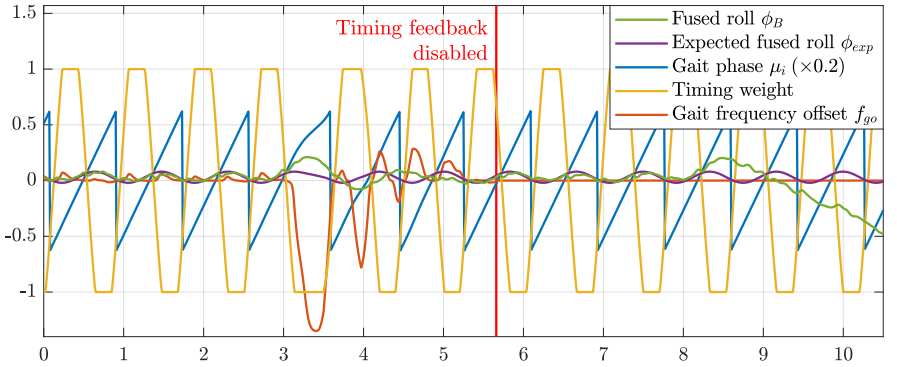
A more detailed idea of how the PD corrective actions activate in the face of push disturbances is given in Figure 9.6. The top plot shows the response of an igus Humanoid Open Platform robot that was pushed forwards while walking on the spot on artificial grass, while the bottom plot shows an equivalent backwards push. In both cases, the fused angle deviation proportional and derivative feedback components were enabled, and used to activate the sagittal arm angle and support foot angle corrective actions. Prior to both pushes, it can be observed from the plots that no significant activations of the feedback mechanisms were occurring. As soon as the destabilisation caused by the pushes was detected however, i.e. via the ensuing deviation caused in the measured fused pitch, the arms and feet sprung into action to counteract the disturbance and return the robot



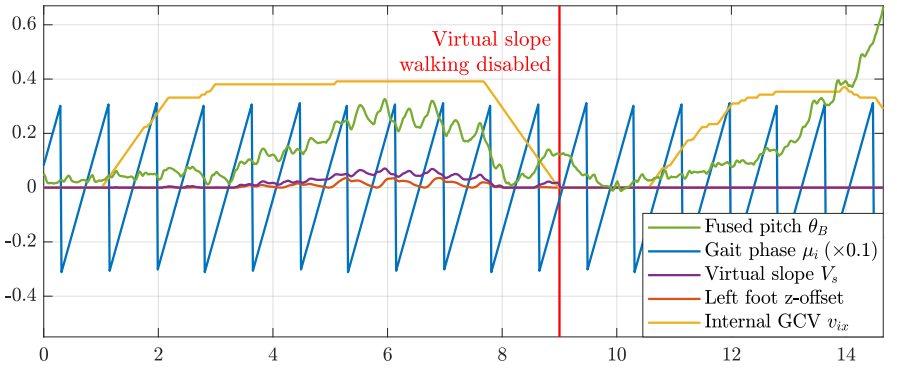
(a) Effect of PD feedback when walking onto a step change in floor height.



(b) Effect of I feedback when walking onto a step change in floor height.



(c) Effect of timing feedback on the lateral transient response to pushes.



(d) Effect of virtual slope feedback while walking and being pushed forwards.

Figure 9.4: Plots of experimental results showing the effects on the stability of the robot of the five implemented feedback mechanisms.

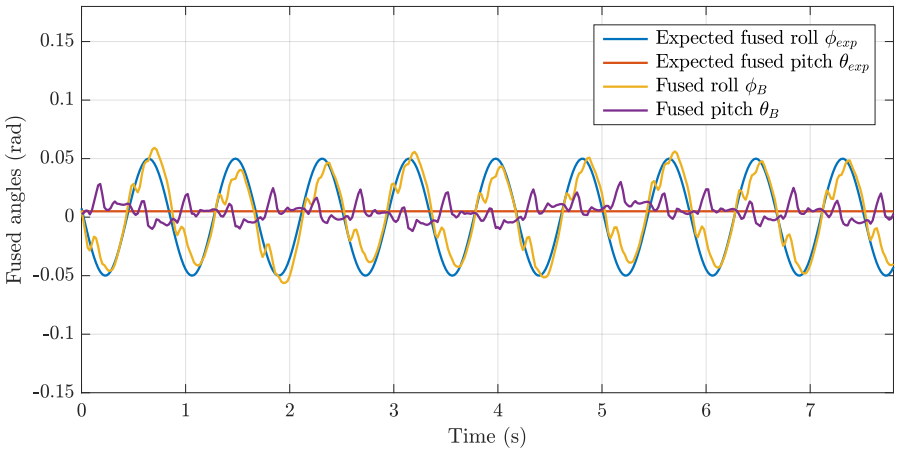


Figure 9.5: Plot of the fused angle waveforms θ_B, ϕ_B, along with the expected fused angle waveforms θ_{exp}, ϕ_{exp}, during on-spot walking. The close correlation between the expected and true orientations of the robot can be observed. Note that the plot is quite enlarged, so the deviations from expected in this case are actually all less than $\approx$0.02 rad $= 1.1°$.

to a stable walking limit cycle. Note that the support foot angle activations alternate between the left and right feet as the robot takes steps, as it only makes sense to adjust the orientation of the foot that is actually on the ground. Although the activations reacted very quickly after the push, and spiked to a 'large' value, the derivative feedback component ensures that as the robot dissipates energy and starts to return towards upright, the magnitude of the activations quickly drops again, and in fact even briefly reverses so as to prevent excessive overshoot. Sagittal overshoot is a common problem of the open-loop CPG. It causes the pitch of the robot to oscillate back and forth in an only lightly damped manner, and makes the robot more sensitive to any further disturbances. This is actively prevented by the feedback controller, leading to greater walking stability.

A more holistic qualitative demonstration of the proposed gait (i.e. the CPG gait combined with the direct fused angles feedback controller) and its resistance to disturbances can be found in Video 9.2. Experiments and real-life situations are shown for multiple different robot platforms, including Dynaped, the igus Humanoid Open Platform, NimbRo-OP2 and NimbRo-OP2X, and impressions of walking performance in simulation are also given. Notable aspects of the video include the wide variety of conditions and disturbances that the gait is confronted with, as well as the fast maximum walking speeds that were achieved on the NimbRo-OP2X ($\approx$59 cm/s).

Despite (by design) not using adaptations of step size for the preservation of balance, large improvements to the push recovery ability of the robot are achieved by the direct fused angle feedback controller. This level of improvement has been quantified using controlled push

Video 9.2: Demonstration of the closed-loop direct fused angle feedback gait. Simple corrective actions based on the estimated fused angles of the torso are used to maintain the balance of the robot, even in the face of disturbances. The gait is demonstrated on the igus Humanoid Open Platform, Dynaped, NimbRo-OP2 and NimbRo-OP2X robots, as well as in simulation.
https://youtu.be/DvxZJVVRdyE
Walking with Corrective Actions Driven by Direct Fused Angle Feedback

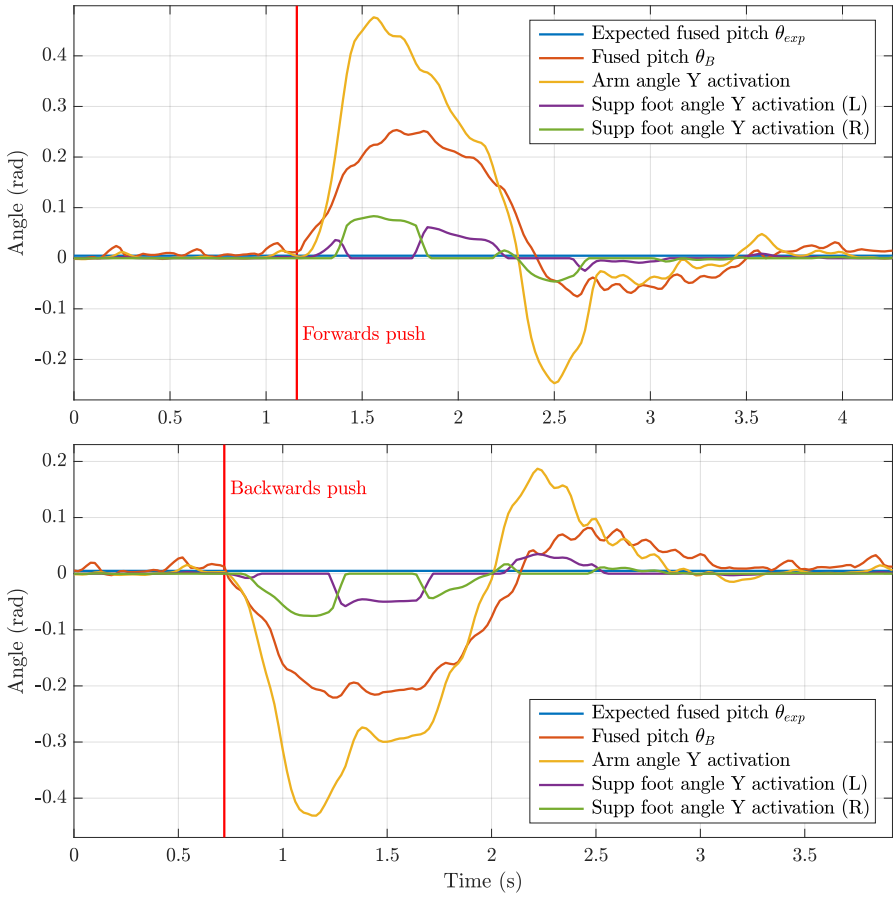


Figure 9.6: Plots of the response of the direct fused angle feedback controller to sagittal pushes. The top plot shows a forwards push during which the robot reaches a maximum forwards tilt of 0.253 rad, while the bottom plot shows a backwards push that results in a maximum backwards tilt of 0.22 rad. Both disturbances were rejected primarily via the arm angle and support foot angle corrective actions, the activations of which are shown in the plot.

experiments in Gazebo simulation. A simulated igus Humanoid Open Platform was made to walk on the spot with the feedback controller either turned on or off, and 20 pushes of equal impulse magnitude but random direction[2] were applied at regular time intervals. If a robot fell over due to a push, it was manually reset to upright in time for the next push. The number of successful pushes, i.e. pushes where the robot did not fall over, even with delayed consequence, were recorded for the open-loop and closed-loop gait for push impulses in the range of 0.3–2.0 s N. The numerical results of the experiment are listed in Table 9.1, and the corresponding videos of the actual experiments are provided in Video 9.3. Figure 9.7 provides a graphical summary of the obtained results, and what it means for the comparative stability of the open-loop and closed-loop gaits. Overall, it can be observed that

2 The pushes were restricted to be 'horizontal' in the sense that they are parallel to the ground plane, but random in terms of their exact heading.

Table 9.1: Number of withstood simulated pushes (out of 20) when walking on the spot with the direct fused angle feedback controller

Impulse (s N)	**0.3**	**0.6**	**0.9**	**1.2**	**1.6**	**2.0**
Open-loop	20	13	11	6	5	3
Closed-loop	20	20	20	19	17	13

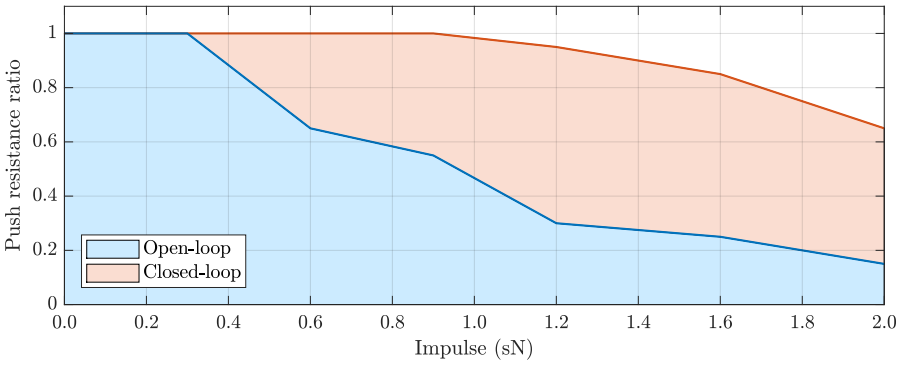


Figure 9.7: Plot of the ratio of withstood pushes against push impulse magnitude for a simulated igus Humanoid Open Platform walking on the spot with the Central Pattern Generator and direct fused angle feedback controller. The performance of the robot with and without the controller activated is contrasted. The raw data corresponding to the plot is given in Table 9.1.

the feedback controller makes a clear difference to the stability of the robot. The effect is most noticeable for higher push impulses, where for example a push impulse of 1.6 s N resulted in only 5 successful pushes for the open-loop gait, but 17 for the closed-loop gait. Note that for reference of scale, a push impulse of 2.0 s N is expected to cause an instantaneous change in CoM velocity of about 30–40 cm/s for the igus Humanoid Open Platform, which is considerable given its CoM height of about 55 cm.

Push experiments were also performed on real hardware using the igus Humanoid Open Platform. As push impulses cannot easily be set or measured in real-life push experiments, a different approach was used, whereby the robot was pushed many times over many minutes of walking, and the corresponding push responses were extracted, synchronised, overlaid, and finally classified as either successful or unsuccessful. The results of this process can be seen in Figure 9.8. Separate plots are provided for forwards and backwards pushes, and for the open-loop and closed-loop gaits (i.e. with and without the direct fused angle feedback controller). Note that the lines in the plot terminate whenever the robot started receiving assistance due to falling (red lines), or when a subsequent push was started (blue lines). It can clearly be observed from the plots that the feedback controller has a strong effect in reducing the duration that a disturbance is experi-

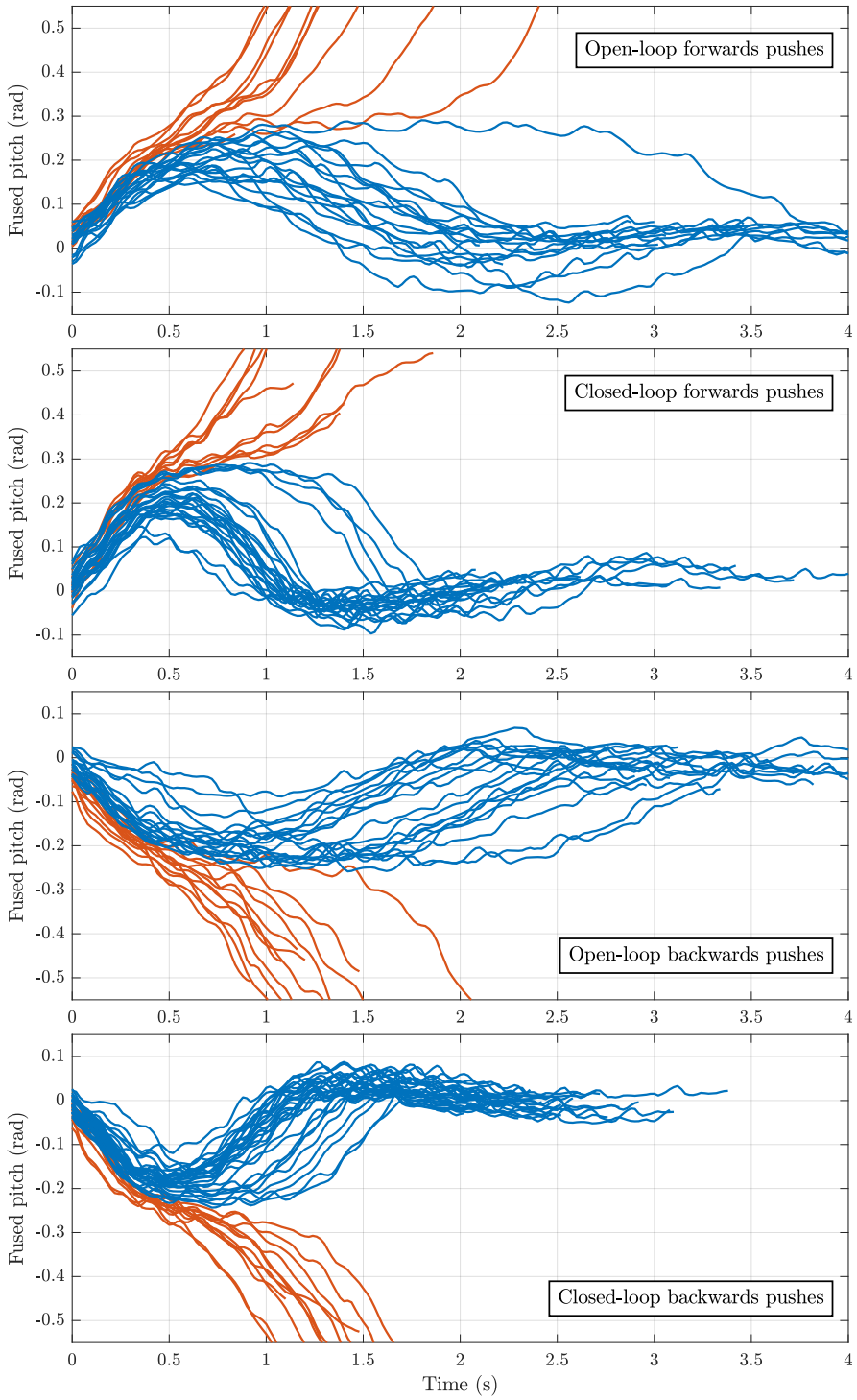
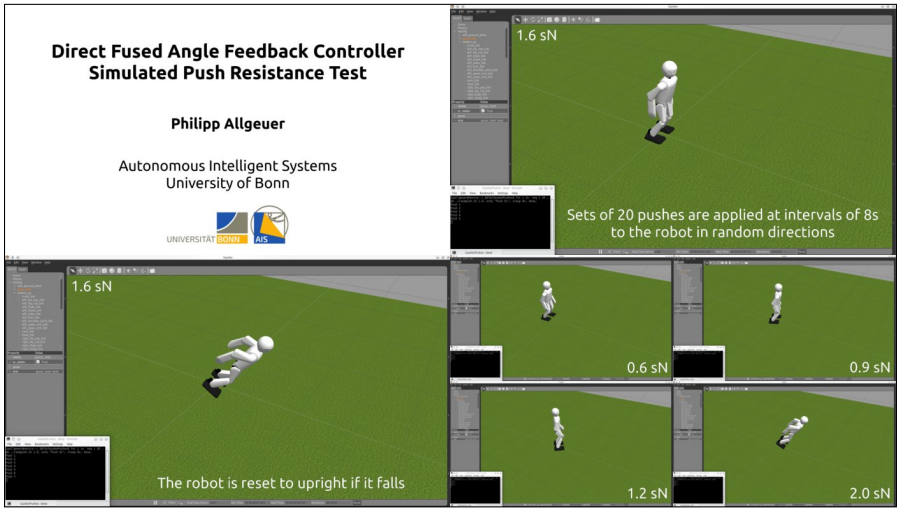


Figure 9.8: Plots of the transient response of a walking igus Humanoid Open Platform robot to sagittal pushes of various strengths. The pushes are grouped by whether they are forwards or backwards, and whether the feedback controller was active at the time or not. From top to bottom, the plots correspond to a) open-loop forwards pushes, b) closed-loop forwards pushes, c) open-loop backwards pushes, and d) closed-loop backwards pushes. The pushes all occurred at time $t = 0$, and each individual curve is coloured blue if the robot survived the push, otherwise red.



Video 9.3: Push resistance test performed on an igus Humanoid Open Platform in simulation. The robot is walking on the spot with the CPG gait, and is disturbed in random directions by impulses of various magnitudes. The effectiveness of the direct fused angle controller is evaluated by comparing it to open-loop performance.
https://youtu.be/c6zlCK4nFG0
Direct Fused Angle Feedback Controller Simulated Push Resistance Test

enced, and the amount that the fused pitch oscillates when it returns to nominal. The controller is able to efficiently leverage the effects of the corrective actions to more quickly dissipate the disturbance energy without causing any extra oscillations or disruptions. Note that the increase in maximum fused pitch that can successfully be recovered from is only modest when comparing open-loop and closed-loop performance. This is because the tipping point of the robot is somewhat mechanically fixed, and can only be slightly altered by moving the arms. The magnitude of the push impulse that it takes to *get* the robot to this maximum fused pitch is significantly higher however (for closed-loop), as it first has to overcome the corrective actions to even reach the tipping point.

The responses of the robot to the pushes shown in Figure 9.8 can also be analysed in the phase space, i.e. in the space of fused pitch velocity vs. fused pitch. Combining the respective forwards and backwards pushes into a single plot yields the curves shown in Figure 9.9. The top plot shows the traces of the push responses for the open-loop gait, while the bottom plot shows the same for the closed-loop gait. Note that the start of each trace ($t = 0$ in Figure 9.8) is marked with a solid dot. It can immediately be observed from Figure 9.9 that the region in the phase space corresponding to stable walking trajectories is larger for the closed-loop gait than for the open-loop gait. This increase in size corresponds predominantly to a wider range of stable fused pitch velocities—an observation that is consistent with the previous observations that the closed-loop gait is more quickly able to dissipate

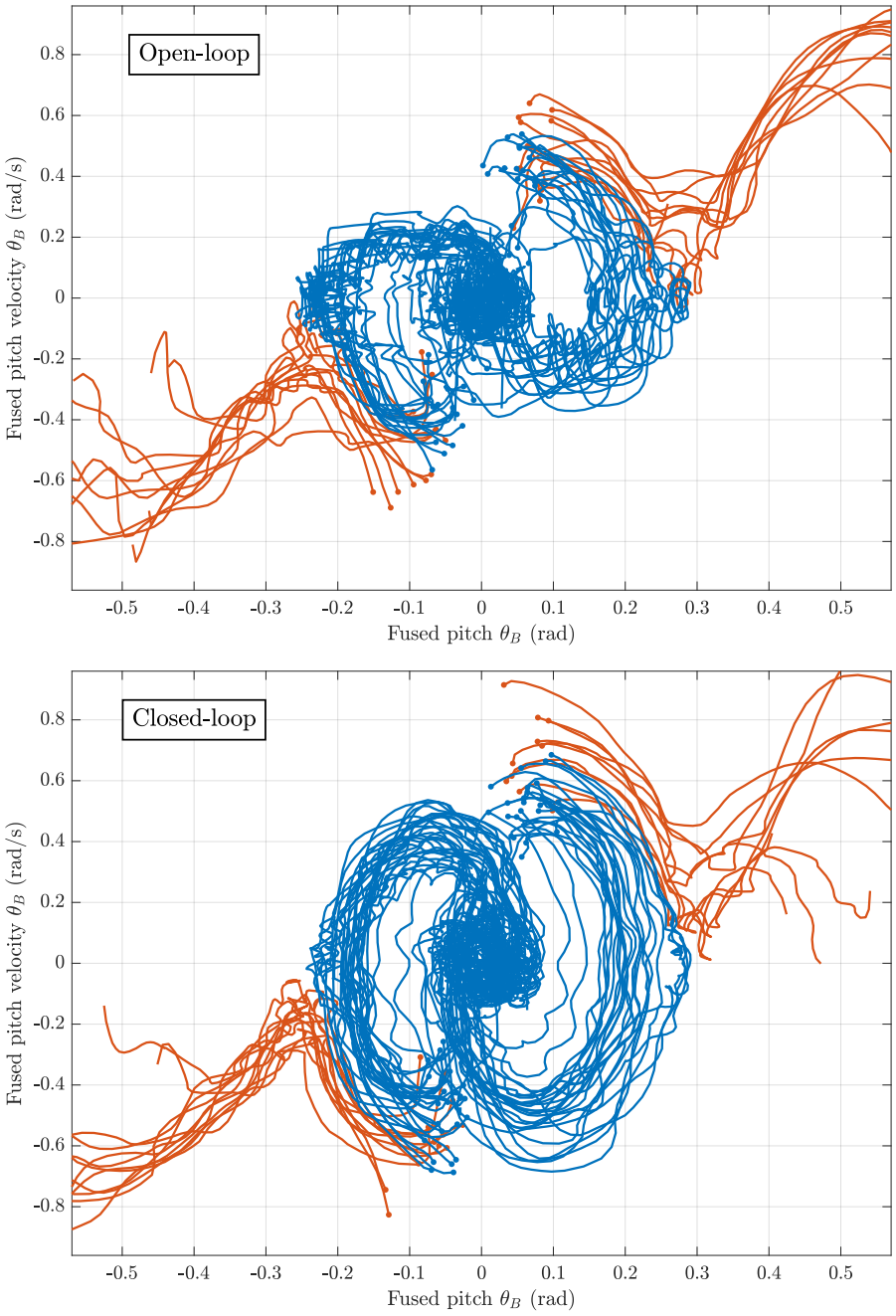


Figure 9.9: Plots of the phase response of a walking igus Humanoid Open Platform robot when sagittal pushes of various strengths are applied. The fused pitch velocity $\dot{\theta}_B$ is plotted against the fused pitch θ_B for each individual push trajectory, and the resulting curve is coloured according to whether the robot successfully withstood the push (blue) or not (red). The beginning of each trajectory is marked with a solid dot. The top plot corresponds to the performance of the robot with the feedback controller disabled, and the bottom plot shows how this performance improves when the feedback controller is enabled.

disturbance energies, and can deal with larger initial CoM velocities after a push. These factors together lead to the greater overall push recovery ability that is observed for the closed-loop gait. In reference to Figure 9.9, it should be noted that some of the red traces start from deep within the blue region of stability, and first sharply increase their velocity (a vertical change in the plot) before subsequently starting to diverge in terms of the fused pitch. This is deduced to correspond to situations where the push from the experimenter was still ongoing when the trace was started. As a similar observation, some of the red unstable trajectories start to decrease their velocity towards the end of the trace. This is deduced to correspond to situations where the experimenter caught the falling robot before the trace was stopped.

The plots in Figure 9.9 can alternatively be viewed as a heat map, as shown in Figure 9.10. The states in the phase space are divided into cells (i.e. bins), and the trajectories that pass through any one cell are collected and used to calculate a success rate for the cell. The success rate is a value in $[0, 1]$, and corresponds to the proportion of the trajectories that passed through the cell that did not end in a fall. Extra data of normal balanced walking has been incorporated into the figure to avoid the issue of unvisited states in the middle of the stable region (i.e. holes in the data as can be seen in Figure 9.9). Note that a minimum cell visit count of two was used, explaining why disconnected cell patches are possible. It can be observed from the figure that the robot is 100% stable close to the phase origin, but as the fused pitch and/or velocity increase, there is a sharp transition to 0% stability. This transition is especially sharp in the case of the closed-loop gait, which exhibits a relatively low amount of intermediate transitional cells between the blue and red areas. Overall, like in Figure 9.9, it can be seen that the closed-loop gait has a significantly larger region of stability, and that the main gain in area is due to the greater stability of states with high fused pitch velocities.

The task of tuning the PD gains of the direct fused angle feedback controller can usually easily be completed manually, but an alternative learning approach for the tuning of the sagittal gains has been proposed by Rodriguez et al. (2018). They used Bayesian optimisation with Gaussian process regression and multi-fidelity entropy search to share experiments and parameter updates between real-robot experiments and physical simulations, all the while trying to account and correct for systematic errors between the two. A video is available[3] that compares the manually tuned sagittal feedback gains to the final optimised ones for an igus Humanoid Open Platform. Although the difference may be difficult to detect at first, on closer inspection the manually tuned robot can be seen to exhibit slightly greater low frequency oscillations in the pitch direction, which at times can impede clean walking progress. More experiments would be needed to evalu-

3 https://youtu.be/twl8cGefsz8

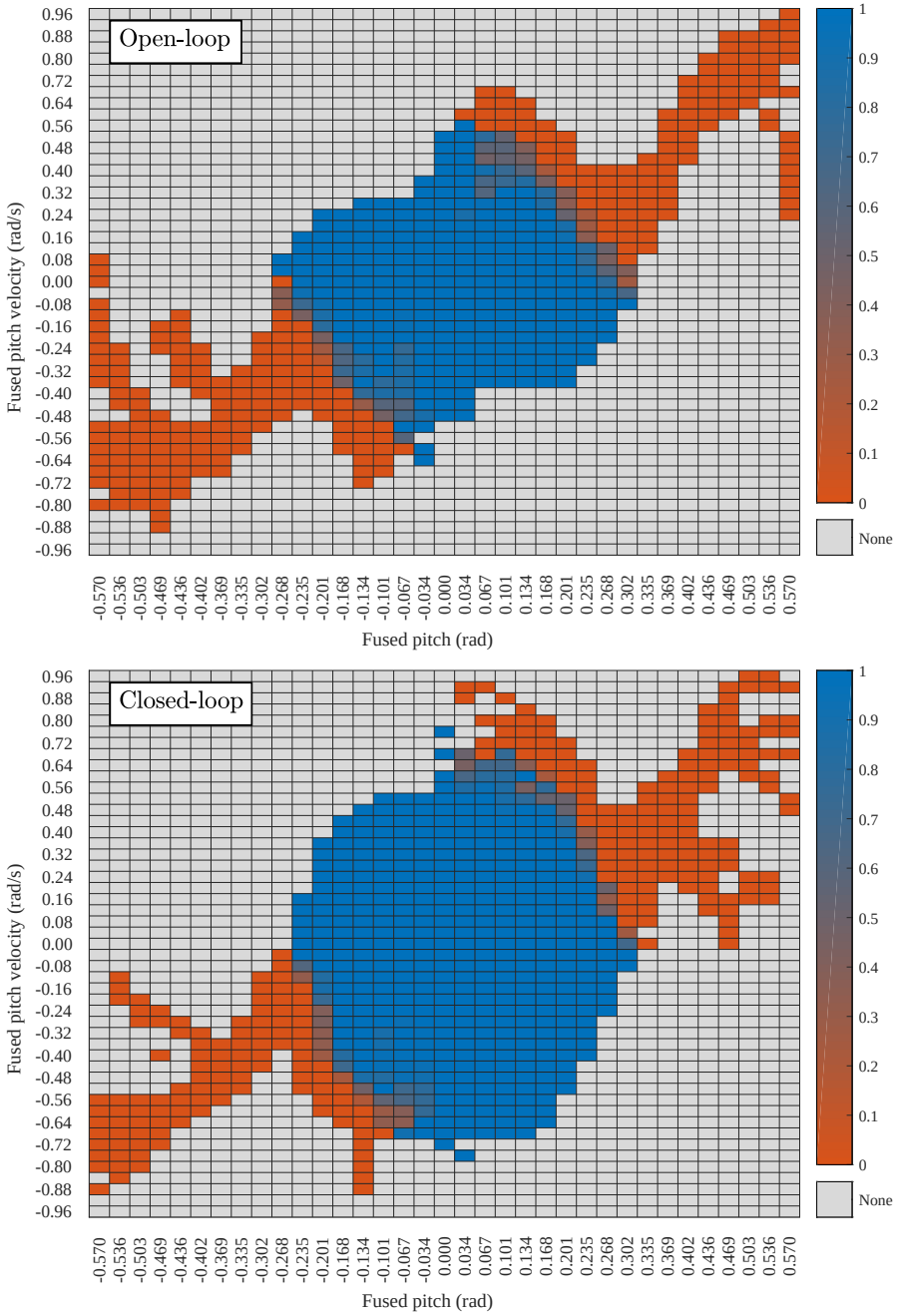


Figure 9.10: Plots of the open-loop vs. closed-loop walking stability of an igus Humanoid Open Platform in the form of a phase space heat map. The blue areas indicate phase states corresponding to stable walking (1.0 = 100% success rate), while the red areas indicate states where the robot tendentially lost balance and fell over (0.0 = 0% success rate). In-between colours indicate that certain binned states were encountered multiple times with different outcomes. All the data from the push tests is shown, including also the states of normal balanced walking between pushes. The grey cells correspond to phase states that were not encountered during the tests.

ate the true potential of this method in surpassing manual tuning in a time and resource-efficient manner.

The gait presented in this chapter has successfully been run in combination with the capture step controller, and produced results notably superior to just using the latter on the igus Humanoid Open Platform. The capture step timing was used in place of the fused angle deviation timing, and the step sizes computed from the capture step algorithm were either used or not used based on the situation (e.g. in testing vs. at competition). The gait was used in this configuration for example at the RoboCup 2016 soccer tournament, and over all games played, none of the five robots ever fell while walking (see Video 1.1 for impressions of the competition), except in cases of strong collisions with other robots. In fact, up to and including 2018 all of the TeenSize and AdultSize robots of the NimbRo RoboCup team always used the direct fused angle feedback controller for gait stabilisation purposes, frequently however with capture step timing enabled. Video 1.2, for example, shows the AdultSize RoboCup performance with capture step timing enabled, while another video[4] shows the performance of the same robots with the simpler fused angle deviation timing. The extensibility of the gait to incorporate other methods of step size and timing validates, amongst other things, the use of the presented gait as a stabilising foundation for more complex step size adaptation schemes.

9.5 CONCLUSION

An inherently robust omnidirectional closed-loop gait has been presented in this chapter that stabilises a central pattern generated open-loop gait using fused angle feedback mechanisms. The gait is simple, model-free, quick to tune, easily transferable between robots, and only requires servo position feedback if feed-forward torque compensation is desired as part of the actuator control scheme (see Chapter 3). The gait is also suitable for larger robots with low-cost sensors and position-controlled actuators. This demonstrates that walking does not always mandate complex stabilisation mechanisms. The gait has been experimentally verified and discussed, and demonstrably made robots walk that were not able to produce even remotely similar results with just a manually tuned open-loop CPG approach. One of the notable merits of the presented gait is that it can combine very well with more complicated model-based approaches that are able to suggest step size and/or alternative timing adjustments. This is what makes the gait so useful and powerful as a building block for more complex and more tailored gait stabilisation schemes.

4 https://youtu.be/RG205OwGdSg

10

KEYPOINT GAIT GENERATOR

The task of bipedal locomotion exposes many facets of the concept of balance—most notably the many and varied methods by which balance can be preserved. While humans, even in early childhood, seem to effortlessly know how to stabilise their gait and best react to pushes while walking, the situation is quite different for humanoid robotic platforms. For robots it must first be delineated by which approaches they can be made to keep their balance, before algorithms can be developed that allow them to reliably execute such strategies.

As discussed in Chapter 2, broadly speaking there are two main paradigms for the implementation of bipedal robotic gaits. A common approach in the state of the art is to use a dynamics model of some kind to capture and predict the physical response of the robot, and calculate or optimise a trajectory to satisfy the required motion and balance criteria. This motion trajectory is then executed on the robot, often with a controller to reject deviations and enforce tracking, and/or under regular recomputation to adapt for differences in the real response of the robot. For imprecise low-cost robots however, where good quality tracking and execution of a trajectory is not given, using such optimised trajectories generated directly from simplified (or even whole-body dynamics) models is often fraught with difficulty. Significant nonlinearities, such as joint backlash, sensor noise, sensor and actuator delays, irregular properties of the contact surface, and unmeasurable external disturbances, are difficult to incorporate into models. This greatly limits the predictive power of such models, and subsequently the applicability of such methods to such robots.

Nevertheless, simpler, cheaper and smaller robots are often easily made to walk despite these nonlinearities using hand-crafted gaits. These gaits are however usually not very flexible or stable on their own. This is the foundation of the second paradigm for the implementation of bipedal gaits—letting the robot find its own natural rhythm and stability with an inspired open-loop gait generator that is entirely self-contained, and extending it with a higher level controller that seeks to preserve and return to that rhythm when there are significant deviations from it. As we did for the Central Pattern Generator (CPG), in this chapter we continue with this second paradigm and present a self-stable omnidirectional gait generator that seeks to strike a balance between the security and simplicity of hand-crafted gaits, and the advantages of analytically computed and optimised gaits. More than just making the robot walk, the so-called Keypoint Gait Generator (KGG) directly embeds a myriad of corrective actions into its generation

algorithm, each of which can be commanded and activated by higher level controllers to systematically allow preservation of balance during walking. The diversity of the implemented corrective actions arguably cover the majority of humanlike strategies by which biped robots can balance during walking.

The idea for the keypoint gait generator originated from the CPG, in particular in the light of the CPG's observed shortcomings. The KGG is somewhat like an extension and complete redesign of the CPG from the ground up, with a focus on greater flexibility and general analytical correctness in the way the gait motion waveforms are generated, in particular from a task space perspective. While the CPG strictly only generates the nominal open-loop walking waveforms, and relies on higher level balance controllers like the direct fused angle controller to later change the waveforms to implement corrective actions, the KGG intrinsically incorporates the effects of a diverse array of corrective actions and directly generates the final required joint waveforms. Amongst many improvements of the KGG over the CPG (see Section 10.1.3), the KGG incorporates many new and/or improved corrective actions in its generation algorithm (as compared to the CPG), and allows situations like for example lateral crossing trajectories to be explicitly addressed and dealt with.

An implementation of the Keypoint Gait Generator (KGG) has been released open source as part of the `feed_gait` package of the Humanoid Open Platform ROS Software.[1]

10.1 MOTIVATION

We first put the KGG into context by identifying the objectives that it intends to fulfil, the balance strategies it intends to implement, and where it fits into the bigger picture of the gait architecture.

10.1.1 Strategies for Balanced Walking

There are many ways in which a humanoid robot is assessed to be able to influence its balance while walking, not just the two that are most commonly addressed in other works—adjustments of step size and timing. Inspired in part by the reactions humans exhibit when disturbed while standing or walking, a list of strategies for preserving or recovering balance in a bipedal robot is as follows:

- Adaptation of step sizes,
- Adaptation of step timing,

1 https://github.com/AIS-Bonn/humanoid_op_ros/tree/master/src/nimbro/motion/gait_engines/feed_gait

- Horizontal planar shifts of the torso, and thereby Centre of Mass (CoM) position, relative to the feet,
- Active changes in torso height, and thereby CoM height, above the ground,
- Active leaning of the robot torso from the hips,
- Arm motions to generate reaction moments and/or statically offset the centre of gravity of the robot,
- Swing leg trajectory adjustments, to generate reaction moments and/or bias the weight of the robot in a particular direction by means of modifying how the swing leg moves between lift-off and foot strike,
- Height adjustment of the predicted locations of lift-off and foot strike to adapt to the ground, for example as required in the case of a tilted robot,
- Intentional utilisation of ground normal forces to apply a restoring moment to the robot to make it more upright, for example by pushing more into the ground at foot strike, or less just before lift-off,
- Adjustments of the foot tilt relative to the ground at the instant of foot strike, resulting in large impulses to the balance state of the robot due to the impactful nature of the ground contacts, and,
- Adjustments of the foot tilt relative to the ground during the support and swing phases.

The Keypoint Gait Generator (KGG) presented in this chapter seeks to address all of these strategies by extending its fundamental walking waveforms with a multitude of corrective actions that can be activated simultaneously (as required) to preserve balance.

10.1.2 Gait Architecture

The gait presented in this chapter is referred to as a gait generator, as it gives the ability to higher level controllers to execute all of the balance strategies implemented therein, without these controllers needing to know anything about joint motions, how they look like, or how they need to be generated.[2] As such, the gait architecture can logically be partitioned into the higher level controller(s) in use, and the KGG that sits underneath them, with the latter receiving its commands from the

2 Recall that this was not the case for the CPG and its corresponding higher level controllers, as for example the direct fused angle controller directly modifies the generated joint waveforms in order to implement its required corrective actions.

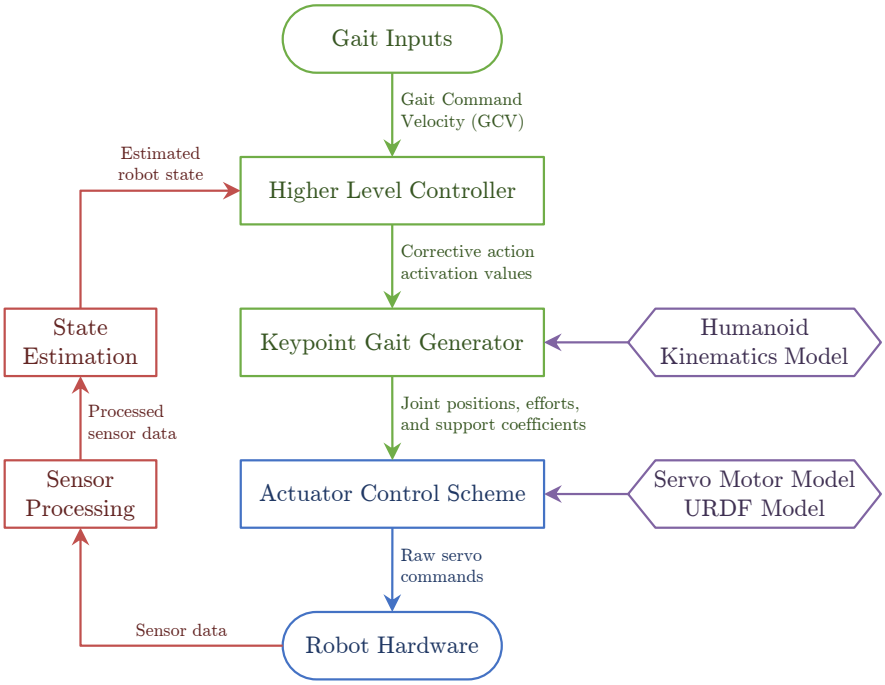


Figure 10.1: Overview of where the KGG fits into the overall keypoint gait architecture. Note that it is possible for there to be multiple higher level controllers running in parallel if each of them calculate activation values for different sets of corrective actions. The actuator control scheme (Chapter 3), sensor processing (Chapter 4), and state estimation (Chapter 7) have been covered in previous chapters. A higher level controller that drives all of the corrective actions implemented in the KGG is presented in Chapter 11.

controllers via the gait interface described later in Section 10.2.2.1. The complete gait architecture, including the sensorimotor management aspects, is shown in Figure 10.1. Despite the appearance in the figure, it should be noted that no higher level controllers are actually mandatory for walking at all—the gait generator can make the robot walk open-loop by itself as it was designed to do—but walking open-loop does not make use of any of the implemented balancing corrective actions, and is subsequently not as stable as feedback-based walking.[3]

The input to the gait as a whole is the dimensionless Gait Command Velocity (GCV) vector

$$\mathbf{v}_g = (v_{gx}, v_{gy}, v_{gz}) \in [-1, 1]^3. \tag{10.1}$$

This is a vector of three dimensionless values in the range $[-1, 1]$, representing the required x, y and yaw velocities of the robot as a ratio

3 Note that in the case of open-loop walking without a higher level controller, the input GCV $\mathbf{v}_g$ is passed directly to the keypoint gait generator as the internal GCV $\mathbf{v}_i$, and the gait frequency f_g that is normally set by the higher level controller (see Section 10.2.2.1) is simply kept constant at its nominal value.

of their respective allowed maximums. The higher level controllers process the input GCV vector $\mathbf{v}_g$, along with the sensory feedback from the robot, and compute commands for the KGG that try to ensure that the robot remains balanced. These commands consist of the required activation values for the KGG corrective actions—including for example the required instantaneous gait frequency f_g—and incorporate in particular also the internal GCV vector

$$\mathbf{v}_i = (v_{ix}, v_{iy}, v_{iz}) \in [-1, 1]^3. \tag{10.2}$$

The internal GCV vector $\mathbf{v}_i$ tracks the input GCV $\mathbf{v}_g$ in all situations other than if the higher level controller wishes to command step size adjustments for the purpose of balance preservation, and is assumed to be continuous, e.g. through filtering or slope limiting applied by the higher level controller.

Given the gait generator commands calculated by the higher level controllers, the KGG in turn calculates the required instantaneous joint position, joint effort and support coefficient commands, where support coefficients are a feed-forward estimation of the ratios of the weight of the robot that will be carried by each leg when the joint commands are executed. As discussed in Chapter 3, all of these outputs are used by the actuator control scheme to generate raw servo commands that are then sent electrically to the servo motors. The purpose of the actuator control scheme is to ensure good position tracking of the servo motors, by accounting in a feed-forward manner for factors such as battery voltage, joint friction, link inertia, and the relative loadings of the legs.

As discussed on page 168, the KGG and associated gait architecture is somewhat related to the Central Pattern Generator (CPG) presented in Chapter 8. Aside from a fundamental shift in how the walking waveforms are generated, one core difference between the two gait generators is that the corrective actions of the KGG are truly embedded and integrated inside the gait generator, as opposed to being imprecisely superimposed onto individual joints post factum and assuming that this does not adversely affect the gait profile in any way. A further difference and advantage of the KGG over the CPG is that the associated higher level balance controllers are abstracted away from having to work with the raw joint waveforms, leading to a more governed output and better separation of duties.

10.1.3 Aims for the Gait Generator

One of the aims of the gait generator is to allow fast walking with large step sizes. In order for this to be possible, the severity of the self-disturbances, impacts and nonessential spikes in acceleration during the gait cycle needs to be minimised as much as possible. This is embodied in the following list of desired properties for the gait generator:

- The final trajectories should be continuous in velocity and acceleration in all dimensions,
- The trunk orientation should remain as constant as possible, up to the moderate lateral oscillations required to assist support exchanges,
- The CoM should not undergo any sudden changes in height, and should generally rise and fall as little as possible during normal walking at a given gait velocity,
- The CoM should avoid start-stopping, i.e. having excessive oscillatory sagittal linear accelerations, as much as possible,
- During ideal walking, the feet should not significantly impact the ground and come down with zero instantaneous normal relative ground velocity, and,
- The feet should always contact the ground with zero instantaneous tangential velocity relative to the ground.

The KGG was designed at every stage with these aims in mind.

In addition to these general desired properties, there are also the following more specific considerations—that in particular were observed to be problems of, or were not addressed by, the CPG presented in Chapter 8. These need to be taken into account in the design of the gait generator:

- The robot torso is not necessarily nominally sagittally upright, so the floor may *nominally* be tilted relative to the torso frame, and the leg lift-swing profile needs to be able to adapt to that.
- The neutral leg pose during walking is not usually vertically downwards, so leg swing rotations in the hip for the purpose of stepping adversely affect the foot height, and in particular do so in a strongly asymmetrical fashion.
- The greater the swing of a leg, the more a shortening of the leg for the purpose of leg lifting also acts to reduce the step size.
- When falling in the direction of walking, if the ankle does not adjust for the extra tilt deviation, the toe may strike the ground prior to the remainder of the foot, preventing a full step from being taken, and possibly destabilising the robot.
- When falling in the direction of walking, the swing leg needs to be lifted higher towards the end of swing to avoid premature contact of the foot with the ground, and vice versa when falling in the direction opposite to walking. Crucially, the continued leg motion should also passively exert a restoring force to counteract the tilt instead of 'giving way' to it.

- For cases of compliant actuation, when a leg is lifted it tends to spring mechanically into a particular direction due to the sudden reduction in joint torque loadings. The leg is only reloaded when it has already reestablished contact with the ground, so leg lifting and placing is inherently asymmetrical and will tend to make the robot drift.

The KGG was designed to address these shortcomings of the CPG.

10.2 KEYPOINT GAIT GENERATION

The KGG uses systems of linear equations and cubic spline interpolation between a set of gait phase-dependent keypoints to dynamically generate walking trajectories for the robot in a constraint-based fashion. In this section, we explicitly define the ten corrective actions that are implemented as part of the KGG, and based on these definitions also explicitly delineate the inputs and outputs of the KGG.

We begin by introducing the notion of the nominal ground plane N, which is a plane in body-fixed coordinates that reflects the nominal orientation of the ground relative to the robot. The nominal ground plane embodies the fact that not all robots nominally walk with a completely upright torso, e.g. as generally assumed in Missura (2015). The N plane is defined by the (fixed) nominal phase pitch p_{yN} of the torso, and the corresponding nominal ground frame {N} is defined by the rotation

$$ {}^{B}_{N}R = R_y(-p_{yN}), \tag{10.3} $$

where $R_y(\theta)$ is notation for a pure y-rotation by θ radians. From Equation (10.3), it can be seen that the nominal ground normal vector ${}^{B}\hat{\mathbf{z}}_N \equiv \hat{\mathbf{z}}_N$ is given by

$$ \hat{\mathbf{z}}_N = \big(-\sin(p_{yN}),\, 0,\, \cos(p_{yN})\big). \tag{10.4} $$

The nominal ground plane N is used as a planar reference level for the generation of the arm and leg motion profiles. Given a nominal ground plane and a point that characterises the centre of a motion profile (i.e. motion centre point), the hip height of that profile is given by the distance perpendicular to the ground plane (i.e. along $\hat{\mathbf{z}}_N$) from the motion centre point to the hip centre point.

10.2.1 Corrective Actions

Numerous corrective actions have been implemented in the KGG. All of these corrective actions are illustrated in Figure 10.2—with the exception of the step size and timing actions—and are given as follows:

Step size: The sizes of the commanded steps are adjusted to capture the energy of the robot if it is falling (or predicted to fall) in a particular direction.

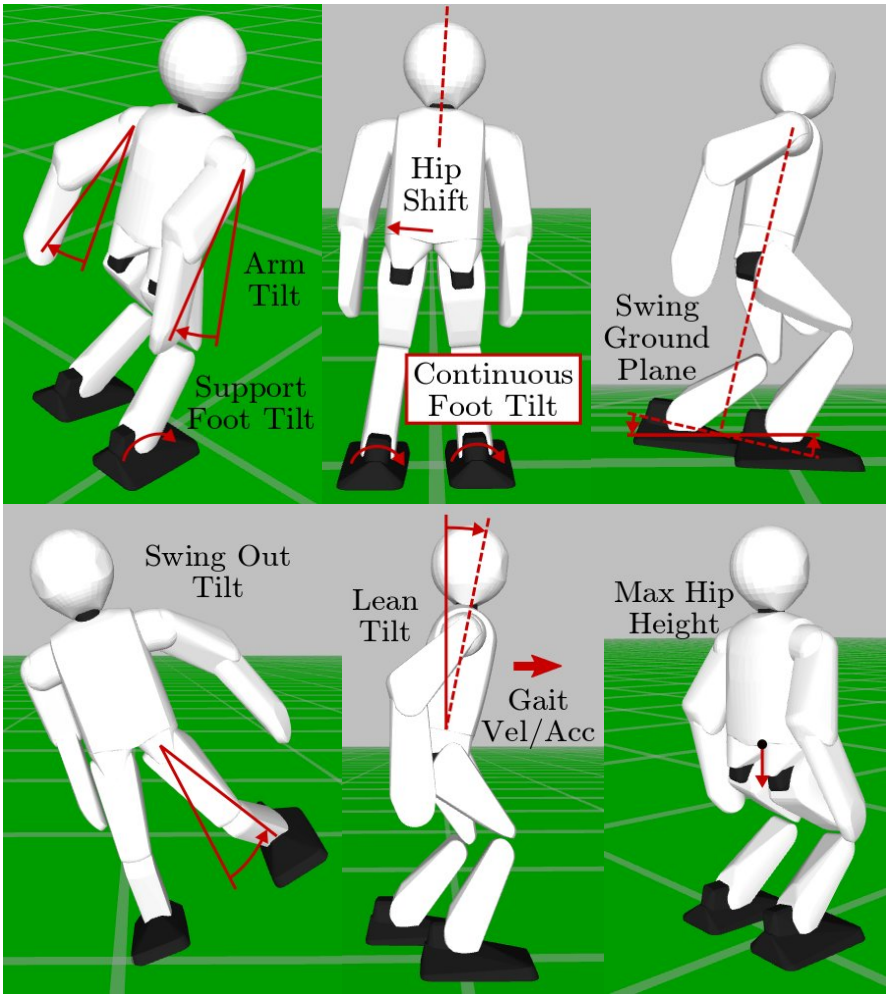


Figure 10.2: Diagrams of the various corrective actions implemented in the KGG. In most cases, the images and annotations are significant 1D simplifications of the true corrective actions for illustrative and explanatory purposes only. For instance, while the arm tilt shown in the top left image seems to be purely sagittal, in truth it is a 3D tilt rotation that is at no point separated into sagittal and/or lateral planes of motion. The arrows in general indicate the effect of the corrective actions in the depicted balance situations. The annotations in the swing ground plane image are trying to show that due to the forwards tilt of the robot, a normal step (lower dashed line) would have collided with the ground, while the adjusted step (solid line) avoids premature contact with the ground and executes the required step size despite the forwards tilt. Note that the swing ground plane is also 3D, and is not restricted in any way to be a pure sagittal tilt. The step size and timing corrective actions are not pictured in this figure as their effects are relatively clear.

Step timing: The rate at which the required stepping trajectories of the robot are executed is adjusted to speed up or slow down the gait as appropriate.

Arm tilt: The modelled CoM positions of the arms are *tilted*[4] to shift their weight and cause corresponding reaction moments.

Support foot tilt: The orientation of the current support foot is tilted to apply a restoring moment to the robot via the thereby altered ground reaction force. Smooth transitions to and from the support foot tilt are used during the double support phases to ensure that the resulting foot orientation trajectories remain continuous and differentiable.

Continuous foot tilt: An equal tilt offset is applied to both feet throughout the entire gait trajectory in order to consistently shift the balance of the robot in a particular 360° direction.

Hip shift: The position of the torso of the robot is adjusted in the **xy** plane to trim the centring of the robot's weight above its feet.

Maximum hip height: The hip height (see page 173) of the generated motion profile is limited to a certain maximum height to temporarily increase the passive stability of the gait.

Swing ground plane: The commanded foot trajectories are adapted to orientation deviations of the torso to avoid premature and/or belated foot strike. Effectively, the swing ground plane S is used as a planar reference level (similar to the nominal ground plane N) for adjusting the relative foot heights and tilts generated by the KGG.

Swing out tilt: The midpoint of the trajectory of the swing leg is tilted around the respective hip point to adjust the path taken by the swing leg to its target footstep location. The modified swing trajectory influences the balance of the robot via the inertial and gravitational effects of the swing leg.

Lean tilt: The torso of the robot is tilted at the hips to intentionally make the robot lean in a particular direction.

It should be noted that the corrective actions were not simply chosen arbitrarily on a basis of trial and error, but were the result of an analysis of the conceivable strategies for balanced bipedal walking, the results of which were presented in Section 10.1.1.

4 Note that all uses of the word 'tilted' precisely mean that a pure *tilt rotation* (see Section 5.3.2) is applied to the corresponding entity.

10.2.2 Gait Generator Interface

As indicated in Figure 10.1, the KGG takes as its inputs the required activation values of the various implemented corrective actions, and outputs the required joint commands for the actuator control scheme. The exact nature of these gait generator inputs and outputs are delineated in this section.

10.2.2.1 *Gait Generator Inputs*

A complete list of the inputs to the gait generator, covering all corrective action activation values, is as follows:

- The dimensionless internal GCV $\mathbf{v}_i = (v_{ix}, v_{iy}, v_{iz})$ to use to set the desired footstep size of the robot,
- The instantaneous gait frequency f_g (in rad/s) to use for updating the gait phase μ_i in each cycle,
- The arm tilt $P_a = (p_{xa}, p_{ya})$ to apply to the arms,
- The support foot tilt $P_s = (p_{xs}, p_{ys})$ to apply to the feet during their respective support phases,
- The continuous foot tilt $P_c = (p_{xc}, p_{yc})$ to apply to the feet as offsets throughout the entire gait trajectory,
- The dimensionless hip shift $\mathbf{s} = (s_x, s_y)$ to apply to the robot (in units of the inverse leg scale L_i),
- The maximum hip height H_{max} to allow relative to the feet for the generated motion profile (in units of the leg tip scale L_t),
- The 2D tilt phase rotation $P_S = (p_{xS}, p_{yS})$ defining the swing ground plane S relative to the nominal ground plane N,
- The swing out tilt $P_o = (p_{xo}, p_{yo})$ to apply to the midpoint of the leg swing trajectory, and,
- The lean tilt $P_l = (p_{xl}, p_{yl})$ to apply to the robot torso.

Given this list of gait generator inputs, it is important to note that:

(i) All strategies for balanced walking listed in Section 10.1.1 are covered by this palette of gait generator inputs.

(ii) Step size adjustments are effectuated via the internal GCV vector $\mathbf{v}_i$, and timing adjustments are effectuated via the gait frequency parameter f_g.

(iii) A small constant bias can be applied to $\mathbf{v}_i$ to negate any minor drifts in the real world walking performance of a particular robot.

(iv) All Cartesian actions are numerically expressed in dimensionless form relative to the nominal ground frame {N}, in units of either the inverse leg scale L_i or leg tip scale L_t (see Allgeuer, 2020).

(v) All rotation-based corrective actions are expressed as pure tilt rotations relative to frame {N}, in the 2D tilt phase space rotation representation $(p_x, p_y) \in \mathbb{P}^2$ (see Section 5.3.5.1, and Section 10.2.2.3 for the one slight exception).

The corrective action activation values are all expressed in a dimensionless manner so that near-identical values can be used for robots of different scales. The same approach, for similar reasons, is followed for all configurable constants used throughout the KGG algorithm.

10.2.2.2 *Gait Generator Outputs*

As required for the actuator control scheme (Chapter 3), the outputs of the gait generator are as follows:

- The commanded joint positions $\mathbf{q}_o \in \mathbb{R}^N$, specifying the desired angular positions of the N joints,
- The commanded joint efforts $\boldsymbol{\xi} \in [0, 1]^N$, specifying how stiff each joint should be, and,
- The commanded support coefficients κ_l, κ_r for the left and right legs respectively, specifying the proportion of the weight of the robot that is expected to be supported by each.

The vector $\boldsymbol{\xi}$ of commanded joint efforts returned by the KGG is kept constant at the desired manually configured values, and so does not need to be further addressed.

As previously mentioned, all of the inputs, outputs and parameters throughout the entire KGG have been chosen and expressed in such a way that they are dimensionless. This is important, as it means that their values are largely independent of size, scale and sample rate, and can be ported directly between different robots (even if, naturally, a renewed fine-tuning would still sometimes be of benefit). It is also worth noting that the entire keypoint gait generator is formulated with relatively few parameters and configuration variables to tune,[5] which offers the benefit of simplicity at least from the user's perspective.

10.2.2.3 *Swing Ground Plane Corrective Action*

As explained on page 175, the purpose of the swing ground plane input is to rotationally and positionally adjust the generated foot motion profiles relative to the ground in order to account for any tilt rotations of the robot torso. This can be done in many ways however,

5 At least, parameters that actually need to be tuned, and whose value are not just a clear analytical design decision from the outset.

and the exact choice of approach is quite important. In the CPG gait (see Chapter 8), the *virtual slope* corrective action performs a similar function to the swing ground plane (at least in the sagittal direction), but is not quite as effective because it induces a moderate form of positive feedback into the balance feedback loop—the more the robot tilts forwards, the more the virtual slope makes the robot lift its legs in front of its body, and therefore the more the robot will as a result end up leaning even further forwards. Despite working as intended to prevent premature foot strike, the problem with the virtual slope corrective action is that it at no point inherently exerts a corrective moment on the robot to counteract the current tilt deviation of the torso. Due to its implementation via foot z-height adjustments, a further problem of the virtual slope corrective action is that even if foot strike occurs exactly when intended, the virtual slope changes the effective commanded step sizes relative to the tilted ground.

The swing ground plane corrective action combats the first of these two problems by ensuring that the foot-ground contacts implicitly generate a restoring force that drives the torso of the robot back towards its nominal orientation. This is done just after foot strike by either intentionally extending the leg into the ground, or by retracting it more than usual. In terms of the second of the two problems, the swing ground plane leaves the exact desired footstep sizes untouched, and just rotates them into the S plane for further processing. It is assumed that the higher level controller(s) saturate the deviation of the swing ground plane from the nominal ground plane, for example through elliptical soft coercion (Allgeuer, 2020), to ensure that the KGG is not commanded to perform unreasonable ground plane adjustments.

Analogously to the N plane, the S plane is defined mathematically with respect to the robot torso by the phase roll and phase pitch variables $(p_{xS}, p_{yS}) \in \mathbb{P}^2$ (see Section 10.2.2.1), albeit not quite as directly. The swing ground normal vector ${}^{B}\hat{\mathbf{z}}_S \equiv \hat{\mathbf{z}}_S$ is defined to be the local z-vector (see Section 5.3.2.1) corresponding to the inverse tilt rotation of (p_{xS}, p_{yS}). That is,

$$\hat{\mathbf{z}}_S = P^l_{\hat{z}}\big(-p_{xS},\, -p_{yS},\, 0\big), \tag{10.5}$$

where $P^l_{\hat{z}}(\cdot,\cdot,\cdot)$ is the local z-vector corresponding to the enclosed relative tilt phase space parameters. The swing ground frame {S} is then defined to be the frame that results when the pure tilt rotation relative to {N} that rotates $\hat{\mathbf{z}}_N$ onto $\hat{\mathbf{z}}_S$ is applied to {N}. Mathematically, the rotation ${}^{N}_{S}R$ is calculated by evaluating

$${}^{N}\hat{\mathbf{z}}_S = {}^{B}_{N}R^T\, \hat{\mathbf{z}}_S, \tag{10.6}$$

and subsequently converting ${}^{N}\hat{\mathbf{z}}_S$ to a full rotation matrix using the constraint $\psi = 0$. The swing ground plane S is given by the **xy** plane

of the {S} frame, and is ideally set to correspond to the planar level of the ground relative to the robot at every instant of time.

10.2.3 Keypoint Trajectory Generation

For a detailed discussion of the inner workings of the KGG motion generation algorithm, refer to Allgeuer (2020).

10.2.4 Implementation

The Keypoint Gait Generator (KGG) and its surrounding gait architecture have been implemented in both Matlab and C++, and released open source in both cases (Allgeuer, 2018a; Team NimbRo, 2018).[6] The Matlab release serves as a reference implementation and test bed for the algorithms involved, and includes extensive plotting of figures and visualisations, as well as highly comprehensive unit testing to ensure that every component functions exactly as it should. The C++ release serves as the practical performance implementation, that through deep-seated modularity achieves truly flexible and robot-agnostic gait code, with zero code duplication. Virtually every single component of the code is customisable and replaceable, without affecting any of the other components at all (including notably the kinematics in use). This is possible due to the well-designed and standardised internal and external interfaces in use. The switching between available implementations of the various components and modules can also be performed freely at runtime. As such, the code is completely flexible and applicable to any robot, any kinematics, any set of higher level controllers, any step size generator, any gait odometry estimator, and even hypothetically any gait generator.

10.3 DISCUSSION

There are many positive aspects of the design choices that were made as part of the KGG. The main ones are discussed here.[7]

10.3.1 Characteristics of the Keypoint Gait Generator

A number of strategies for balanced walking were listed in Section 10.1.1. Every one of these strategies was specifically addressed

6 Matlab: https://github.com/AIS-Bonn/keypoint_gait_generator
C++: https://github.com/AIS-Bonn/humanoid_op_ros/tree/master/src/nimbro/motion/gait_engines/feed_gait

7 In order to fully understand some of the points of this discussion, more details about the keypoint trajectory generation algorithm may be required. This section has been included to at least discuss the benefits of the KGG, even if it is not easy to appreciate the arguments without knowing the KGG's inner workings (details in Allgeuer, 2020).

by one or more of the corrective actions listed in Section 10.2.1. For example,

- The swing leg trajectory adjustment was implemented in the form of swing out tilt,
- The utilisation of ground normal forces to apply a restoring moment to the robot was addressed by the handling of the swing ground plane S, and,
- The adjustment of the foot tilt relative to the ground at the instant of foot strike was realised as part of the continuous foot tilting.

The corrective actions that were built into the KGG are highly independent, and have very clear, specific functions and resulting effects on the motion of the walking robot. This makes the process of tuning relatively simple. As a result of all this, the corrective actions constitute a solid underlying framework on which higher level controllers can be constructed to stabilise the robot.

The desired properties of the gait generator were listed in Section 10.1.3 as part of the motivation and aims. It can be seen from the details of the KGG (Allgeuer, 2020) that each of these points have been addressed, or in some direct manner have influenced the way in which the trajectories are formed. For example, the desire to have continuous velocities and accelerations flowed directly into the constraint equations that were used for the generation of the keypoint velocities, as well as the final joining of the keypoints into abstract space trajectories.

As prescribed by the list of desired gait generator properties, the KGG was designed with the aim of minimal self-disturbances in mind, especially in terms of the desire for a smooth CoM profile that does not 'bounce' up and down during walking (which would require non-negligible oscillations in vertical momentum). The use of a nominal hip height H_{nom} for example, which accounts for the effects of the various ground planes and leaning, ensures that the hip—and therefore the CoM—remains at an essentially constant height off the ground for as long as the walking velocity and kinematic workspace permit it. Another example is the leg support keypoints, which are constructed in a way that ensures that, during ideal walking (i.e. $\mathrm{S} \equiv \mathrm{N}$), all of the support keypoints and paths between them are coplanar in the N plane, leading to minimal changes in CoM height during the gait cycle. The achieved coplanarity also leads to a diminution of ground impacts, and produces zero instantaneous normal relative ground velocities at foot strike and lift-off in the ideal case. The corresponding required zero instantaneous tangential relative ground velocities during the support phase—also listed as a desired property of the gait generator in Section 10.1.3—are for example addressed during the reconciliation

phase in terms of the adjustment of the **AC** and **BD** line segments, and by optimising all support keypoint velocities at once (Allgeuer, 2020).

Section 10.1.3 also listed other more specific considerations that needed to be, and are indeed taken into account by the KGG. For example, the fact that the robot torso is not always, not even nominally, sagittally upright, is handled in great depth by the definition and use of the N, S, I and J planes. Another example is the mitigation of premature toe strike by adjustment of the ankle tilt during the swing phase to account for torso tilt. This is achieved by the rotation of the foot orientations at the **F** keypoints, from the N to the S plane (Allgeuer, 2020). Numerous decisions were also made in the design of the KGG to ensure that none of the corrective actions alter the step size of the robot, avoiding effects such as some of the ones listed as specific considerations in Section 10.1.3. Self-collisions were also explicitly avoided through soft coercion of the required keypoints and poses at suitable instances along the gait generation pipeline.

In summary, the chosen gait architecture (see Section 10.1.2), and specifically the KGG, are advantageous and well-constructed because the KGG:

- Is analytic, implying that the computational intensity is low, and that there are guarantees about the properties of the generated trajectories,
- Can work with any step size generator and arm base motion generator, allowing the final generated trajectories to be made to have a similar walking style and properties as any purely open-loop gait that is known to essentially work on the robot,
- Can work with essentially any robot kinematics, with appropriate adjustment of the underlying robot-specific humanoid kinematic conversions,
- Has the inherent ability to flexibly perform all corrective actions that it requires, without a higher level controller ever later intervening on a joint trajectory level,
- Utilises virtually all 'normal' ways that a robot may wish to keep its balance while walking, somewhat similar to the reactions a human may have when disturbed while walking, i.e. not just step placement and timing,
- Works in a target-oriented manner that first establishes what exact properties the keypoints and trajectories should have, and then finds a way to calculate them so that they do, as opposed to being a more 'manually intuitively constructed' approach like the CPG, and,

- Is guaranteed to be kinematically feasible and safe for both the robot and human operator(s), through the extensive use of soft coercion and limiting to ensure that all calculations are stable and remain within the allowed workspace.

10.3.2 Advantages of the Abstract Space

The advantages of using the abstract space (Allgeuer, 2020) to help generate the keypoint locations and final trajectories are multifold:

- The formulation of the gait generator, at least in part, in the abstract space results in smooth and simple joint trajectories that have few high frequency components or sudden changes, and are more favourable for the servo motors. This comes about due to the close relationship between the joint and abstract spaces.
- Use of the abstract space simplifies the approach of complying with the workspace boundaries, i.e. ensuring that $\epsilon_l \geq 0$ and that all joints are in range, and provides an easy to work with guarantee of feasibility, along with a known margin thereof. It is significantly more difficult in purely the leg tip space to optimise and maximise a trajectory to 'just fit' inside the robot workspace, and it is especially difficult to make any associated guarantees, especially for varying robots and kinematics.
- Combined use of the abstract and leg tip spaces allows motion profiles to be easily constructed that are simultaneously joint-centred about a neutral halt pose, and inverse-centred in terms of the workspace and balance of the robot.
- The parameters of the abstract space are by design more intuitive and gait-related than both the joint and leg tip spaces, and allow natural concepts like hip swing and leg retraction to be used. This is especially useful for tuning, as using these, the required configuration can be accomplished with fewer and more intuitive parameters. Parameters relating to the abstract space are also more directly portable between robots of different dimensions.
- Planning motions in the abstract space gives relatively direct control over the associated joint velocities, unlike the leg tip space, but still possesses good interpolation performance.

10.4 EXPERIMENTAL RESULTS

The Keypoint Gait Generator (KGG) has been implemented on the igus Humanoid Open Platform and NimbRo-OP2X robots, as well as on a virtual igus Humanoid Open Platform in physical simulation. Similar to the open-loop Central Pattern Generator (CPG), the open-loop KGG

allows for semi-stable walking—not for indefinite amounts of time, and not in the face of medium to strong external disturbances—but is good enough for basic locomotion. Due to the analytically computed trajectory-sensitive nature of the gait, it has more difficulties than the CPG dealing with backlash and looseness in the servos and robot, but with use of the actuator control scheme these effects are at least for the most part mitigated.

Figure 10.3 shows sample plots of the generated joint space waveforms for the left and right limbs of an igus Humanoid Open Platform robot. The robot started from a standing position, was triggered to commence walking at time $t = 0.65\,\text{s}$, and accelerated to a constant GCV of $\mathbf{v}_i = (0.7,\, 0.4,\, 0.4)$ after first waiting 1.2 s to ensure that it gets into the correct walking rhythm. During this wait time, pose blending was applied to smoothly transition the robot from standing to walking.

After walking for a number of steps with the desired GCV, the robot was triggered to stop walking at time $t = 6.4\,\text{s}$. After decelerating to a GCV of zero and stopping walking at the next completed step, pose blending was used to return the robot to its halt pose. Note that throughout the entire experiment no corrective actions were used, so the visualised waveforms correspond directly to the open-loop KGG gait. In Figure 10.3, the green curve in the top plot corresponds to the instantaneous gait phase μ_i, which is propagated at a constant gait frequency of $f_g = 2.4\,\text{Hz}$, and the cyan curve corresponds to the pose blending factor b, for which a value of 0 means that the generated gait pose should be used, while a value of 1 means that the gait halt pose should be used. For intermediate values of b, linear interpolation is used on a joint level.

Video 10.1 provides a detailed kinematic demonstration of the trajectories generated by the KGG, including in particular sample visualisations of the various implemented corrective actions. Forwards, sidewards and turning motions are shown in isolation before being combined with the stepping motion model (Allgeuer, 2020) to achieve full 3D gait odometry. The effects of the individual corrective actions are shown by manually setting activation values live with sliders and visualising the resulting motions of the robot.

Further demonstrations of the KGG and its corrective actions are performed implicitly in the next chapter (see Section 11.3) as part of the evaluation of the tilt phase controller.

10.5 CONCLUSION

An analytic method for the generation of bipedal gait trajectories was presented in this chapter. The method incorporates a myriad of 3D corrective actions for the purpose of gait stabilisation into the underlying stepping motions of the robot. The so-called Keypoint Gait Generator (KGG), along with its corresponding overarching gait

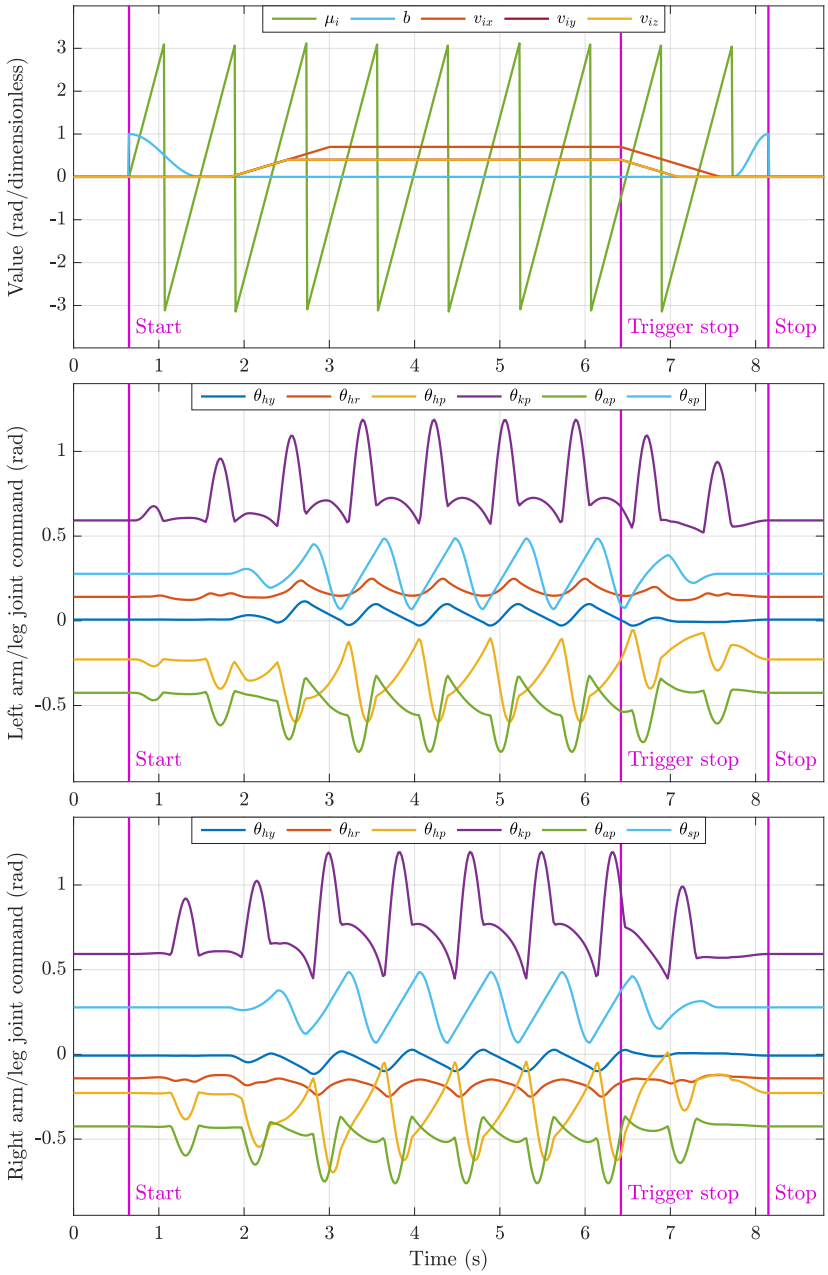


Figure 10.3: Sample output waveforms of the Keypoint Gait Generator (KGG) for an input GCV of $\mathbf{v}_g = (0.7, 0.4, 0.4)$, and a constant input instantaneous gait frequency of $f_g = 2.4\,\text{Hz}$. The top plot shows the values of the gait phase μ_i and internal GCV vector $\mathbf{v}_i$ (note that v_{iy} is hidden by v_{iz}), as well as the dimensionless pose blending factor b. The KGG gait is activated at time $t = 0.65\,\text{s}$, and receives the trigger to stop walking at time $t = 6.4\,\text{s}$. After a further two steps, during which the robot slows down to an internal GCV of zero, the robot stops walking and blends back to the halt pose. Note how the waveforms for the left limbs (middle plot) are in general antiphase to the corresponding ones for the right limbs (bottom plot). The effect of the initial GCV zero time (1.2 s), in addition to the effect of the GCV slope limiting, can be seen in the top plot.

Video 10.1: Kinematic demonstration of the open-loop Keypoint Gait Generator (KGG), including the various corrective actions it is able to actuate. The stepping motion model as applied to the KGG is also demonstrated kinematically in the context of a soccer field.
https://youtu.be/XIfxTRLFbwI
Kinematic View of Keypoint Gait Generated Open-loop Walking

architecture, which was presented for the purpose of context, is a powerful building block for the construction of complex feedback gaits that require the ability to go beyond just step placement and timing. The role and effect of each corrective action has preliminarily been demonstrated, in anticipation of the tilt phase controller presented in Chapter 11, which properly drives the corrective actions. Two implementations of the KGG and the associated kinematics calculations have been released open source, and can be complemented by higher level controllers, like the tilt phase controller, that utilise the full potential of the 3D corrective actions to preserve the balance of the robot.

11 TILT PHASE CONTROLLER

As discussed in detail in Section 10.1.1, many feedback strategies exist by which a robot can seek to maintain its balance while walking bipedally. In related works, the online adjustment of step size and timing is often considered, e.g. by Kryczka et al. (2015). While these are quite effective strategies if done right, numerous other forms of feedback beyond just ankle torque, like for example arm motions and swing leg trajectory adjustments, can also be employed to significantly increase the stability of the robot, especially in a broader spectrum of walking situations. For instance, step size feedback cannot help when a robot is about to tip over the outside of one of its feet, and cannot effectively correct for systematic biases in the robot. It also has little effect until the next step is actually taken, meaning that there is an inherent dead time until disturbances can be counteracted. Furthermore, changing the target step size modifies the footstep locations, and thus directly leads to the non-realisation of footstep plans. As such, step size feedback is envisioned as a valuable tool for gait stabilisation, but one that ideally only activates for large disturbances, when there really is no other option. The corrective actions presented as part of the Keypoint Gait Generator (KGG) in Chapter 10 aim to address all of these issues. The tilt phase controller presented in this chapter (see Figure 11.1) is a higher level controller that suitably drives the corrective actions of the KGG, and solves the more general problem of how to achieve balanced push-resistant walking with minimal changes to the walking intent of the robot.

In the interest of reducing the required tuning effort and making the tilt phase controller applicable to low-cost robots with cheap sensors and actuators, the use of physical models in the feedback path is avoided. Physical models usually require extensive model identification and tuning to sufficiently resemble the behaviour of a robot, and even then, cheap actuators lead to significant nonlinearities that can often cause such models to have poor results or even fail. Physical models are also frequently quite sensitive to small changes in the robot, making frequent retuning necessary. The implementation difficulty and cost of good sensors also limits the type and accuracy of sensors that can be incorporated into a humanoid robot. In order to facilitate the greatest possible portability of the developed gait between robots of different builds and proportions—a design decision that is supported by the nominally model-free nature of the gait—only the presence of a 6-axis Inertial Measurement Unit (IMU) sensor is assumed. Apart from that, no additional sensors, joint positions, robot masses or

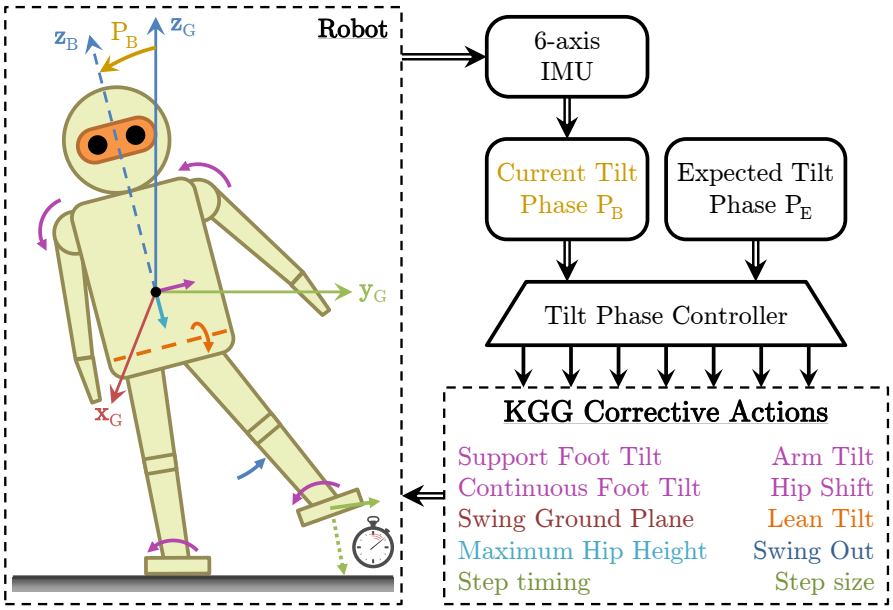


Figure 11.1: Overview of the tilt phase controller approach to walking, and how the controller interacts with the robot, the IMU state estimation, and the Keypoint Gait Generator (KGG). All ten different corrective actions of the KGG are utilised and activated by the tilt phase controller.

inertias are assumed at all for the tilt phase controller. The only further 'assumptions' that are made are trivial, like for example that tilting of the support foot in one direction makes the robot tendentially tilt in the other.

When compared to the direct fused angle feedback controller (see Chapter 9), the main advancements of the tilt phase controller lie in the methods of calculation of the various feedback components, which are extensions of the direct fused angle controller only for the Proportional-Derivative (PD), leaning and timing components. Many of the corrective actions are also completely new or substantially revised, with the remaining ones being extended to a full 3D treatment, so as in particular not to treat the sagittal and lateral directions independently. This is aided by the novel use of the tilt phase space as a source of truly axisymmetric orientation feedback.

Ultimately, the tilt phase controller (in combination with the KGG) seeks to demonstrate that not overly complex feedback mechanisms with very limited information of the robot suffice to produce a very stable gait, capable of rejecting significant disturbances. The presented feedback controller has been released open source in C++[1], and works in conjunction with the Humanoid Open Platform ROS Software (Team NimbRo, 2018).

1 https://github.com/AIS-Bonn/humanoid_op_ros/tree/master/src/nimbro/motion/gait_engines/feed_gait/include/feed_gait/model/tilt_phase

11.1 GAIT ARCHITECTURE

As illustrated in Figure 10.1, and described in detail in Section 10.1.2, the overall gait architecture consists of three layers, namely

- The **actuator control scheme**, responsible for generating the low-level servo commands that are sent out to the robot hardware,
- The **keypoint gait generator**, responsible for generating the required walking motions in the form of joint angle, joint effort, and support coefficient waveforms, and,
- The **higher level controller**, i.e. the tilt phase controller in this case, responsible for calculating the required activations (see Section 10.2.2.1) of the KGG corrective actions based on the estimated state of the robot and input Gait Command Velocity (GCV). The implemented KGG corrective actions are listed in Figure 11.1 and Section 10.2.1, and are illustrated in detail in Figure 10.2.

11.1.1 Gait Command Velocity

The primary input to the tilt phase controller is given by the dimensionless GCV vector

$$\mathbf{v}_g = (v_{gx}, v_{gy}, v_{gz}) \in [-1, 1]^3, \tag{11.1}$$

referred to as the input GCV vector. Based on this, and the effects of any step size feedback calculated from the robot state, the tilt phase controller needs to calculate the required internal GCV vector

$$\mathbf{v}_i = (v_{ix}, v_{iy}, v_{iz}) \in [-1, 1]^3, \tag{11.2}$$

that the KGG should then use for the generation of the desired walking waveforms. As a first step, p-norm limiting as per Equation (8.3) is applied in order to limit the total magnitude of the input GCV.

As $\mathbf{v}_g$ is an external input, and is thus not in any way guaranteed to be smooth or continuous (for instance, it may come directly from a joystick), care needs to be taken to apply appropriate GCV smoothing, without thereby limiting the ability of the controller to react quickly to large disturbances. The GCV smoothing approach used by the tilt phase controller is to allow each component of the controller to add a contribution to one of three intermediate output GCV vectors, namely $\mathbf{v}_{LF}$, $\mathbf{v}_{HF}$ and $\mathbf{v}_{EOS}$, where

- $\mathbf{v}_{LF}$ is the low frequency GCV, which is significantly slope-limited (yielding $\hat{\mathbf{v}}_{LF}$) to ensure that it is safe for the robot no matter what values are passed to it,

- $\mathbf{v}_{HF}$ is the high frequency GCV, which is only mildly slope-limited (yielding $\hat{\mathbf{v}}_{HF}$) to ensure that more sudden deviations in step size are possible, while still avoiding the possibility of discontinuities, and,
- $\mathbf{v}_{EOS}$ is the end-of-step GCV, which is a step size component that is linearly faded to over the course of the current KGG step.

Thus, if $\hat{\mathbf{v}}_{LF}$, $\hat{\mathbf{v}}_{HF}$ and $\hat{\mathbf{v}}_{EOS}$ are GCV vectors that start at $\mathbf{0}$ at the beginning of the gait, then in each execution cycle of the tilt phase controller, these vectors are updated using the equations

$$\hat{\mathbf{v}}_{LF} \leftarrow \hat{\mathbf{v}}_{LF} + \Delta t \,\text{coerce}\big(\tfrac{1}{\Delta t}(\mathbf{v}_{LF} - \hat{\mathbf{v}}_{LF}),\, -\mathbf{k}_{lfs},\, \mathbf{k}_{lfs}\big), \tag{11.3a}$$

$$\hat{\mathbf{v}}_{HF} \leftarrow \hat{\mathbf{v}}_{HF} + \Delta t \,\text{coerce}\big(\tfrac{1}{\Delta t}(\mathbf{v}_{HF} - \hat{\mathbf{v}}_{HF}),\, -\mathbf{k}_{hfs},\, \mathbf{k}_{hfs}\big), \tag{11.3b}$$

$$\hat{\mathbf{v}}_{EOS} \leftarrow \hat{\mathbf{v}}_{EOS} + \Delta t \,\text{coerce}\big(\tfrac{1}{\hat{t}_s}(\mathbf{v}_{EOS} - \hat{\mathbf{v}}_{EOS}),\, -\mathbf{k}_{eos},\, \mathbf{k}_{eos}\big), \tag{11.3c}$$

where Δt is the execution cycle time, $\hat{t}_s$ is the current predicted time to step, and $\mathbf{k}_*$ are suitably configured 3D vector constants specifying the required slope limits. The final GCV vector $\mathbf{v}_i$ that is then passed to the KGG in each cycle is given by

$$\mathbf{v}_i = \hat{\mathbf{v}}_{LF} + \hat{\mathbf{v}}_{HF} + \hat{\mathbf{v}}_{EOS}. \tag{11.4}$$

The low frequency GCV component $\mathbf{v}_{LF}$ works much like the manual GCV slope limiting used for the Central Pattern Generator (CPG) in Section 8.1.1, and is set to the value of $\mathbf{v}_g$ at the beginning of each execution cycle of the tilt phase controller. The $\mathbf{v}_{HF}$ component is a high frequency version of $\mathbf{v}_{LF}$ (albeit not also initialised to $\mathbf{v}_g$ of course), and the end-of-step GCV $\mathbf{v}_{EOS}$ was inspired by the way the CPG makes provisions for step size feedback (see Equation (8.12) in Section 8.1.3.2).

11.1.2 The Tilt Phase Space

One significant difference between the tilt phase controller and the direct fused angle controller from Chapter 9 is the full 3D treatment given to the corrective actions, made possible in part by the use of the *tilt phase space* (see Section 5.3.5) instead of *fused angles* (see Section 5.3.4). While fused angles work very well for separate treatments of the sagittal and lateral planes, the tilt phase space has advantages for concurrent treatments, in particular in relation to *magnitude axisymmetry* (see Section 6.2.4.3). This property is important in ensuring that feedback magnitudes are the same scale no matter what continuous direction the robot is tilted in. Furthermore, the tilt phase parameters share all of the critical advantages (see page 83) that fused angles have over lesser options, like in particular Euler angles, mainly due to the tight relationship between the two representations. Further advantages

of the tilt phase space include that it can naturally represent and deal with tilt rotations of more than 180°, and that using it, tilt rotations can be unambiguously commutatively added (see Section 5.3.5.3). Both of these are useful features in feedback scenarios where rotation deviation feedback components are scaled by arbitrary gains and need to be combined in a well-defined and logical manner.

If $(\psi, \gamma, \alpha) \in \mathbb{T}$ are the *tilt angles* parameters of a rotation, where ψ is the fused yaw, γ is the tilt axis angle and α is the tilt angle (see Section 5.3.3), the corresponding *3D tilt phase* representation is given by

$$P = (p_x, p_y, p_z) = (\alpha \cos\gamma, \alpha \sin\gamma, \psi) \in \mathbb{P}^3. \tag{11.5}$$

Omitting the yaw component, the *2D tilt phase* representation of the resulting *tilt rotation component* is given by

$$P = (p_x, p_y) = (\alpha \cos\gamma, \alpha \sin\gamma) \in \mathbb{P}^2. \tag{11.6}$$

Note that the relative and absolute 2D tilt phase spaces (see Section 5.3.5.2) are identical for pure tilt rotations ($\psi = 0$), so for the majority of the rotation-based corrective actions, just the relative tilt phase notation is used (i.e. no tildes).

11.2 TILT PHASE CONTROLLER FORMULATION

The aim of the tilt phase controller is to calculate corrective action activation values that will keep the robot balanced and walking in the intended direction. An overview of the feedback pipeline corresponding to the tilt phase controller is shown in Figure 11.2. How the activation values are calculated specifically for each of the individual corrective actions is presented in detail in the following subsections.

11.2.1 Preliminaries

We recall from the Keypoint Gait Generator (KGG) described in the previous chapter that

- The nominal ground plane N (along with the corresponding nominal ground frame {N}) is a plane in body-fixed coordinates that reflects the nominal orientation of the ground relative to the robot,
- All Cartesian corrective actions are numerically expressed in dimensionless form relative to the nominal ground frame {N}, in units of either the inverse leg scale L_i or the leg tip scale L_t, and,
- All rotation-based corrective actions are expressed as pure tilt rotations in the 2D tilt phase space relative to the {N} frame.

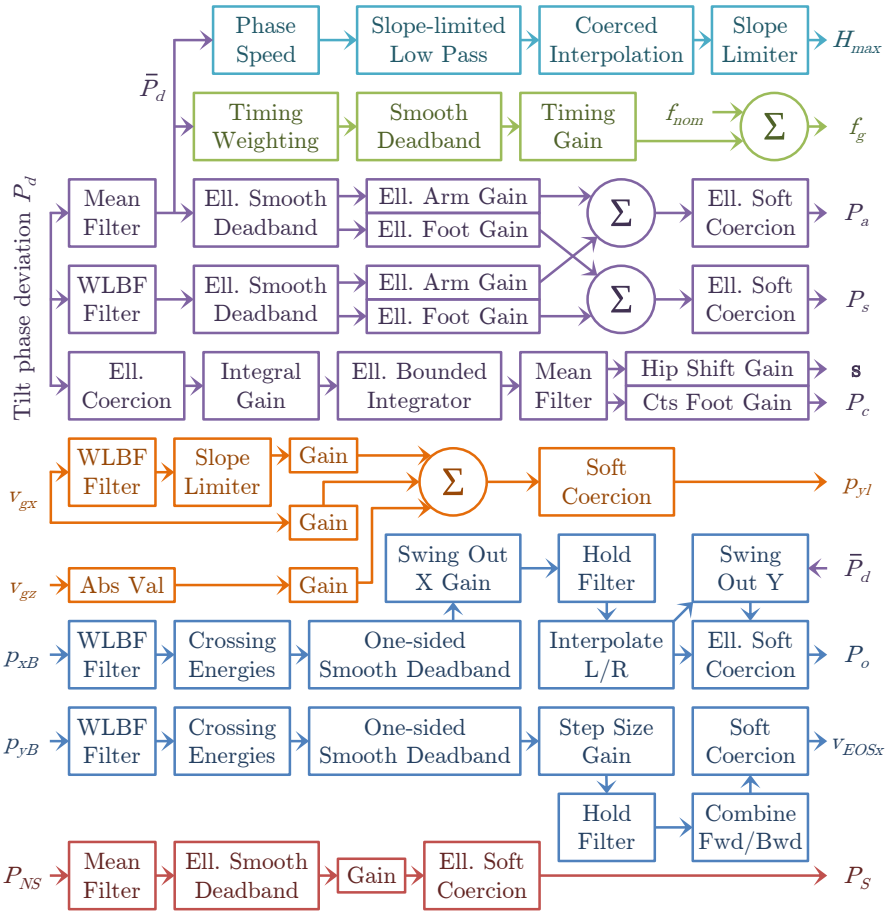


Figure 11.2: Overview of the entire tilt phase controller feedback pipeline, including the calculation of the activations of all ten KGG corrective actions. Refer to Section 11.2.2 for details on how the tilt phase deviation P_d is calculated, and refer to Section 11.2.3 onwards for more details about each individual calculation pipeline.

Aside from this, we also note that the tilt phase controller uses a number of recurring filters and mathematical constructs, briefly discussed as follows:

Filters The mean and Weighted Line of Best Fit (WLBF) filters used in Chapter 9 have been taken and generalised to n dimensions. The former computes the moving average of an n-dimensional vector, while the latter performs weighted time-based linear least squares regression to smooth and estimate the time derivative of n-dimensional data. The advantages of WLBF filters over alternatives for the numerical computation of derivatives are discussed in depth in Section 9.3.2.

Coerced interpolation Standard linear interpolation can lead to extrapolation outside of the interval domain. Coerced interpolation limits the input variable to ensure that the output cannot be outside the range of the two data points.

Soft coercion The soft coercion function that was used in Chapter 9 has been taken and extended ellipsoidally to n dimensions. Given an input vector $\mathbf{x}$, the principal semi-axis lengths $\mathbf{a}$ of the required limiting ellipsoid $\mathcal{E}$, and a scalar soft coercion buffer b, scalar soft coercion is applied to the magnitude of $\mathbf{x}$ based on b and the radius of $\mathcal{E}$ in that direction. This method of higher dimensional soft coercion is significantly better than applying soft limits along each axis independently, as the latter would result in unexpectedly large radial limits on the diagonals between the principal axes.

Smooth deadband The smooth deadband from Chapter 9 has been taken and extended ellipsoidally to n dimensions. Given an input vector $\mathbf{x}$ and the principal semi-axis lengths $\mathbf{a}$ of the deadband ellipsoid $\mathcal{E}$, scalar smooth deadband is applied radially along $\mathbf{x}$ with a deadband radius corresponding to the radius of $\mathcal{E}$ in that direction.

11.2.2 Deviation Tilt

In the tilt phase controller, most of the calculated corrective action activation values depend directly on how the robot is currently tilted relative to what is expected for the current gait phase μ_i. As such, the first step in the feedback pipeline is to retrieve the current trunk orientation from the attitude estimation (see Chapter 7), and express it as a 2D tilt phase orientation

$$P_B = (p_{xB},\, p_{yB}). \tag{11.7}$$

As only the tilt rotation component (i.e. no heading component) of the trunk orientation is required to construct P_B, a single 6-axis IMU suffices for the estimation of P_B (see Section 7.6.2).

The two estimated tilt phase parameters from Equation (11.7), p_{xB} and p_{yB}, follow an expected periodic pattern as a function of the gait phase during ideal undisturbed walking of the robot. We model this periodic pattern as a function

$$f_{exp} : (-\pi, \pi] \to \mathbb{P}^2,\ \mu_i \mapsto P_E, \tag{11.8}$$

where

$$P_E = (p_{xE},\, p_{yE}) \tag{11.9}$$

is the so-called *expected* 2D tilt phase orientation. For the purposes of the tilt phase controller, a sinusoidal function with an offset is used to model each the expected phase roll p_{xE} and phase pitch p_{yE},

as this was seen to sufficiently generalise and describe the observed behaviour of the robot. As such, we define

$$p_{xE} = k_{eox} + k_{emx} \sin(\mu_i - k_{epx}), \tag{11.10a}$$
$$p_{yE} = k_{eoy} + k_{emy} \sin(\mu_i - k_{epy}), \tag{11.10b}$$

where k_* are constants that are tuned based on walking experiments to make P_E closely track the observed P_B data (for as long as the robot is walking undisturbed).

It can be seen by definition that the deviation between P_B and P_E is a measure of the magnitude and direction that the orientation of the trunk is currently disturbed by. We therefore construct a trunk orientation error feedback term that quantifies the unique 3D rotation q_d that rotates P_E onto P_B, up to a small possible deviation in fused yaw ψ_d. What makes q_d unique is that we constrain it to be a pure tilt rotation relative to the nominal ground frame {N}, as all of the KGG corrective actions that will use q_d as a source of feedback are defined to act relative to the N plane (see Section 11.2.1). Mathematically, if q_B and q_E are the quaternions corresponding to the pure tilt rotations P_B and P_E respectively, then we can express the quaternion q_d as

$$\begin{aligned} q_d \equiv {}^{NE}_{B}q &= {}^{N}_{B}q \, {}^{E}_{B}q \, {}^{N}_{B}q^* \\ &= {}^{N}_{B}q \left(q_E^* \, q_z(\psi_d) \, q_B \right) {}^{N}_{B}q^* \\ &= q_y(p_{yN}) \, q_E^* \, q_z(\psi_d) \, q_B \, q_y(-p_{yN}), \end{aligned} \tag{11.11}$$

where $q_y(\cdot)$ and $q_z(\cdot)$ correspond to pure quaternion y and z-rotations respectively, and p_{yN} is the nominal phase pitch of the torso (see page 173). The value of ψ_d is calculated by solving[2]

$$\Psi(q_d) = 0, \tag{11.12}$$

with ψ_d as the only unknown—i.e. by solving the aforementioned constraint that q_d is a pure tilt rotation. The value of q_d can subsequently be evaluated using Equation (11.11), and converted to the 2D tilt phase space representation to give the deviation tilt $P_d = (p_{xd}, p_{yd})$.

To summarise, the deviation tilt P_d is a quantity that indicates in which direction the robot is tilted relative to the N plane away from where it is actually normally expected to be (at that specific moment of walking). Consequently, if the negative (i.e. inverse) of P_d were to be used as the activation of a rotation-based corrective action (such as the arm tilt P_a), it would be expected that the resulting effect on the robot would be to tilt it towards its expected orientation P_E, as desired. This is the idea behind the proportional feedback in Section 11.2.3, with additional measures being incorporated to address scaling and sensor noise issues.

2 The notation $\Psi(q_d)$ refers to the fused yaw of the quaternion q_d.

11.2.3 PD Feedback: Arm and Support Foot Tilt

The most important thing for the stability of the robot in the short term is to ensure that transient disturbances, such as pushes or steps on larger irregularities in the ground, are swiftly counteracted with little delay. 3D rotational proportional (P) and derivative (D) feedback components, activating the arm tilt and support foot tilt corrective actions, are used for this purpose. The arm tilt rotates the Centre of Mass (CoM) of the arms out in the required direction relative to the N plane, so as to bias the CoM of the entire robot and apply a reactive moment on the torso that helps mitigate the disturbance. At the same time, the support foot tilt applies smooth corrections to the tilt of each foot during its respective support phase, to push the robot back towards its expected orientation.

In order to reduce signal noise, the proportional feedback path of the tilt phase controller first mean filters the deviation tilt using a small filter order, yielding $\bar{P}_d$. This tilt phase is then elliptically smooth deadbanded to ensure that P feedback only takes effect when the robot is non-negligibly away from its expected orientation P_E. An elliptically directionally dependent gain is then applied to the resulting tilt phase, once independently for each the arm tilt and the support foot tilt, to get the corresponding P feedback components (see upper purple section in Figure 11.2). The gain in each case is calculated elliptically from specifications of the required gains in the sagittal and lateral directions. Importantly, the directions of the final proportional feedback components are both unchanged from $\bar{P}_d$—all changes are purely radial.

In the derivative feedback path, a smoothed derivative of P_d is first computed using a 2D WLBF filter. A WLBF filter was chosen for its many advantages, including, amongst other things, its favourable balance between robustness to high frequency noise and responsiveness to input transients (see Section 9.3.2). The computed derivative is elliptically smooth deadbanded to ensure that D feedback only takes effect if the robot torso has a non-negligible angular velocity relative to its expected orientation. Then, as for the P feedback, independent elliptically directionally dependent gains are applied to get the D feedback components for the two nominated PD corrective actions.

Once the separate P and D components have been calculated, they are combined using tilt vector addition (see Section 5.3.5.3) and elliptically soft-coerced to obtain the final required activation values P_a and P_s (see Section 10.2.2.1). Note that although it is not generally acceptable to just add 3D rotations, the special properties of the tilt phase space allow us to do just that in a meaningful, unambiguous, self-consistent and mathematically supported way. In fact, the tilt phase space turns tilt rotations into a well-defined vector space over

$\mathbb{R}$, explaining why the scaling and addition of tilt phases used in this chapter is actually mathematically valid and geometrically meaningful.

The tuning of the PD feedback paths is relatively straightforward, as there are only a few gains, and each gain has a clearly visible effect on the robot. The P feedback is tuned first, and then appropriated with D feedback to add damping to the system and limit oscillatory behaviour.

11.2.4 I Feedback: Hip Shift and Continuous Foot Tilt

The implemented PD feedback works well for rejecting the majority of short-term transient disturbances, but if there are continued regular disturbances or a systematic imbalance in the robot, the PD feedback (in combination with other corrective actions) will constantly need to act to oppose them. PD feedback can only act however, if there is a non-zero position and/or velocity error present. Thus, without integral (I) feedback, which in this case is applied in the form of the hip shift and continuous foot tilt corrective actions, the system in such a case would at best settle with a steady state deviation to normal walking, which is undesirable. The continuous foot tilt applies continuous tilt corrections to the generated orientations of the feet, while the hip shift applies an offset to the generated hip positions relative to the feet. Both are applied relative to the N plane, and bias the balance of the robot in the desired direction to overcome systematic errors in the walking of the robot. The implemented I feedback can effectively reduce the need for fine tuning of the robot, and make the gait insensitive to small changes in the hardware or walking surface that would otherwise have been noticeable in the resulting walking quality.

Starting with the deviation tilt P_d, standard elliptical coercion is first applied, the output of which is scaled by a scalar integral gain (see lower purple section in Figure 11.2). A scalar gain is used instead of a directionally dependent one, so as not to distort the 'aggregated' direction of measured instability once integration is applied. The initial coercion is useful to ensure that the integrated value is determined predominantly by small and consistent deviation tilts, rather than large and brief transients, which have little correlation to the finer balance of the robot. The coerced and scaled deviation tilt is passed to an *elliptically bounded integrator* (Allgeuer, 2020). This kind of integrator performs updates of 2D trapezoidal integration and elliptical soft coercion in each step. Note that the two steps are interlinked, as the output of the coercion is used as the starting point for the next integral update. Apart from providing the required integral behaviour to eliminate steady state errors, and ensuring that the integral remains conveniently bounded, this special kind of integrator also inherently combats integral windup in a more effective way than other options. The initial coercion of P_d reduces the extent to which integrator

windup is possible, but the elliptically bounded integrator ensures that the integral can move away from the elliptical boundary as quickly as it can approach it, and that it cannot get stuck there due to 'over-integration'. The dynamic response of the corrective actions is on a much quicker time scale than the integration, so this is the main type of windup concern in the integral feedback pipeline.

The integrated tilt phase value is passed through a final mean filter to combat ripple, before being separately scaled to yield the final corrective action activations P_c and $\mathbf{s}$ (see Section 10.2.2.1). The order of the mean filter[3] is chosen to correspond exactly to the duration of an even number of steps at the nominal gait frequency. Due to the periodicity and general regularity of the gait, this leads to almost perfect cancellation of ripple. This would not be achievable with an Infinite Impulse Response (IIR) low-pass filter, which would also have the downside of not as efficiently 'forgetting', i.e. diminishing the influence of, old data.

During tuning, it is attempted to keep at least one of the elliptical integral bounds at 1. This form of normalisation makes the tuning of the integral and corrective action gains relatively simple and intuitive, as the former gain then inversely relates to the parameters of the initial elliptical coercion, and the latter gain then corresponds to the desired maximum magnitude of each respective corrective action.

11.2.5 Leaning

The lean tilt corrective action could be activated based on the integral feedback path, but this would promote suboptimal tilted walking postures of the robot, in part because the lean tilt directly changes the measured orientation of the trunk without this necessarily ameliorating the overall balance of the robot. Leaning driven by the PD feedback would also be possible, but although maybe not immediately intuitively obvious, neither attempting to lean forwards nor backwards is particularly useful for dissipating energy when the robot is disturbed and for example falling dynamically forwards. Pure hip rotations are only useful if they are performed quite significantly, early enough so as to precede tipping, and in specifically controlled scenarios, e.g. clean push disturbances, purely in the sagittal direction, with the robot not walking or stopping walking immediately after the disturbance, and so on. In most other situations, reactive leaning has a negative impact on walking robustness. As such, only feed-forward leaning components based on the Gait Command Velocity (GCV) are implemented. These seek to avoid disturbances due to changes in walking velocity before they even occur. The gait acceleration is first estimated using a WLBF filter followed by a slope limiter (see orange section in Figure 11.2). A linear combination of the sagittal velocity v_{gx}, absolute

3 That is, the number of data points that are used in the evaluation of the mean filter.

turning velocity $|v_{gz}|$ and sagittal gait acceleration is then taken and soft-coerced to give p_{yl} (see Section 10.2.2.1).[4] Feed-forward sagittal leaning in particular helps during strong turns, and when starting or stopping forwards walking.

11.2.6 Swing Out

The robot is said to be on a lateral crossing trajectory if it has enough lateral momentum to tip over the outside of its support foot. This is a difficult situation, as no simple reactive stepping or waiting strategy can prevent the fall. Acting alongside the arm tilt and support foot tilt actions, the swing out tilt was designed specifically to allow recovery from lateral crossing trajectories. When significant lateral energy is detected, the current swing leg is rotated outwards to bias the balance of the robot, and to apply a reactive moment that dissipates crossing energy.

The lateral tilt phase p_{xB} is first smoothed and differentiated using a WLBF filter. The line of best fit is evaluated at the mean of the recorded data points so that the resulting estimated phase $\hat{p}_{xB}$ (smoothed) and phase velocity $\dot{\hat{p}}_{xB}$ (differentiated) are synchronised in time. The values of $\hat{p}_{xB}$ and $\dot{\hat{p}}_{xB}$ are used to calculate the left and right crossing angles ϕ_L and ϕ_R respectively, as well as the corresponding crossing velocities $\dot{\phi}_L$ and $\dot{\phi}_R$, using the equations

$$\phi_L = p_{xL} - \hat{p}_{xB}, \qquad \dot{\phi}_L = -\dot{\hat{p}}_{xB}, \tag{11.13a}$$

$$\phi_R = \hat{p}_{xB} - p_{xR}, \qquad \dot{\phi}_R = \dot{\hat{p}}_{xB}, \tag{11.13b}$$

where p_{xL} and p_{xR} are tuned constants corresponding to the values of $\hat{p}_{xB}$ at the cusp of crossing for the left and right legs respectively. Both ϕ_L and ϕ_R are normally negative during normal walking, but become positive in the case of crossing over the respective leg, and more positive as the robot then subsequently falls to the ground.

We model the behaviour of the lateral tilt phase $\hat{p}_{xB}$ as approximately following the nonlinear pendulum model

$$\ddot{\phi}_X = C^2 \sin \phi_X, \tag{11.14}$$

where $X = L$, R, and C is the pendulum constant. This leads to the core result, and thereby assumption, that the so-called orbital energy

$$E_X(\phi_X, \dot{\phi}_X) = \tfrac{1}{C^2}\dot{\phi}_X^2 + 2(\cos\phi_X - 1) \tag{11.15}$$

remains constant over any undisturbed trajectory.[5] Note that the orbital energy has intentionally been divided by $\frac{1}{2}C^2$ from its usual

4 Actually, to avoid possible discontinuities with $\mathbf{v}_g$ (as it is an external input), $\hat{\mathbf{v}}_{LF}$ is used as the basis of leaning feedback instead (see Section 11.1.1). For all intents and purposes, $\hat{\mathbf{v}}_{LF}$ is essentially just a slope-limited version of $\mathbf{v}_g$ though.

5 It is easy to prove that the nonlinear orbital energy stays constant by expanding $\frac{d}{dt}(E_X(\phi_X, \dot{\phi}_X))$.

form so as to make it dimensionless. This is of significant benefit later on when it comes to the tuning of the swing out feedback mechanism.

One can observe from Equation (11.15) that $E_X(\phi_X, \dot{\phi}_X)$ is the sum of a kinetic energy component (based on $\dot{\phi}_X$) and a potential energy component (based on ϕ_X). Depending on the signs of $\dot{\phi}_X$ and ϕ_X, these energy components either help or hinder crossing. As such, we introduce the notion of the crossing energy $CE_X(\phi_X, \dot{\phi}_X)$, which incorporates this fact into the energy calculations, and constructs a measure of how much energy is present in the robot that is going into crossing. For $X = L, R$, we define

$$CE_X(\phi_X, \dot{\phi}_X) = \frac{1}{C^2}\dot{\phi}_X^2 \operatorname{sgn}(\dot{\phi}_X) + 2(\cos\phi_X - 1)\operatorname{sgn}(\phi_X). \tag{11.16}$$

CE_X is a $\mathcal{C}^1$ function of ϕ_X and $\dot{\phi}_X$, is zero for lateral tilt phase trajectories that come to rest exactly on the verge of crossing, and is more positive the greater the severity of crossing.

In each execution cycle of the tilt phase controller, the crossing energies CE_L and CE_R are evaluated and individually passed through a one-sided smooth deadband function (see upper blue section in Figure 11.2). The result is scaled to give an initial measure of how much swing out is required in the left and right lateral directions. The one-sided deadband ensures that the swing out is zero below a minimum crossing energy of CE_{min}, and that it smoothly transitions to a linear relationship beyond that. A pair of hold filters (Allgeuer, 2020) is applied to ensure that the greatest activation over the most recent time is kept and used for each side. The filtered lateral swing out values are then linearly interpolated based on the expected support conditions of the robot (a function of the gait phase μ_i). At this point, a sagittal swing out component is added that ensures that the resultant swing out is, within limits, in the direction of $\bar{P}_d$ (see Section 11.2.3). The final resulting swing out tilt P_o is then elliptically soft-coerced to ensure that the swing out stays within reasonable limits.

The tuning of the swing out feedback mechanism is done by examining real crossing trajectories of the robot. The p_{xL} and p_{xR} values are read from the average points of inflection of the observed curves, and C is chosen to give the most constant observed profiles of E_X possible. A suitable value for CE_{min} can be calculated by choosing a value of ϕ_X that is just less than zero, and calculating the crossing energy that it would correspond to if the robot were at rest at that tilt.

11.2.7 Swing Ground Plane

While the ground nominally coincides with the N plane during walking, if disturbances are present this is no longer the case. This can cause premature or belated foot strike during leg swing, which is both destabilising and prevents the robot from taking the intended step sizes. The swing ground plane S (see Section 10.2.2.3) defines the plane

that is used to adjust the stepping trajectories to avoid such issues. This is different to most implementations of virtual slope walking, e.g. Missura (2015), in that it does not just linearly slant the foot motion profile—it analytically computes a smooth trajectory that respects the S plane at foot strike, yet intentionally presses into or eases off the ground immediately after, so as to apply a restoring moment to the robot. Standard virtual slope implementations can often actually *decrease* walking robustness in non-extreme situations, as the more the robot leans forwards for instance, the higher the feet are lifted at the front, and thus the less resistance there is to falling further forwards.

The desired orientation of the S plane is first computed by finding a pure tilt rotation P_{NS} relative to N that makes the N plane coincident with where the N plane would be if the robot had its expected orientation P_E. Using the same notation as in Section 11.2.2, the required tilt rotation is mathematically given as

$$P_{NS} = -P_q(q_{SN}), \tag{11.17}$$

where $P_q(\cdot)$ is a function that returns the 2D tilt phase representation of a quaternion and

$$q_{SN} = q_y(p_{yN})\, q_E^*\, q_B\, q_y(-p_{yN}). \tag{11.18}$$

Note that if the robot has its expected orientation, q_B equals q_E and $P_{NS} = (0, 0)$, i.e. the identity 2D tilt phase rotation, so S ≡ N. To reduce noise and prevent swing ground plane adjustments from happening when walking is near nominal, a mean filter followed by elliptical smooth deadband is applied to P_{NS} (see red section in Figure 11.2). A nominally unit gain is then applied to allow the strength of the S plane feedback to be tuned if this helps with passive stability. The resulting tilt phase is then passed through elliptical soft coercion to ensure that it always stays within limits. This yields the final required activation P_S of the swing plane corrective action (see Section 10.2.2.1).

11.2.8 Maximum Hip Height

As a result of repeated disturbances or self-disturbances, it can occur that the robot enters a semi-persistent limit cycle of sagittal oscillations. In such situations, limiting the height of the hips above the feet can help lower the CoM, and thereby increase the passive stability of the robot, as greater rotations are then required for tipping. As such, by temporarily restricting the maximum hip height of the robot, unwanted oscillations of the robot can be dissipated.

A measure I of the instability of the robot is first computed by applying a slope-limited low pass filter to normed speed values s_d of the mean-filtered deviation tilt $\bar{P}_d$, i.e. to

$$s_d = \tfrac{1}{\Delta t} \|\Delta \bar{P}_d\|. \tag{11.19}$$

Note that only *changes* in orientation contribute to I, so consistent leaning in a particular direction, for instance, does not contribute to the quantified 'instability' I of the robot. Note also that the slope-limited low pass filter is nominally chosen to have a relatively long settling time, and that $\Delta\bar{P}_d$ can optionally be masked to only include sagittal components, if desired. Given the quantified instability I, coerced interpolation is used to map this to a desired range of maximum hip heights, so that greater levels of instability correspond to smaller allowed hip heights. A final slope limiter ensures that all changes to the resulting H_{max} activation value occur continuously, and suitably gradually.

The tuning of the maximum hip height feedback mechanism essentially reduces to the choice of a settling time for the low pass filter, usually on the order of a few seconds, and the choice of an instability range to use for interpolation. The former is tuned based on how responsive the maximum hip height is desired to be, and the latter is tuned by artificially disturbing the robot and gauging as of what measured instability hip height feedback would have been suitable.

11.2.9 Timing Adjustment

Timing is an important feedback mechanism for the preservation of balance. In addition to its own stabilising effect, it also allows other corrective actions like the swing out mechanism to work most effectively. The approach to timing feedback that was used as part of the direct fused angle feedback controller (see Section 9.3.4) is also used for the tilt phase controller. The only modification is that the calculated feedback is reformulated to be in terms of the lateral deviation tilt p_{xd} instead of the fused roll deviation d_ϕ. As shown in the green section in Figure 11.2, the final calculated frequency offset f_{go}, given by Equation (9.9), is added to the nominal frequency f_{nom}, and used to drive the instantaneous gait frequency activation value f_g, as required (see Section 10.2.2.1).

11.2.10 Step Size Adjustment

Just like swing out is used to combat lateral crossing trajectories, the step size adjustment corrective action is used to combat sagittal crossing trajectories. While other corrective actions like the arm tilt and support foot tilt do indeed dampen any sagittal disturbances that are experienced, sometimes this damping is not enough to stabilise the robot, and reactive steps need to be taken as a last resort of keeping balance. Often, the fact that the robot is on a sagittal crossing trajectory can be inferred from the state (i.e. sagittal energy) of the robot long before the robot actually reaches its tipping point. This means that there is frequently enough time for the robot to take a preemptive

step that avoids significant tipping altogether. The quantification of the sagittal crossing energies of the robot, and conversion thereof to suitable step size adjustments, is performed using the so-called tripendulum model. This model is evaluated as a function of the estimated phase pitch $\hat{p}_{yB} \equiv \theta$ and phase pitch velocity $\dot{\hat{p}}_{yB} \equiv \dot{\theta}$ of the trunk, which are calculated by applying a WLBF filter to the raw phase pitch data p_{yB}, and by evaluating the resulting line of best fit at the mean of the fitted data points.[6]

When the robot is standing on one foot, the sagittal transient response of the robot has three distinct zones of behaviour. When the robot is leaning far forwards, the contact patch of the robot reduces to just the front edge of the foot, and a forwards-leaning tipping behaviour results. This is referred to as the front zone of the sagittal dynamics. Similarly, if the Centre of Pressure (CoP) moves to the back edge of the foot, a backwards-leaning tipping behaviour results that is referred to as the back zone. In between the front and back zones is a passively stable middle zone that in general returns the robot back to its nominal orientation when small disturbances are applied.

The three zones of behaviour of the sagittal dynamics of the robot are modelled using the nonlinear tripendulum model (see Figure 11.3), which can be summarised by the differential equation

$$\ddot{\theta} = \begin{cases} C_f^2 \sin(\theta - \theta_f) & \text{for } \theta \in [\theta_s, \infty), \\ -C_m^2 \sin(\theta - \theta_m) & \text{for } \theta \in (\theta_t, \theta_s), \\ C_b^2 \sin(\theta - \theta_b) & \text{for } \theta \in (-\infty, \theta_t], \end{cases} \tag{11.20}$$

where

- θ is the WLBF-filtered phase pitch as before,
- C_f, C_m and C_b are three separate pendulum constants,
- θ_f, θ_m and θ_b are tuned constants that represent the centres (i.e. the equilibrium points) of the three pendulum zones, and,
- θ_s and θ_t are two constants that are calculated such that Equation (11.20) is continuous.

The phase pitches θ_s and θ_t mark the two transition points between the three pendulum zones, and are calculated using

$$\theta_s = \bar{\theta}_{fm} + \operatorname{atan2}\big((C_f^2 - C_m^2)\sin\Delta\theta_{fm},\ (C_f^2 + C_m^2)\cos\Delta\theta_{fm}\big), \tag{11.21a}$$

$$\theta_t = \bar{\theta}_{bm} + \operatorname{atan2}\big((C_b^2 - C_m^2)\sin\Delta\theta_{bm},\ (C_b^2 + C_m^2)\cos\Delta\theta_{bm}\big), \tag{11.21b}$$

where

$$\bar{\theta}_{fm} = \tfrac{1}{2}(\theta_f + \theta_m), \qquad \bar{\theta}_{bm} = \tfrac{1}{2}(\theta_b + \theta_m), \tag{11.22a}$$

$$\Delta\theta_{fm} = \tfrac{1}{2}(\theta_f - \theta_m), \qquad \Delta\theta_{bm} = \tfrac{1}{2}(\theta_b - \theta_m). \tag{11.22b}$$

6 Recall that this is analogous to the way it was done for the swing out corrective action in Section 11.2.6.

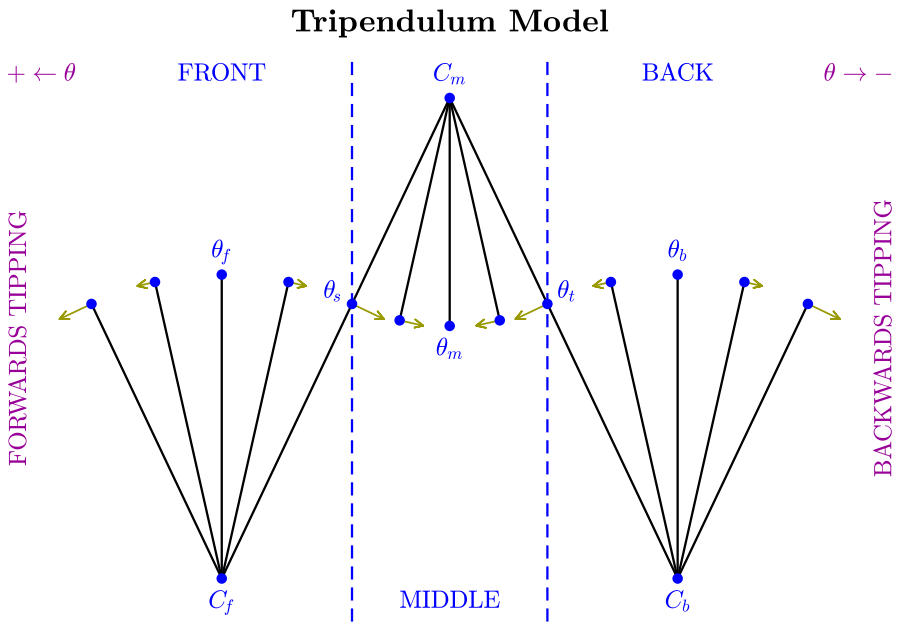


Figure 11.3: An illustration of the tripendulum model, and how it divides up the behaviour of the phase pitch angle θ of the torso into three distinct zones—the front, middle and back pendulums. Note that these pendulums are figurative—not literal—and are only seen as a mathematical description of how the phase pitch angle behaves for undisturbed trajectories of the robot. The centre angles θ_f, θ_m and θ_b of the pendulums (i.e. the equilibrium points) are shown, as well as the corresponding pendulum constants C_f, C_m and C_b, and the transition angles θ_s and θ_t. The dark yellow arrows indicate the direction and strength of the angular acceleration $\ddot{\theta}$, as calculated by Equation (11.20).

As was the case in Section 11.2.6 for the standard nonlinear pendulum model, instead of using Equation (11.20) to make temporal predictions of the behaviour of the robot and using this as a basis for feedback, we begin by constructing a notion of orbital energy, and use this to define a notion of crossing energy for the forwards and backwards directions. Within the front, middle and back pendulum zones, the following three respective expressions of orbital energy remain constant:

$$E_f(\theta, \dot{\theta}) = \dot{\theta}^2 - 2C_f^2\big(1 - \cos(\theta - \theta_f)\big), \tag{11.23a}$$

$$E_m(\theta, \dot{\theta}) = \dot{\theta}^2 + 2C_m^2\big(1 - \cos(\theta - \theta_m)\big), \tag{11.23b}$$

$$E_b(\theta, \dot{\theta}) = \dot{\theta}^2 - 2C_b^2\big(1 - \cos(\theta - \theta_b)\big). \tag{11.23c}$$

By identifying the kinetic and potential energy components of these orbital energy expressions, and accounting for whether they help or hinder sagittal crossing, the following expressions for the so-called

forwards and backwards crossing energies can be developed, similar to as was done in Section 11.2.6:

$$CE_f(\theta, \dot{\theta}) = \dot{\theta}^2 \operatorname{sgn}(\dot{\theta}) - PE_f(\theta), \tag{11.24a}$$
$$CE_b(\theta, \dot{\theta}) = -\dot{\theta}^2 \operatorname{sgn}(\dot{\theta}) - PE_b(\theta), \tag{11.24b}$$

where $PE_f(\theta)$ and $PE_b(\theta)$ are expressions of 'required potential energy to crossing' that are given by

$$PE_f(\theta) = \begin{cases} -2C_f^2(1 - \cos(\theta - \theta_f)) & \text{if } \theta \in [\theta_f, \infty), \\ 2C_f^2(1 - \cos(\theta - \theta_f)) & \text{if } \theta \in [\theta_s, \theta_f), \\ 2C_f^2(1 - c_{sf}) + 2C_m^2(\cos(\theta - \theta_m) - c_{sm}) & \text{if } \theta \in [\theta_t, \theta_s), \\ 2C_f^2(1 - c_{sf}) + 2C_m^2(c_{tm} - c_{sm}) & \\ \qquad +2C_b^2(c_{tb} - \cos(\theta - \theta_b)) & \text{if } \theta \in (-\infty, \theta_t), \end{cases} \tag{11.25a}$$

$$PE_b(\theta) = \begin{cases} -2C_b^2(1 - \cos(\theta - \theta_b)) & \text{if } \theta \in (-\infty, \theta_b], \\ 2C_b^2(1 - \cos(\theta - \theta_b)) & \text{if } \theta \in (\theta_b, \theta_t], \\ 2C_b^2(1 - c_{tb}) + 2C_m^2(\cos(\theta - \theta_m) - c_{tm}) & \text{if } \theta \in (\theta_t, \theta_s], \\ 2C_b^2(1 - c_{tb}) + 2C_m^2(c_{sm} - c_{tm}) & \\ \qquad +2C_f^2(c_{sf} - \cos(\theta - \theta_f)) & \text{if } \theta \in (\theta_s, \infty), \end{cases} \tag{11.25b}$$

where we use the shorthand $c_{xy} \equiv \cos(\theta_x - \theta_y)$, for $x, y = f, s, m, t, b$. Note that both $PE_f(\theta)$ and $PE_b(\theta)$, and thus $CE_f(\theta, \dot{\theta})$ and $CE_b(\theta, \dot{\theta})$, are $\mathcal{C}^1$ functions of θ due to the way that θ_s and θ_t are calculated in Equation (11.21). The value of the crossing energy $CE_f(\theta, \dot{\theta})$ can be interpreted as the amount of energy that the robot has in the direction of forwards falling ($\theta \to \infty$), while analogously, the value of $CE_b(\theta, \dot{\theta})$ can be interpreted as the amount of energy that the robot has in the direction of backwards falling ($\theta \to -\infty$). Both crossing energy values are respectively zero for trajectories that come to rest exactly on the verge of crossing, i.e. exactly at the front or back pendulum equilibrium points.

In each execution cycle of the tilt phase controller, the front and back crossing energies are calculated and individually passed through a one-sided smooth deadband function (see lower blue section in Figure 11.2). The result is scaled to give a pair of appropriate GCV adjustment values in the sagittal direction. These values are passed through hold filters (Allgeuer, 2020) to ensure a more temporally consistent activation of the step size adjustments, after which they are combined through basic addition. Final soft coercion is applied to the resulting combined sagittal GCV to ensure that the final adjustment stays within reasonable ranges. The output of the soft coercion is added to the x-component of the desired end-of-step GCV $\mathbf{v}_{EOS}$, which

is then later used to calculate $\mathbf{v}_i$ as described in Section 11.1.1. It is a rare occurrence that both non-zero forwards tipping and non-zero backwards tipping GCV adjustments are produced, but if this happens, the calculated adjustments are in opposite directions, so the process of adding them effectively selects an intermediate adjustment value that balances the strengths of the desired adjustments in each direction.

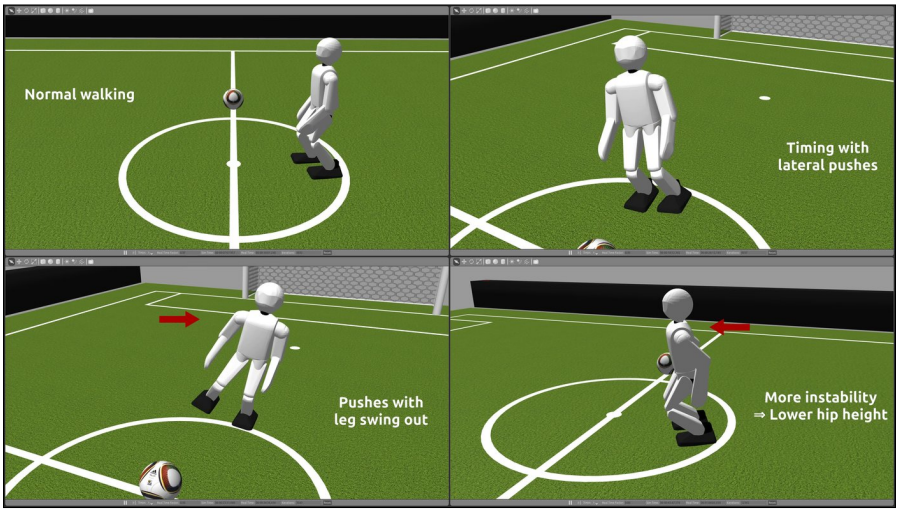
Tuning of the implemented step size adjustment scheme essentially amounts to tuning of the tripendulum model. While the values of θ_f and θ_b are tuned by examining the points of inflection of real sagittal crossing trajectories of the robot, the value of θ_m is generally kept constant at the nominal phase pitch p_{yN} of the robot (see page 173). The three pendulum constants C_f, C_m and C_b are chosen so that the orbital energies calculated in Equation (11.23) remain as constant as possible within the three pendulum zones. The crossing energy thresholds beyond which step size adjustments are invoked (a parameter of the one-sided deadband function) are tuned by choosing a value of θ that is close to the respective tipping point, and calculating the crossing energy that a robot at rest at that tilt would have.

One important property of the presented approach to step size adjustments is that adjustments are only made as a last resort if they are really needed. The calculated crossing energies will always be significantly negative in normal undisturbed walking situations, leading to zero being emitted by the one-sided smooth deadband functions. Only if significant disturbances are present that risk having the robot tip over sagittally does the deadband function emit non-zero values, and cause reactive steps to be taken.

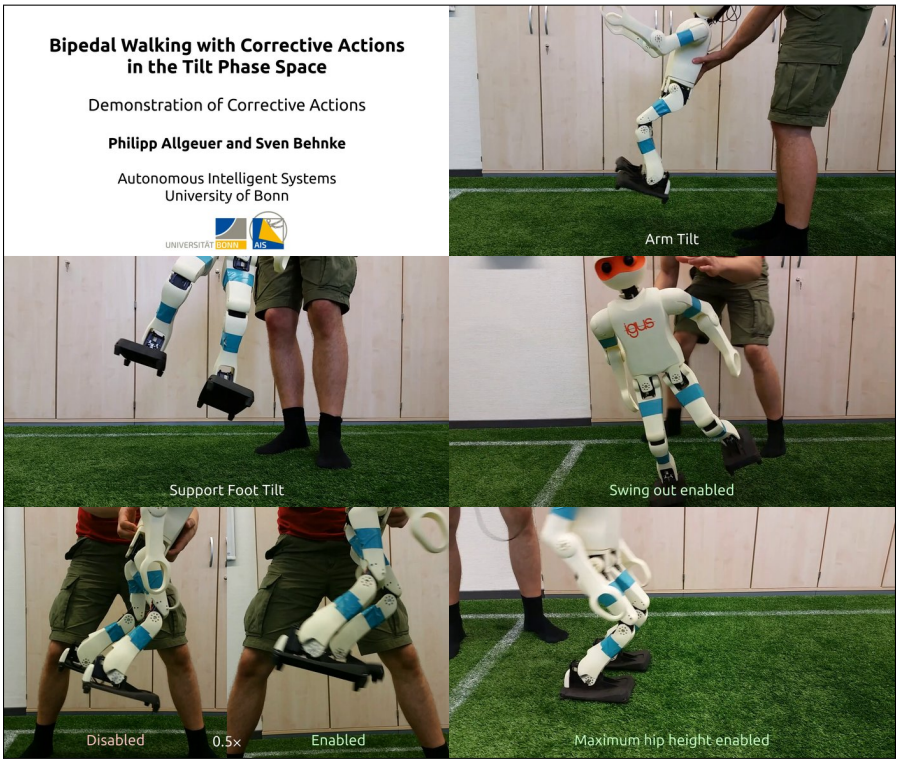
11.3 EXPERIMENTAL RESULTS

The proposed feedback controller has been implemented in C++ in the open-source igus Humanoid Open Platform ROS software (Team NimbRo, 2018), which also supports the NimbRo-OP2 and NimbRo-OP2X robots. The entire controller takes just 2.1 µs to execute on a single 3.5 GHz core. As such, it is expected that the implementation of this method at 100 Hz on even a modest microcontroller would be possible. Such portability is of great advantage in the area of low-cost robotics. Also, given the relative complexity of the gait and the diverse range of corrective actions, the number of important configuration constants has been kept rather low. The constants are in all cases expressed in a way that they are dimensionless, easy to understand and tune, and more than often just the default values can be used due to these two factors.

A short demonstration of some of the main corrective actions that set the tilt phase controller apart from other controllers is shown in Video 11.1. A more detailed demonstration and analysis of all of the implemented corrective actions is shown in Video 11.2, with step size



Video 11.1: A short demonstration of bipedal walking in physical simulation using the KGG and tilt phase controller. Not all corrective actions are shown. Note that due to limitations of the version of Gazebo used, uncontrollable fluctuations in the simulation log playback speed resulted in noticeable speed fluctuations in the video.
https://youtu.be/ub0GvZ7AbLc
Short Demonstration of the Action of the Tilt Phase Controller



Video 11.2: Individual demonstrations of the effects of the various implemented KGG corrective actions, as activated by the tilt phase controller. Plots of the experiments are provided in Figures 11.4 to 11.6.
https://youtu.be/spFqqktZ1s4
Demonstration of corrective actions: Bipedal walking with corrective actions in the tilt phase space

Video 11.3: Demonstration of step size adaptation using the tilt phase controller and tripendulum model. The NimbRo-OP2X is pushed repetitively in the sagittal direction and compared in performance to the same gait without step size adaptation. A plot corresponding to the second-last backwards push of the robot in the video is provided in Figure 11.7.
https://youtu.be/R9gThzV1hTQ
Step Size Adaptation Using the Tripendulum Model

adaptation being handled separately in Video 11.3. The improvement of the robustness of the gait when the tilt phase controller is enabled is evident across all videos. In particular, in Video 11.3 one can see that the closed-loop gait with step size adjustments enabled requires significantly stronger pushes to make the robot fall than the open-loop gait.[7] Importantly, while the robot reacts quickly with large steps to strong pushes, for small pushes no discernible change in step size occurs. This ensures that the activation of the step size adjustment remains a 'last resort' to combating disturbances, and does not cause a change in walking behaviour for regular lightly-disturbed walking. The size of the reactive steps taken in Video 11.3 was limited to an internal sagittal GCV of 1.2. More severe reactive steps could have been allowed by increasing this limit, but it is expected that at some point with the increased resulting volatility of the robot, the overall reliability of the gait would degrade and risk damaging the robot. Significant unnecessary self-destabilisations would also be an issue (even if they do not ultimately lead to a fall), e.g. like frequently observed for the capture step controller.

The individual corrective action experiments shown in Video 11.2 have been plotted in detail in Figures 11.4 to 11.6. The experiments were performed on a real igus Humanoid Open Platform, and were

7 In order to isolate the effect of the step size adaptation, the swing ground plane corrective action was not enabled in the video. It is expected that in particular forwards balance recovery would improve with this additional correction enabled.

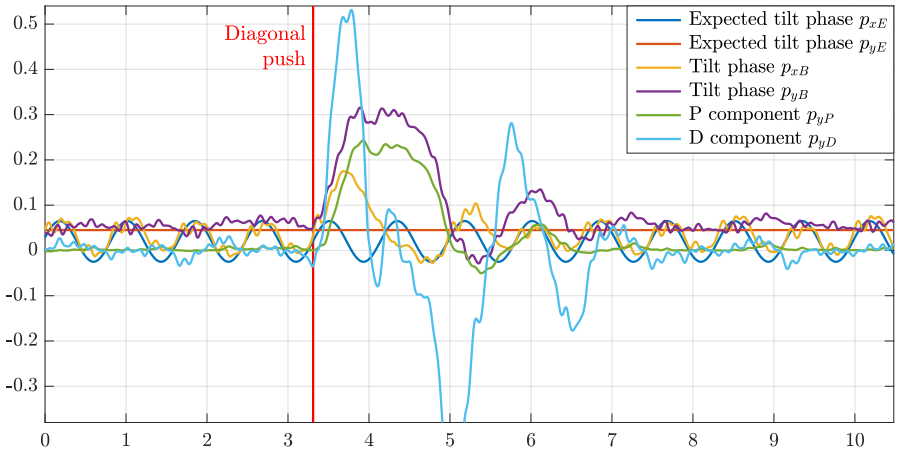
specifically designed to isolate and contrast the walking performance of the robot with and without the effect of the various individual corrective actions. For almost all of the experiments, all corrective actions except for the one in focus were turned off to better illustrate the true effect of the feedback.

In Figure 11.4a, it can be observed that the tilt phase corresponds closely to the expected waveforms until a large diagonal push disturbs the robot. The PD activations quickly spike, preventing a forwards fall, and aiding the robot in returning to its expected tilt phase trajectory. Note that the plotted P and D components correspond to the outputs of the elliptical smooth deadband blocks (see Figure 11.2), prior to the application of the individual elliptical arm and foot gains. It can be seen that when the robot starts returning to upright, the sign of the derivative feedback changes quickly to dampen the system and prevent excessive overshoot.

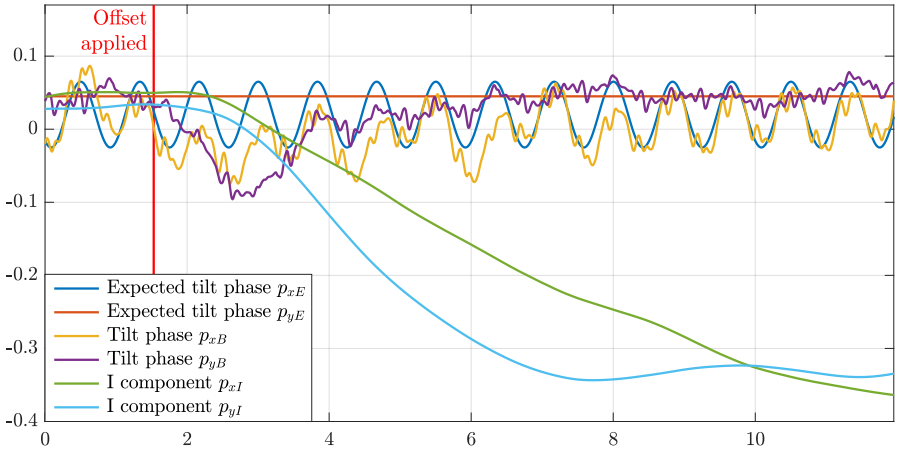
In Figure 11.4b, the robot begins by walking on the spot. At time $t = 1.5\,\text{s}$, a sudden unknown software offset (external to the gait module) is applied to both ankles of the robot, causing the robot to pitch and roll away from its nominal expected tilt phase waveforms. In response, the I components p_{xI} and p_{yI} quickly integrate up the observed 3D deviations in orientation and cause the robot to learn 2D hip shifts and continuous foot tilts that perfectly negate the applied software offsets. Note that the plotted I components correspond to the output of the mean filter in Figure 11.2, i.e. the one that is used right before the hip shift and continuous foot gains in the integral feedback pathway. At approximately 10° of resultant rotation, and despite the relative severity of the applied offsets, the robot returns to completely upright nominal walking within 6.6 s.

In Figure 11.4c, the robot is made to start walking forwards, and then slow down and stop again shortly after. Without the feed-forward effects of the leaning corrective action, the robot falls backwards after taking a few forwards steps. With leaning enabled, the robot tilts slightly in the direction it is accelerating, both at the start and end of walking, and thereby mitigates a fall. In the plot, the magnitude of the sagittal leaning component p_{yl} has been scaled up by a factor of 10 to make it more visible in relation to the other plotted quantities. The plotted sagittal GCV acceleration is calculated by numerically differentiating the sagittal GCV v_{gx} in a smooth manner using a WLBF filter followed by a slope limiter (see Figure 11.2).

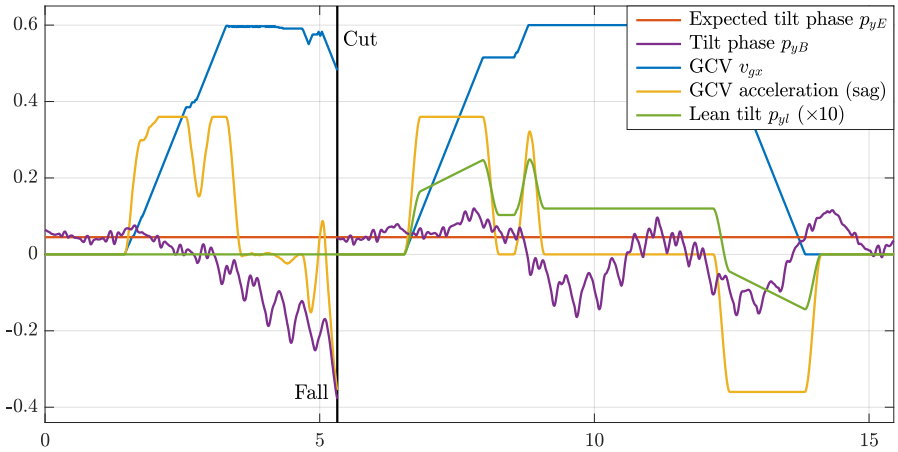
In Figure 11.5a, a large lateral push is applied to the robot, putting it on a crossing trajectory that would normally result in a lateral fall over the outside of the support foot. The crossing energies CE_L and CE_R, calculated using Equation (11.16), are well below zero prior to the push, but as the excess lateral energy is detected, the right foot crossing energy CE_R quickly exceeds $CE_{min} = -0.044$ and causes swing out of the left leg in the lateral direction. The counterbalancing



(a) The effect of the arm tilt and support foot tilt PD feedback in recovering balance after a diagonal push with significant destabilising power.

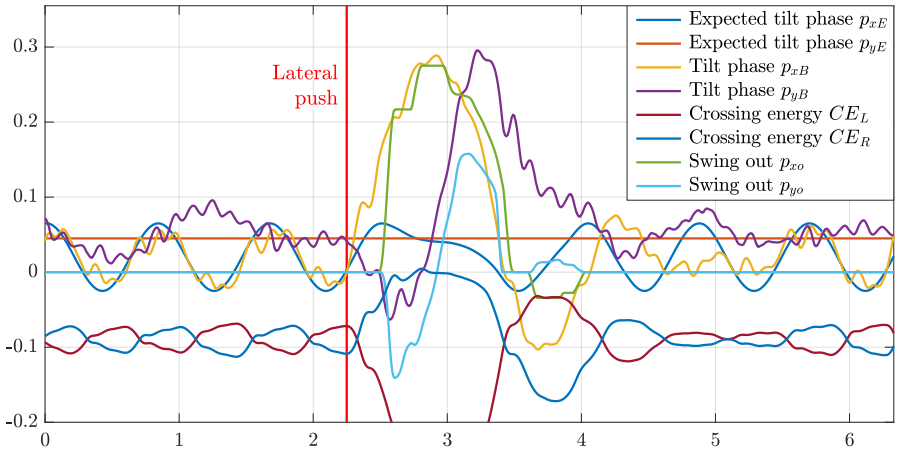


(b) The effect of the hip shift and continuous foot tilt integral feedback after a sudden unknown software offset is applied to the ankles of the robot.

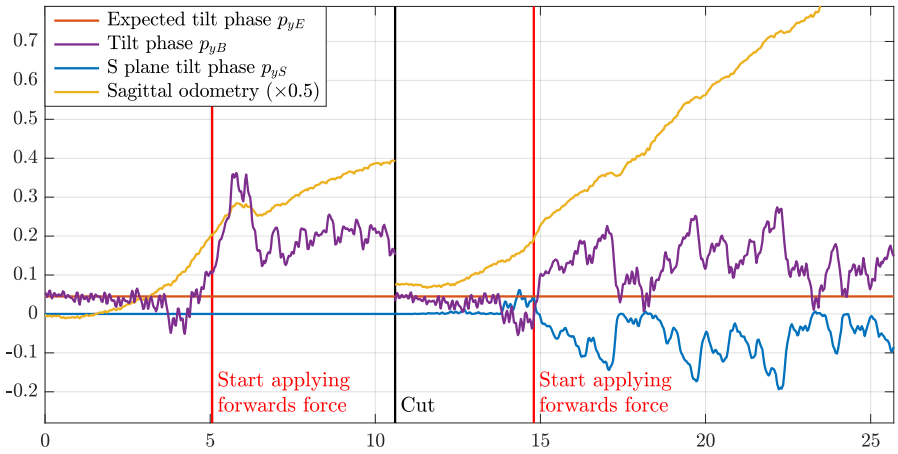


(c) The effect of leaning on sagittal walking. Without leaning (before cut), the forwards GCV acceleration makes the robot tip over backwards. With leaning (after cut), there is no fall.

Figure 11.4: Plots of the PD, I and leaning corrective actions on a real robot. Refer to Video 11.2 for the corresponding footage.



(a) The effect of the swing out tilt in recovering from a lateral push that would otherwise have led to a fall. The full 2D swing out allows the robot to remain balanced sagittally as well in the 1.2 s it takes to return laterally.

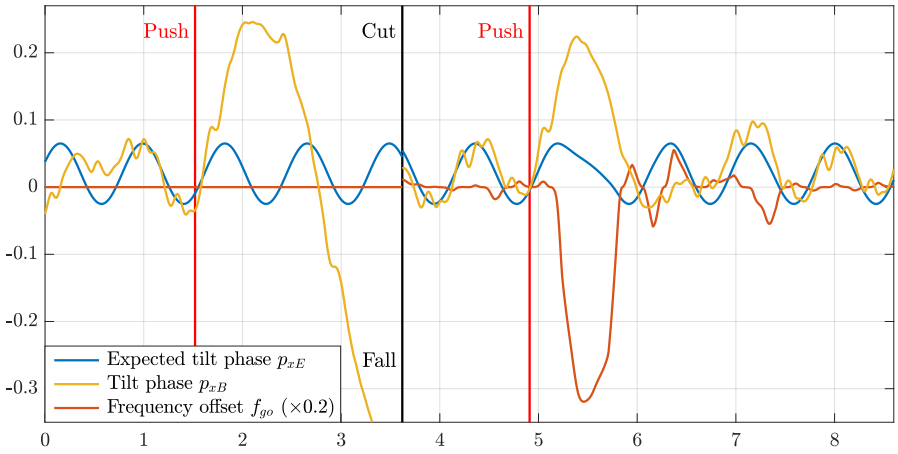


(b) Effect of the swing ground plane in reducing premature foot strike, demonstrated by applying a continuous force to the robot that induces sagittal tilt. Before cut: S plane disabled, After cut: S plane enabled.

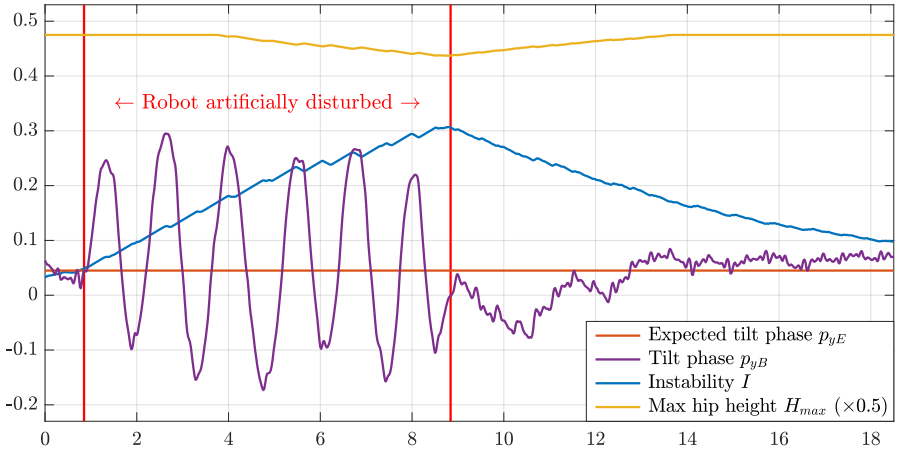
Figure 11.5: Plots of the swing out and swing ground plane corrective actions on a real robot. Refer to Video 11.2 for the corresponding footage.

effect of the left leg dissipates some of the crossing energy, and makes sure CE_R never significantly surpasses zero, which would indicate a predicted non-returning trajectory. To maintain sagittal balance while the robot waits to return from its lateral trajectory, the sagittal swing out component p_{yo} is used, first in one direction, and then the other.

In Figure 11.5b, a continuous forwards force is applied to the robot during forwards walking, causing the robot to tilt forwards. Without swing ground plane adjustment, the robot tends to walk 'into the ground', and gets more stuck than if the swing ground plane is enabled. This is evidenced by the factor-of-two difference in sagittal gait odometry, also plotted. The S plane tilt phase p_{yS} is opposite in sign to the torso orientation p_{yB} as, for example, leaning forwards



(a) The effect of timing feedback in avoiding self-disturbances that can lead to a fall. Before cut: Timing disabled, After cut: Timing enabled.



(b) Oscillations in the robot orientation, induced artificially here, cause the calculated instability to increase and limit the allowed hip height.

Figure 11.6: Plots of the timing and maximum hip height corrective actions on a real robot. Refer to Video 11.2 for the corresponding footage.

corresponds to a backwards tilt of the ground plane relative to the robot.

In Figure 11.6a, lateral pushes are applied to the robot to disrupt its natural walking rhythm. Without timing adjustments, this leads to a lateral fall, as can be seen in the left half of the plot. With timing adjustments however, the robot slows down its stepping motion as soon as it detects the disturbance, and tries to place its next foot on the ground exactly when the lateral tilt is close to zero again. Importantly, during normal stable walking only negligible contributions of the timing corrective action to the gait frequency can be observed.

In Figure 11.6b, the robot is artificially disturbed over multiple seconds to demonstrate an extreme case of how oscillations lead to a higher quantified level of instability, and subsequent reduction in hip height. During real walking, this is relevant for situations where

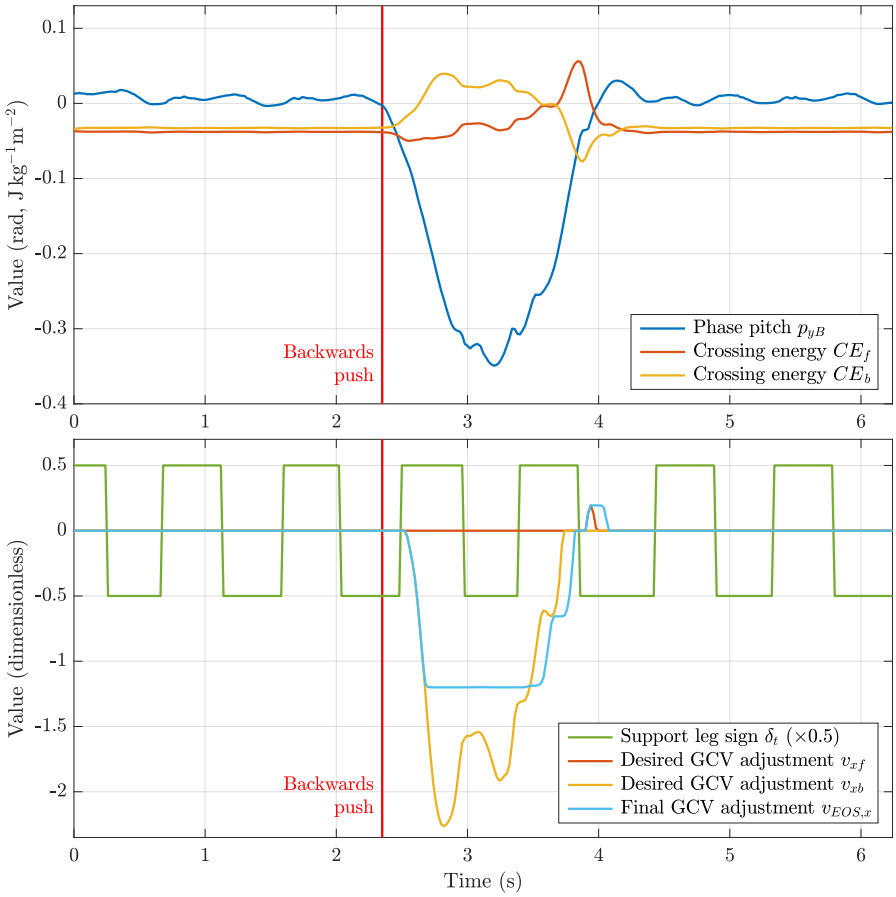


Figure 11.7: Plots corresponding to the second-last backwards push of the NimbRo-OP2X in Video 11.3, demonstrating the effect of step size adjustment in maintaining the balance of the robot. The crossing energies are plotted in units of $\mathrm{J\,kg^{-1}m^{-2}} \equiv \mathrm{s^{-2}}$, as this normalises the values for each robot. This is because the crossing energies then effectively become energy (J) per moment of inertia ($\mathrm{kg\,m^2}$).

the robot gets stuck in a limit cycle of oscillations. This can (and does) occur, but is difficult to replicate intentionally. Lowering the hip height increases passive stability and changes the natural frequency of the dynamics of the tilting motions. Both factors generally lead to damping of the oscillations, as can be seen in the plot.

The second-last backwards push in Video 11.3, demonstrating the action of the step size adjustment scheme, has been plotted in detail in Figure 11.7. In the top plot, the instant is shown where the push was applied, causing a significant drop in the phase pitch of the robot. The calculated crossing energies up until that point were decisively negative, but as the robot accelerated as a result of the backwards push, the backwards crossing energy CE_b quickly spiked well above zero, meaning that the robot would have had no chance of avoiding tipping over if it had not been for the reactive steps. The activation of the reactive steps, in the form of an adjustment to the sagittal GCV, is

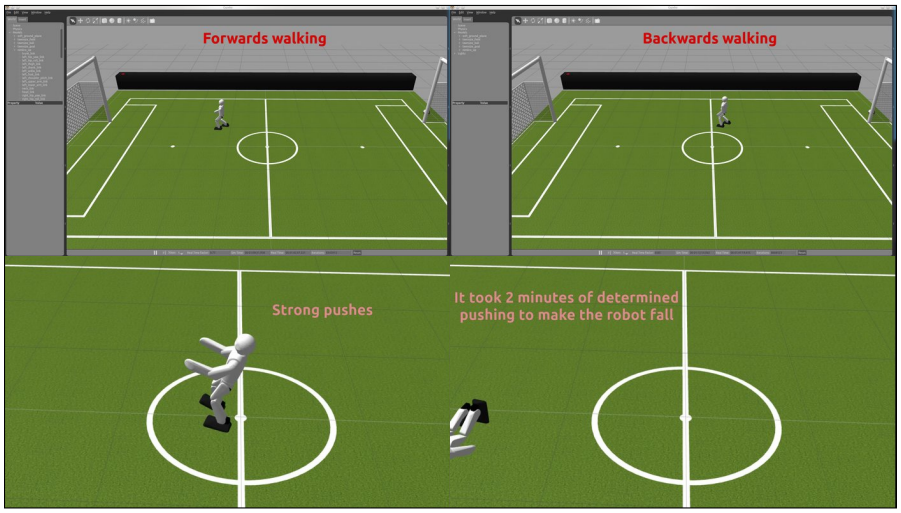
shown in the bottom plot of Figure 11.7. The current support foot at each instant is also plotted in the form of the support leg sign δ_t.[8]

As the robot returned to upright, it had considerable forwards angular momentum, as evidenced by the large rate of change of phase pitch at the time. In order to avoid overshooting and potentially falling forwards, it can be seen that the forwards crossing energy CE_f correctly predicted around time $t = 3.6$–$3.9\,\text{s}$ that this would be an issue, and caused a small forwards step to be taken soon after, as can be seen in the bottom plot. The result is that the robot had very little overshoot in the forwards direction, and was able to return to normal balanced walking very soon after the push. Note that at $t = 3.6\,\text{s}$ the robot was still leaning backwards, far away from upright, yet it was already aware at this time that the greater issue was falling forwards, not falling backwards.

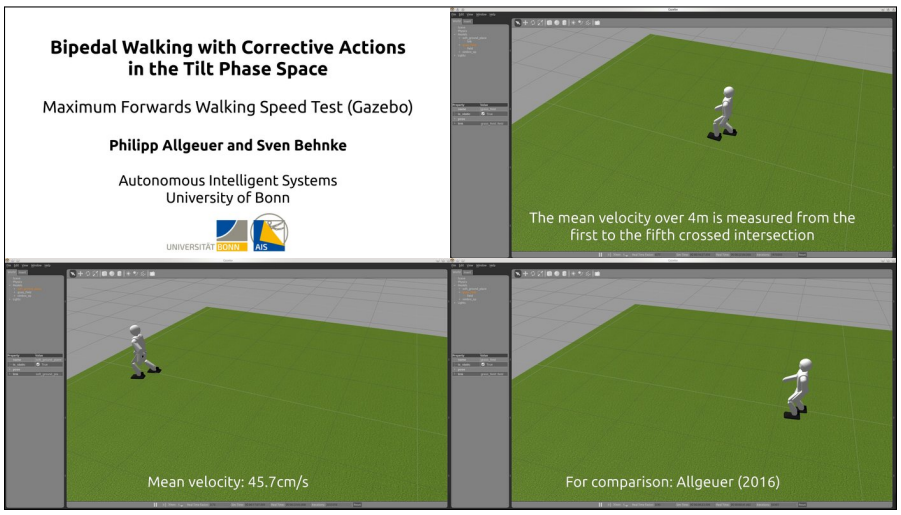
All together, the corrective actions and feedback pipelines implemented by the tilt phase controller make a significant improvement to walking robustness, both on real hardware (see previous experiments), and in simulation. The dimensionless construction of the feedback pipeline also allows the controller to be easily tuned and ported between various robot platforms without significant changes. The parameters of the feedback pipeline are generally relatively insensitive to their value, so while tuning the capture step controller for example often requires careful tuning methods and 3–4 digits of precision in order to be successful, for the tilt phase controller it is quite the opposite. Video 11.4 shows the results of the very first walking tests in simulation, after most of the configuration parameters had only been 'guessed', and no previous tests had been done. The controller performs remarkably well, and it takes two minutes of determined pushing to ultimately make the robot fall.

The tilt phase controller has also been evaluated quantitatively, as opposed to qualitatively, in Gazebo simulation. Maximum forwards walking speed tests were performed over a 4 m distance with a 1 m run-up. A maximum mean velocity of 45.7 cm/s was achieved, while with the direct fused angle feedback controller (see Chapter 9), 30.5 cm/s was achieved. A video of the associated walking speed tests is provided in Video 11.5. While both gaits provide a good walking performance, it can be seen from the video how the 'smoother' nature of the KGG allows it to achieve higher overall walking speeds than the CPG before self-disturbances become a limiting factor. The minimisation of self-disturbances for the purpose of maximising walking speed was in fact one of the principal aims of the KGG (see Section 10.1.3), and exactly this goal was distilled into a detailed list of desired properties of the KGG that influenced its design at every stage (see page 171).

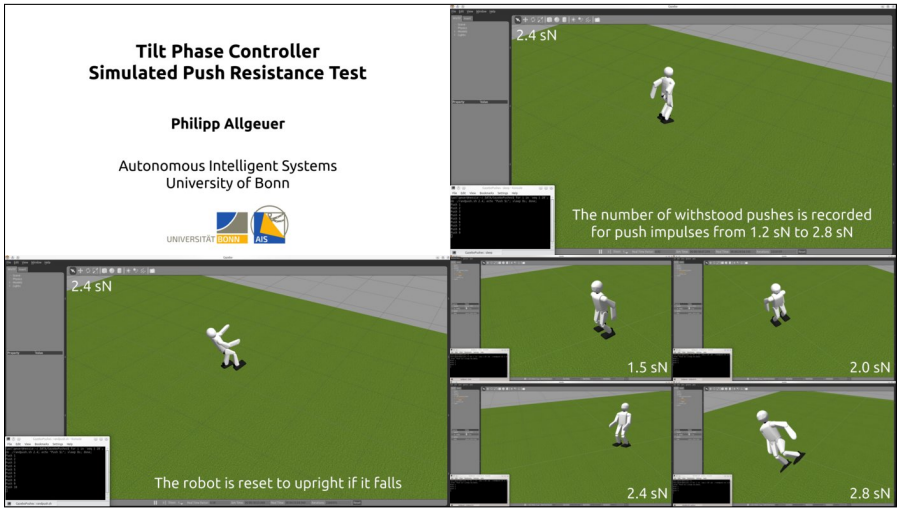
8 The support leg sign is the limb sign of the current support leg. $\delta_t = 1$ corresponds to the left leg, and $\delta_t = -1$ corresponds to the right leg.



Video 11.4: Summary video of the first walking and balance tests that were performed (in simulation) to evaluate the effectiveness of the tilt phase controller. The simulated igus Humanoid Open Platform withstood two minutes of strong pushes before eventually falling.
https://youtu.be/A_HQQfCRhDE
First Walking and Balance Test of the Tilt Phase Controller



Video 11.5: Maximum forwards walking speed test using the Keypoint Gait Generator and tilt phase controller.
https://youtu.be/yY0kpUZpjO4
Maximum forwards walking speed test: Bipedal walking with corrective actions in the tilt phase space



Video 11.6: Push resistance test performed on an igus Humanoid Open Platform in simulation. The robot is walking on the spot with the KGG gait, and is disturbed in random directions by impulses of various magnitudes. The effectiveness of the tilt phase controller with and without step size adaptation enabled is evaluated.
https://youtu.be/0SvHgVIYquc
Tilt Phase Controller Simulated Push Resistance Test

The tilt phase controller has also been put to the test in the context of push recovery. As shown in Video 11.6, sets of 20 pushes at a time were applied to an igus Humanoid Open Platform walking on the spot in simulation. Each set of 20 pushes maintained the same impulse strength, in the range from 1.2–2.8 s N, but the pushes were applied in random directions. The aim of the test was for the robot to withstand as many of the 20 pushes as possible. Table 11.1 compares the push recovery performances of the direct fused angle feedback controller, tilt phase controller without step size adaptation, and tilt phase controller with step size adaptation. This data is plotted in Figure 11.8, alongside the corresponding curve for the open-loop CPG gait. It is clear that all controllers provide a great margin of stability beyond the passive stability of open-loop walking. It can be observed however, that the tilt phase controller without step size adjustment (in order to make it a fair comparison) outperforms the direct fused angle controller evenly across all push impulses, with the number of withstood pushes at 2.6 s N, for example, being 7 compared to 3. With step size adaptation enabled, the tilt phase controller makes even further improvements over the direct fused angle controller, with the number of withstood pushes at 2.6 s N, for example, increasing to 12. Note that for reference of scale, a push impulse of 2.8 s N is expected to cause an instantaneous change in CoM velocity of about 42–56 cm/s for the igus Humanoid Open Platform, which is very severe given its CoM height of only 55 cm.

Table 11.1: Number of withstood simulated pushes (out of 20) for various controllers and controller configurations

Impulse (s N)	**1.2**	**1.5**	**1.8**	**2.0**	**2.2**	**2.4**	**2.6**	**2.8**
Direct fused angle	19	19	15	12	11	8	3	3
Tilt phase (no steps)	20	20	17	16	14	9	7	4
Tilt phase (full)	20	20	18	16	15	15	12	10

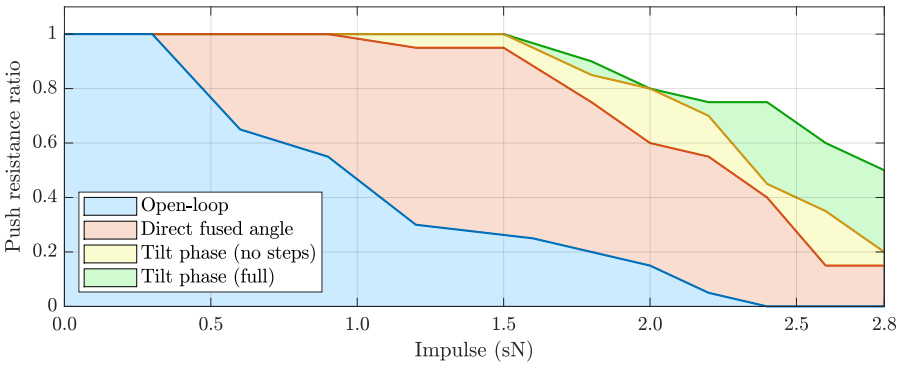


Figure 11.8: Plot of the ratio of withstood pushes against push impulse magnitude for a simulated igus Humanoid Open Platform walking on the spot with the direct fused angle feedback controller, tilt phase controller without step size adaptation enabled, and full tilt phase controller. A push resistance curve for the open-loop Central Pattern Generator is provided as a reference. The raw data corresponding to the plot is given in Tables 9.1 and 11.1.

Although the step size adaptation makes slight improvements for moderate pushes, the true power of the reactive steps is seen for strong pushes, where it makes a decisive difference in the ability of the robot to capture its imbalances. Importantly however, this added stability does not come at the cost of interfering with the reaction of the robot to milder pushes, as it visibly does for the capture step controller.

To further quantify the positive stabilising effects of the tilt phase step size adjustment scheme, in particular in combination with the arm tilt corrective action, push experiments were performed on a real NimbRo-OP2X robot. As can be seen in the excerpts shown in Video 11.3, sagittal pushes were applied to the robot while it was walking on the spot, and the transient responses were recorded. The push recovery performance of the open-loop and tilt phase feedback gaits are compared in Figure 11.9 by means of a phase plot. The blue trajectories are the pushes and subsequent transient responses that were successfully withstood by the robot, while the red trajectories show pushes that led to a fall. It is immediately apparent that the closed-loop gait has much greater push recovery ability than the open-loop gait. The open-loop gait, for example, can only deal with phase pitch velocities from -0.5 to $0.45\,\mathrm{rad/s}$, while the closed-loop gait has

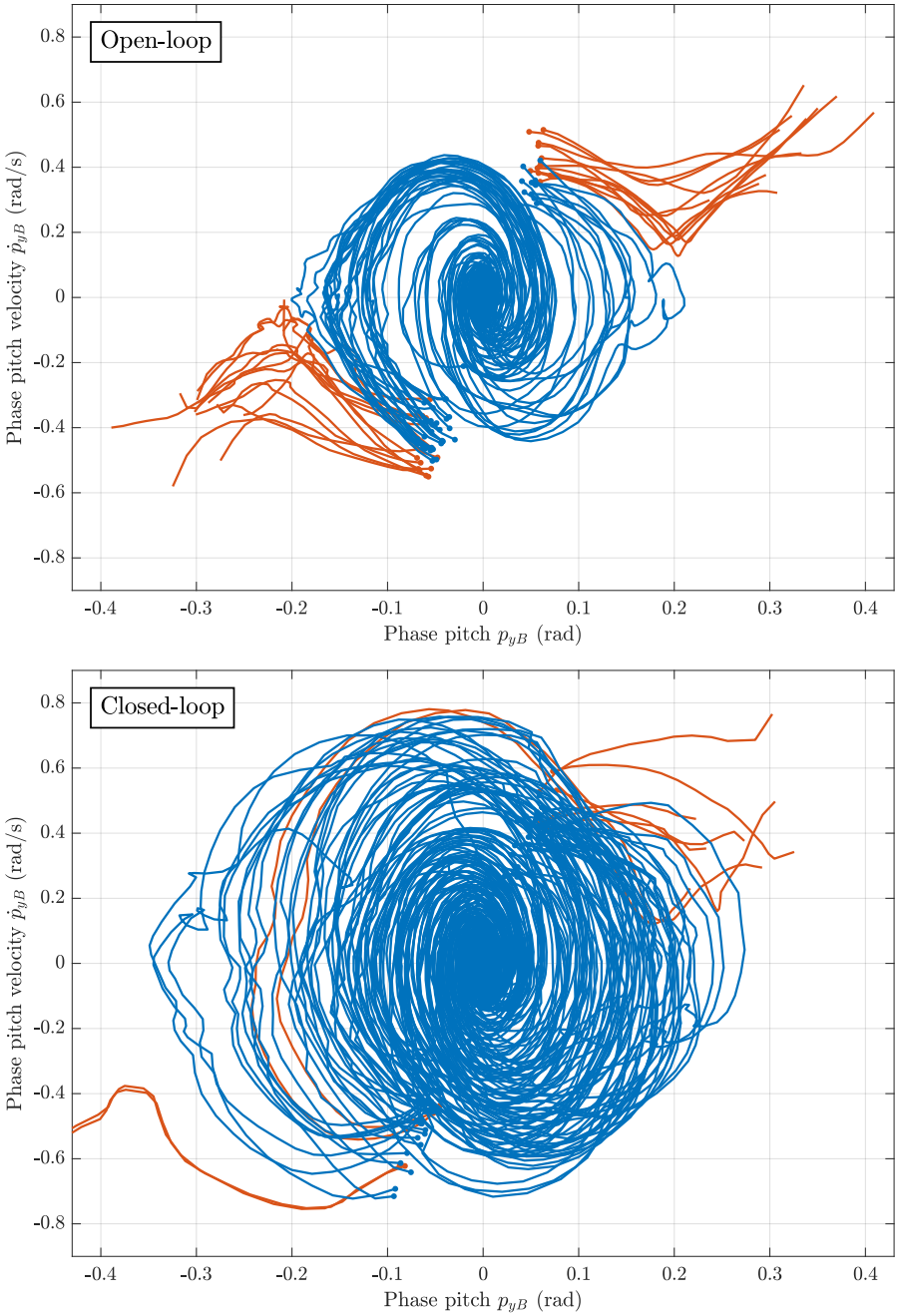


Figure 11.9: Plots of the phase response of a walking NimbRo-OP2X robot when sagittal pushes of various strengths are applied. The phase pitch velocity $\dot{p}_{yB}$ is plotted against the phase pitch p_{yB} for each individual push trajectory, and the resulting curve is coloured according to whether the robot successfully withstood the push (blue) or not (red). The beginning of each trajectory is marked with a solid dot. The top plot corresponds to the performance of the robot with the tilt phase controller disabled (open-loop), and the bottom plot shows how this performance improves with the controller enabled (closed-loop), including in particular with step size adjustments enabled. Refer to Video 11.3 for videos of a subset of the performed pushes.

a stable range closer to -0.72 to $0.75\,\mathrm{rad/s}$. The maximum values of phase pitch that occurred during stable blue trajectories also increased from -0.2 to $0.2\,\mathrm{rad}$ for open-loop, to -0.35 to $0.27\,\mathrm{rad}$ for closed-loop. This increase is mostly attributed to the step size adjustment scheme, as it allows otherwise irrecoverable phase pitches (beyond the sagittal tipping point of the robot) to be recovered through a change of contact point with the ground. Note that the natural tipping point of the NimbRo-OP2X is less than for the igus Humanoid Open Platform due to differences in mass distribution and foot size to height ratio.

The difference between stable and unstable trajectories is relatively clear-cut in the open-loop case, as the situation is relatively simple. In the closed-loop case however, the use of reactive steps complicates the response of the robot, and makes it more dependent on the gait phase of the robot at the time of the push. As an effect of this, the exact line of separation between falling and non-falling trajectories becomes somewhat less well-defined. In fact, for two of the failed backwards pushes, the robot actually fell forwards after recovering from the backwards direction.

The data in Figure 11.9 can alternatively be viewed as a heat map, as shown in Figure 11.10. The states in the phase space are divided into cells (i.e. bins), and the trajectories that pass through any one cell are collected and used to calculate a success rate for the cell. The success rate is a value in the range $[0, 1]$, and corresponds to the proportion of trajectories that pass through the cell that did not end in a fall. Overall, Figure 11.10 shows once again the significant improvement in stability that can be attributed to the tilt phase controller, and how it effectively leads to a robust humanoid gait.

11.4 CONCLUSION

Walking does not always require overly complex stabilisation mechanisms to achieve high levels of robustness. In this chapter, a feedback controller for robust bipedal walking has been presented that relies solely on measurements from a single 6-axis IMU, and is applicable to low-cost robots with noisy sensors, imperfect actuation and limited computing power. No highly tuned or complex physical models are required, and great portability is ensured through the use of dimensionless parameters and configuration constants. The wide variety of corrective actions that are employed cover many different aspects of balanced walking, including both short-term and long-term stability. In summary, although conceptually relatively simple, the tilt phase controller, in combination with the Keypoint Gait Generator (KGG), achieves genuinely good walking results with minimal assumptions about the robot hardware and performance.

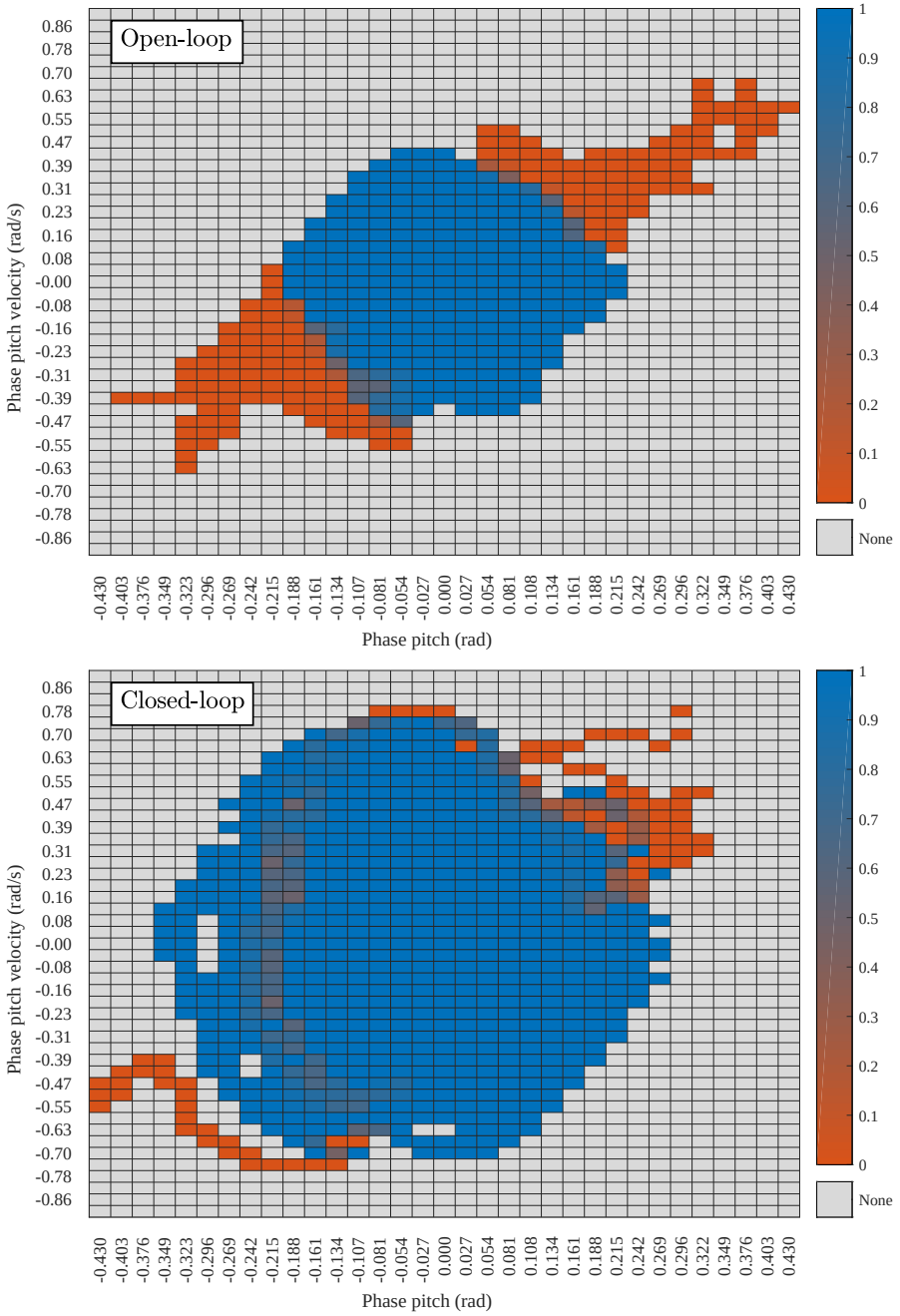


Figure 11.10: Plots of the open-loop vs. closed-loop walking stability of a NimbRo-OP2X robot in the form of a phase space heat map. The blue areas indicate phase states corresponding to stable walking (1.0 = 100% success rate), while the red areas indicate states where the robot tendentially lost balance and fell over (0.0 = 0% success rate). In-between colours indicate that certain binned states were encountered multiple times with different outcomes. The entirety of the data from the push tests has been incorporated into this figure, including also the states of normal balanced walking between pushes. The grey cells correspond to phase states that were not encountered during the tests.

12 CONCLUSION

Bipedal walking does not always require overly complex stabilisation mechanisms to achieve high levels of robustness. The tilt phase and direct fused angle feedback controllers developed in this thesis are a demonstration of this fact. Both controllers extend an underlying open-loop gait with a diverse array of corrective actions that apply effective long and short-term adjustments to the balance of the robot.[1] With just the pitch and roll orientations of the trunk as the sole source of feedback, the controllers apply relatively simple yet well-thought-out feedback pipelines to activate the required corrective actions at the required times. It is a very important feature of the controllers that they do not apply corrective actions at all if the robot is walking within a certain margin of its nominal orientation limit cycle. The application of step size adjustments is crucially also only used as a last resort for keeping balance, i.e. only in cases where the deviation from normal walking is predicted to be too severe for the remaining corrective actions to be able to deal with it. This is important as it minimises the effect that the balance controllers have on disturbing the intended walking direction and footstep placements of the robot.

The use of open-loop gait generators as the underlying principle of walking is advantageous, as it reduces an otherwise high-dimensional control problem into an interface that has just a few very manageable corrective action inputs. Further advantages of the use of gait generators include the flexibility in manual correction and tuning that they provide, and the observation that they allow robots to find their own natural rhythm while walking, instead of imposing one that was calculated from some trajectory optimisation problem. Both the Central Pattern Generator (CPG) and Keypoint Gait Generator (KGG) are analytically formulated, and along with their respective aforementioned balance controllers, are very computationally inexpensive due to this trait.

Most of the methods presented in this thesis were specifically designed with low-cost robots in mind. Low-cost robots often do not have a wide variety of sensors available, and the sensors that they do have are not always of good quality. As a result, the feedback controllers that were designed to stabilise the gaits in this thesis were formulated in such a way that they only assumed the presence of a single 6-axis Inertial Measurement Unit (IMU). This makes the gaits extraordinarily portable between robots, as there is no requirement

1 Examples of such corrective actions include step size, timing and arm/foot tilt adjustments.

for force-torque sensors, foot contact sensors, or anything of the like. Extensive thought was also put into calibration and filtering of the sensor data, to make the most of the signal amongst the noise. One particular success in this direction was the introduction of the gyroscope bias autocalibration scheme, which significantly increased the integration accuracy of the attitude estimator presented in Chapter 7.

In addition to their general limitations in sensing, low-cost robots are also fundamentally imprecise about the motions they try to execute. Problems such as low power-to-weight ratios, structural non-rigidities, joint backlash and stiction make the tracking of any precise trajectories difficult, or even impossible. The use of tuneable gait generators in combination with a feed-forward actuator control scheme helped deal with these limitations. The sparing use of any physical and/or dynamic models of the robot also contributed to a wider and arguably easier field of applicability of the developed feedback gaits.

Many of the developments in state estimation and bipedal walking that were made in this thesis were founded on other developments that were made in the field of 3D rotation theory. The most significant of these developments were the novel use of fused yaw and tilt to partition 3D rotations into highly problem-relevant rotation subcomponents, and the corresponding development of the tilt angles, fused angles and tilt phase space representations. In addition to providing mathematically and geometrically useful concepts of 'yaw', 'pitch' and 'roll', these developments in 3D rotation theory also led to the introduction of the concept of tilt vector addition. This concept proved to be very useful throughout the formulation of the KGG and the corresponding tilt phase controller.

Validated by experiments that were performed in both simulation and on real robot hardware, the core principle of this thesis can be summarised as follows—if the sensor management and feedback chains are carefully constructed, comparatively simple model-free and robot-agnostic feedback mechanisms can successfully and robustly stabilise a generic bipedal gait.

12.1 FUTURE DIRECTIONS

Several possible directions of future research exist that would build on this thesis. These directions include:

- Investigating the use of multiple intentionally non-aligned gyroscope and accelerometer sensors for the purposes of increased attitude estimation fidelity and better rejection of noise and outliers.

- Applying the nonlinear tripendulum model to the lateral plane of motion as well, and activating the lateral step size (and possibly swing out) corrective action based on that.
- Using learning methods to learn the interplay between lateral step size and timing and the respective effects on the progression of the lateral tilt of the robot, in order to concurrently predict suitable activation values.
- Using learning methods to learn a higher level controller for the KGG, and all of its corrective actions.[2]
- Reformulating the capture step controller to avoid the complete asynchronicity of the RX and MX models. Gradual total dissociation of the states of these two models is the leading observed cause of failure of tuning of the capture step controller.
- Modelling the changing z-rotations of the feet in a more explicit way in terms of how it affects the balance of the robot throughout the duration of a step, and appropriately modifying the constructed feedback controllers in light of this.
- Extending the capture step controller to a full 2.5D treatment (2D position plus heading), to avoid the segregated treatment of the sagittal and lateral planes of motion, and avoid the lack of consideration of the effects of foot yaw.

2 As a step in this direction, for example, Bayesian optimisation for feedback stabilisation of the CPG gait has been performed (Rodriguez et al., 2018).

BIBLIOGRAPHY

J. Allali, R. Fabre, L. Gondry, L. Hofer, O. Ly, S. N'Guyen, G. Passault, A. Pirrone and Q. Rouxel (2018). "Rhoban Football Club: RoboCup Humanoid Kid-Size 2017 Champion Team Paper". In: *RoboCup 2017: Robot World Cup XXI*. LNCS 11175, Springer, pp. 423–434.

J. Allali, L. Gondry, L. Hofer, P. Laborde-Zubieta, O. Ly, S. N'Guyen, G. Passault, A. Pirrone and Q. Rouxel (2019). *Rhoban Football Club – Team Description Paper*. Tech. rep. University of Bordeaux.

P. Allgeuer (Aug. 2016). *Attitude Estimator*. URL: https://github.com/AIS-Bonn/attitude_estimator.

P. Allgeuer (Jan. 2018a). *Keypoint Gait Generator (Reference Matlab Implementation)*. URL: https://github.com/AIS-Bonn/keypoint_gait_generator.

P. Allgeuer (Jan. 2018b). *Matlab/Octave Rotations Library*. URL: https://github.com/AIS-Bonn/matlab_octave_rotations_lib.

P. Allgeuer (Jan. 2018c). *Rotations Conversion Library*. URL: https://github.com/AIS-Bonn/rot_conv_lib.

P. Allgeuer (2020). "Analytic Bipedal Walking with Fused Angles and Corrective Actions in the Tilt Phase Space". *Full version of this thesis*. PhD thesis. University of Bonn. URL: https://arxiv.org/pdf/2011.10339.

P. Allgeuer and S. Behnke (2013). "Hierarchical and State-based Architectures for Robot Behavior Planning and Control". In: *8th Workshop on Humanoid Soccer Robots, International Conference on Humanoid Robots (Humanoids)*. Atlanta, USA.

P. Allgeuer and S. Behnke (2014). "Robust Sensor Fusion for Robot Attitude Estimation". In: *International Conference on Humanoid Robots (Humanoids)*. Madrid, Spain.

P. Allgeuer and S. Behnke (2015). "Fused Angles: A Representation of Body Orientation for Balance". In: *International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany.

P. Allgeuer and S. Behnke (2016). "Omnidirectional Bipedal Walking with Direct Fused Angle Feedback Mechanisms". In: *International Conference on Humanoid Robots (Humanoids)*. Cancún, Mexico.

P. Allgeuer and S. Behnke (2018a). "Bipedal Walking with Corrective Actions in the Tilt Phase Space". In: *International Conference on Humanoid Robots (Humanoids)*. Beijing, China.

P. Allgeuer and S. Behnke (2018b). "Fused Angles and the Deficiencies of Euler Angles". In: *International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain.

P. Allgeuer and S. Behnke (2018c). "Tilt Rotations and the Tilt Phase Space". In: *International Conference on Humanoid Robots (Humanoids)*. Beijing, China.

C. Azevedo, P. Poignet and B. Espiau (2002). "Moving Horizon Control for Biped Robots Without Reference Trajectory". In: *International Conference on Robotics and Automation (ICRA)*. Washington, USA.

J. Balaram (2000). "Kinematic Observers for Articulated Rovers". In: *International Conference on Robotics and Automation (ICRA)*. San Francisco, USA.

C. Brasseur, A. Sherikov, C. Collette, D. Dimitrov and P.-B. Wieber (2015). "A Robust Linear MPC Approach to Online Generation of 3D Biped Walking Motion". In: *International Conference on Humanoid Robots (Humanoids)*. Seoul, Korea.

S. Caron (2019). "Biped Stabilization by Linear Feedback of the Variable-Height Inverted Pendulum Model". In: *arXiv preprint 1809.07073 [cs.RO]*.

S. Caron, A. Kheddar and O. Tempier (2019). "Stair Climbing Stabilization of the HRP-4 Humanoid Robot using Whole-body Admittance Control". In: *International Conference on Robotics and Automation (ICRA)*. Montréal, Canada.

A. Cavallo, A. Cirillo, P. Cirillo, G. De Maria, P. Falco, C. Natale and S. Pirozzi (2014). "Experimental Comparison of Sensor Fusion Algorithms for Attitude Estimation". In: *19th International Federation of Automatic Control World Congress* 47.3, pp. 7585–7591.

J. Crassidis, F. Markley and Y. Cheng (2007). "A Survey of Nonlinear Attitude Estimation Methods". In: *Journal of Guidance, Control and Dynamics* 30.1, pp. 12–28.

H. Diedam, D. Dimitrov, P.-B. Wieber, K. Mombaur and M. Diehl (2008). "Online Walking Gait Generation with Adaptive Foot Positioning Through Linear Model Predictive Control". In: *International Conference on Intelligent Robots and Systems (IROS)*. Nice, France.

J. Englsberger, G. Mesesan and C. Ott (2017). "Smooth Trajectory Generation and Push-recovery Based on Divergent Component of Motion". In: *International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, Canada.

J. Englsberger, G. Mesesan, A. Werner and C. Ott (2018). "Torque-based Dynamic Walking – A Long Way from Simulation to Experiment". In: *International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia.

J. Englsberger, C. Ott and A. Albu-Schäffer (2013). "Three-dimensional Bipedal Walking Control Using Divergent Component of Motion". In: *International Conference on Intelligent Robots and Systems (IROS)*. Tokyo, Japan.

J. Englsberger, C. Ott and A. Albu-Schäffer (2015). "Three-dimensional Bipedal Walking Control Based on Divergent Component of Motion". In: *IEEE Transactions on Robotics* 31.2, pp. 355–368.

J. Englsberger, C. Ott, M. Roa, A. Albu-Schäffer and G. Hirzinger (2011). "Bipedal Walking Control Based on Capture Point Dynamics". In: *International Conference on Intelligent Robots and Systems (IROS)*. San Francisco, USA.

M. Euston, P. Coote, R. Mahony, J. Kim and T. Hamel (2008). "A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV". In: *International Conference on Intelligent Robots and Systems (IROS)*. Nice, France.

M. Felis (2017). "RBDL: An Efficient Rigid-body Dynamics Library Using Recursive Algorithms". In: *Autonomous Robots* 41.2, pp. 495–511.

S. Feng, B. Xinjilefu, C. Atkeson and J. Kim (2015). "Optimization Based Controller Design and Implementation for the Atlas Robot in the DARPA Robotics Challenge Finals". In: *International Conference on Humanoid Robots (Humanoids)*. Seoul, Korea.

D. Gebre-Egziabher, R. Hayward and J. D. Powell (2004). "Design of Multi-Sensor Attitude Determination Systems". In: *IEEE Transactions on Aerospace and Electronic Systems* 40.2, pp. 627–649.

R. Griffin, S. Bertrand, G. Wiedebach, A. Leonessa and J. Pratt (2017). "Capture Point Trajectories for Reduced Knee Bend Using Step Time Optimization". In: *International Conference on Humanoid Robots (Humanoids)*. Birmingham, UK.

R. Griffin and A. Leonessa (2019). "Model Predictive Control for Stable Walking Using the Divergent Component of Motion with Footstep Location and Yaw Adaptation". In: *International Journal of Humanoid Robotics* 16.5.

K. Harada, S. Kajita, K. Kaneko and H. Hirukawa (2004). "An Analytical Method on Real-time Gait Planning for a Humanoid Robot". In: *International Conference on Humanoid Robots (Humanoids)*. Santa Monica, USA.

K. Harada, S. Kajita, K. Kaneko and H. Hirukawa (2006). "An Analytical Method on Real-time Gait Planning for a Humanoid Robot". In: *International Journal of Humanoid Robotics* 3.1, pp. 1–19.

B. Hengst (2014). *rUNSWift Walk2014 Report RoboCup Standard Platform League*. Tech. rep. University of New South Wales.

B. Henze, M. A. Roa and C. Ott (2016). "Passivity-based Whole-body Balancing for Torque-controlled Humanoid Robots in Multi-contact Scenarios". In: *International Journal of Robotics Research* 35.12, pp. 1522–1543.

K. Hirai, M. Hirose, Y. Haikawa and T. Takenaka (1998). "The Development of Honda Humanoid Robot". In: *International Conference on Robotics and Automation (ICRA)*. Leuven, Belgium.

M. Hofmann, I. Schwarz, O. Urbann and A. Larisch (2018). *Nao Devils Team Report 2018*. Tech. rep. Technische Universität Dortmund.

M. Hopkins, D. Hong and A. Leonessa (2014). "Humanoid Locomotion on Uneven Terrain Using the Time-Varying Divergent Component of

Motion". In: *International Conference on Humanoid Robots (Humanoids)*. Madrid, Spain.

K. Jensen (2011). "Generalized Nonlinear Complementary Attitude Filter". In: *Journal of Guidance, Control and Dynamics* 34.5, pp. 1588–1592.

S. Kajita, M. Benallegue, R. Cisneros, T. Sakaguchi, S. Nakaoka, M. Morisawa, H. Kaminaga, I. Kumagai, K. Kaneko and F. Kanehiro (2018). "Biped Gait Control Based on Spatially Quantized Dynamics". In: *International Conference on Humanoid Robots (Humanoids)*. Beijing, China.

S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi and H. Hirukawa (2003). "Biped Walking Pattern Generation by Using Preview Control of Zero-Moment Point". In: *International Conference on Robotics and Automation (ICRA)*. Taipei, Taiwan.

S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi and H. Hirukawa (2001). "The 3D Linear Inverted Pendulum Mode: A Simple Modeling for a Biped Walking Pattern Generation". In: *International Conference on Intelligent Robots and Systems (IROS)*. Maui, USA.

M. Khadiv, S. Kleff, A. Herzog, A. Moosavian, S. Schaal and L. Righetti (2016). "Stepping Stabilization Using a Combination of DCM Tracking and Step Adjustment". In: *International Conference on Robotics and Mechatronics (ICROM)*. Tehran, Iran.

M. Kim, D. Lim and J. Park (2019). "Online Walking Pattern Generation for Humanoid Robot with Compliant Motion Control". In: *International Conference on Robotics and Automation (ICRA)*. Montréal, Canada.

T. Koolen, S. Bertrand, G. Thomas, T. de Boer, T. Wu, J. Smith, J. Englsberger and J. Pratt (2016). "Design of a Momentum-based Control Framework and Application to the Humanoid Robot Atlas". In: *International Journal of Humanoid Robotics* 13.1.

T. Koolen, T. de Boer, J. Rebula, A. Goswami and J. Pratt (2012). "Capturability-based Analysis and Control of Legged Locomotion, Part 1: Theory and Application to Three Simple Gait Models". In: *International Journal of Robotics Research* 31.9, pp. 1094–1113.

P. Kryczka, P. Kormushev, N. Tsagarakis and D. Caldwell (2015). "Online Regeneration of Bipedal Walking Gait Optimizing Footstep Placement and Timing". In: *International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany.

P. Kryczka, Y. Shiguematsu, P. Kormushev, K. Hashimoto, H.-o. Lim and A. Takanishi (2013). "Towards Dynamically Consistent Real-time Gait Pattern Generation for Full-size Humanoid Robots". In: *International Conference on Robotics and Biomimetics (ROBIO)*. Shenzhen, China.

S. Kuindersma, F. Permenter and R. Tedrake (2014). "An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion".

In: *International Conference on Robotics and Automation (ICRA)*. Hong Kong, China.

S. Madgwick, A. Harrison and R. Vaidyanathan (2011). "Estimation of IMU and MARG Orientation Using a Gradient Descent Algorithm". In: *International Conference on Rehabilitation Robotics*. Zürich, Switzerland.

R. Mahony, T. Hamel and J.-M. Pflimlin (2008). "Nonlinear Complementary Filters on the Special Orthogonal Group". In: *IEEE Transactions on Automatic Control* 53.5, pp. 1203–1218.

S. McGill, S.-J. Yi, D. Hong and D. Lee (2015). *Team THORwIn: Team Description for Humanoid AdultSize League of RoboCup 2015*. Tech. rep. University of Pennsylvania.

G. Mesesan, J. Englsberger, G. Garofalo, C. Ott and A. Albu-Schäffer (2019). "Dynamic Walking on Compliant and Uneven Terrain using DCM and Passivity-based Whole-body Control". In: *International Conference on Humanoid Robots (Humanoids)*. Toronto, Canada.

G. Mesesan, J. Englsberger, C. Ott and A. Albu-Schäffer (2018). "Convex Properties of Center-of-Mass Trajectories for Locomotion Based on Divergent Component of Motion". In: *IEEE Robotics and Automation Letters* 3.4, pp. 3449–3456.

M. Missura (2015). "Analytic and Learned Footstep Control for Robust Bipedal Walking". PhD thesis. School of Mathematics and Science. URL: http://hss.ulb.uni-bonn.de/2016/4268/4268.pdf.

M. Missura and S. Behnke (2013). "Self-stable Omnidirectional Walking with Compliant Joints". In: *8th Workshop on Humanoid Soccer Robots, International Conference on Humanoid Robots (Humanoids)*. Atlanta, USA.

M. Morisawa, K. Harada, S. Kajita, K. Kaneko, J. Sola, E. Yoshida, N. Mansard, K. Yokoi and J.-P. Laumond (2009). "Reactive Stepping to Prevent Falling for Humanoids". In: *International Conference on Humanoid Robots (Humanoids)*. Paris, France.

M. Morisawa, K. Harada, S. Kajita, S. Nakaoka, K. Fujiwara, F. Kanehiro, K. Kaneko and H. Hirukawa (2007). "Experimentation of Humanoid Walking Allowing Immediate Modification of Foot Place Based on Analytical Solution". In: *International Conference on Robotics and Automation (ICRA)*. Roma, Italy.

M. Morisawa, F. Kanehiro, K. Kaneko, N. Mansard, J. Sola, E. Yoshida, K. Yokoi and J.-P. Laumond (2010). "Combining Suppression of the Disturbance and Reactive Stepping for Recovering Balance". In: *International Conference on Intelligent Robots and Systems (IROS)*. Taipei, Taiwan.

R. Munguía and A. Grau (2014). "A Practical Method for Implementing an Attitude and Heading Reference System". In: *International Journal of Advanced Robotic Systems* 11.4, pp. 1–12.

K. Nishiwaki and S. Kagami (2010). "Strategies for Adjusting the ZMP Reference Trajectory for Maintaining Balance in Humanoid

Walking". In: *International Conference on Robotics and Automation (ICRA)*. Anchorage, USA.

A. Pajon and P.-B. Wieber (2019). "Safe 3D Bipedal Walking through Linear MPC with 3D Capturability". In: *International Conference on Robotics and Automation (ICRA)*. Montréal, Canada.

J. Pratt, J. Carff, S. Drakunov and A. Goswami (2006). "Capture Point: A Step Toward Humanoid Push Recovery". In: *International Conference on Humanoid Robots (Humanoids)*. Genova, Italy.

J. Pratt, T. Koolen, T. de Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson and P. Neuhaus (2012). "Capturability-based Analysis and Control of Legged Locomotion, Part 2: Application to M2V2, a Lower-body Humanoid". In: *International Journal of Robotics Research* 31.10, pp. 1117–1133.

J. Pratt, B. Krupp, V. Ragusila, J. Rebula, T. Koolen, N. van Nieuwenhuizen, C. Shake, T. Craig, J. Taylor, G. Watkins, P. Neuhaus, M. Johnson, S. Shooter, K. Buffinton, F. Canas, J. Carff and W. Howell (2009). "The Yobotics-IHMC Lower Body Humanoid Robot". In: *International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, USA.

D. Rodriguez, A. Brandenburger and S. Behnke (2018). "Combining Simulations and Real-robot Experiments for Bayesian Optimization of Bipedal Gait Stabilization". In: *22nd RoboCup International Symposium*. Montréal, Canada.

T. Röfer, T. Laue, A. Baude, J. Blumenkamp, G. Felsch, J. Fiedler, A. Hasselbring, T. Haß, J. Oppermann, P. Reichenberg, N. Schrader and D. Weiß (2019). *B-Human Team Report and Code Release 2019*. Tech. rep. University of Bremen.

Q. Rouxel, G. Passault, L. Hofer, S. N'Guyen and O. Ly (2015). "Rhoban Hardware and Software Open Source Contributions for RoboCup Humanoids". In: *10th Workshop on Humanoid Soccer Robots, International Conference on Humanoid Robots (Humanoids)*. Seoul, Korea.

M. Schwarz and S. Behnke (2013). "Compliant Robot Behavior using Servo Actuator Models identified by Iterative Learning Control". In: *17th RoboCup International Symposium*. Eindhoven, Netherlands.

B. Stephens and C. Atkeson (2010). "Push Recovery by Stepping for Humanoid Robots with Force Controlled Joints". In: *International Conference on Humanoid Robots (Humanoids)*. Taipei, Taiwan.

J. Stuelpnagel (1964). "On the Parametrization of the Three-Dimensional Rotation Group". In: *SIAM Review* 6.4.

T. Takenaka, T. Matsumoto and T. Yoshiike (2009). "Real Time Motion Generation and Control for Biped Robot 1st Report: Walking Gait Pattern Generation". In: *International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, USA.

T. Takenaka, T. Matsumoto, T. Yoshiike, T. Hasegawa, S. Shirokura, H. Kaneko and A. Orita (2009). "Real Time Motion Generation and Control for Biped Robot 4th Report: Integrated Balance Control". In:

International Conference on Intelligent Robots and Systems (IROS). St. Louis, USA.

T. Takenaka, T. Matsumoto, T. Yoshiike and S. Shirokura (2009). "Real Time Motion Generation and Control for Biped Robot 2nd Report: Running Gait Pattern Generation". In: *International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, USA.

Team NimbRo (Sept. 2018). *Humanoid Open Platform ROS Software*. URL: https://github.com/AIS-Bonn/humanoid_op_ros.

R. Tedrake, S. Kuindersma, R. Deits and K. Miura (2015). "A Closed-form Solution for Real-time ZMP Gait Generation and Feedback Stabilization". In: *International Conference on Humanoid Robots (Humanoids)*. Seoul, Korea.

J. Urata, K. Nishiwaki, Y. Nakanishi, K. Okada, S. Kagami and M. Inaba (2011). "Online Decision of Foot Placement Using Singular LQ Preview Regulation". In: *International Conference on Humanoid Robots (Humanoids)*. Bled, Slovenia.

O. Urbann, I. Schwarz and M. Hofmann (2015). "Flexible Linear Inverted Pendulum Model for Cost-effective Biped Robots". In: *International Conference on Humanoid Robots (Humanoids)*. Seoul, Korea.

J. Vaganay, M. Aldon and A. Fournier (1993). "Mobile Robot Attitude Estimation by Fusion of Inertial Data". In: *International Conference on Robotics and Automation (ICRA)*. Atlanta, USA.

M. Vukobratović and B. Borovac (2004). "Zero-Moment Point — Thirty Five Years of its Life". In: *International Journal of Humanoid Robotics* 1.1, pp. 157–173.

P.-B. Wieber (2006). "Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations". In: *International Conference on Humanoid Robots (Humanoids)*. Genova, Italy.

S.-J. Yi, D. Hong and D. Lee (2013). "A Hybrid Walk Controller for Resource-Constrained Humanoid Robots". In: *International Conference on Humanoid Robots (Humanoids)*. Atlanta, USA.

S.-J. Yi, B.-T. Zhang, D. Hong and D. Lee (2011). "Online Learning of a Full Body Push Recovery Controller for Omnidirectional Walking". In: *International Conference on Humanoid Robots (Humanoids)*. Bled, Slovenia.