Learning Probabilistic Feedforward and Feedback Policies for Stable Walking

Lernen von probabilistischen Regelungskonzepten für stabile Gehbewegungen
Master-Thesis von Svenja Stark aus Darmstadt
Tag der Einreichung:

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Dr. Elmar Rückert

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Learning Probabilistic Feedforward and Feedback Policies for Stable Walking
Lernen von probabilistischen Regelungskonzepten für stabile Gehbewegungen

Vorgelegte Master-Thesis von Svenja Stark aus Darmstadt

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Dr. Elmar Rückert

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 13. Januar 2016

(Svenja Stark)

Abstract

The compliant quadruped robot Oncilla is used as platform to explore the benefits of the interaction between a feedforward gait and a simultaneous active stabilizing feedback controller. The chosen approach is motivated by findings in biology as well as by the advantages of modern stochastic methods. In this thesis, we present a balancing controller, a simple feedforward gait and first results of a system combining both components. The basic components can be modified in further research.
The developed balancing controller is based on a common criteria for static stability, the current center of pressure (CoP) of the Oncilla. It is calculated from force data obtained from mounted OptoForce sensors and the endeffector positions calculated by a simplified forward kinematic model. Locally weighted regression is used to calculate motor commands that bring the Oncilla's CoP closer to a desired one.
The static walking gait is based on analyses of the walking behavior of four-legged animals. The trajectory for each leg is further carefully hand-tuned and parameterized as a combination of sine curves. In addition, rhythmic movement primitives modulating the handcrafted gait have been applied.
Finally, we explored how the final gait is influenced by combining the forward gait and the balancing controller. The tuning parameter was the amount of applied feedback, which determines how much the Oncilla relies on the gait or on the feedback. In the current setup, the balancing controller could not visibly improve the feedforward gait.
This thesis provides first results towards a versatile platform allowing further experiments on the benefits of feedback for gait learning.

Zusammenfassung

Der nachgiebige Roboter Oncilla wird als Plattform verwendet, um die Vorteile der Interaktion zwischen einem vorkoppelnden Gang und einem gleichzeitig aktiven, stabilisierenden Rückkoppelungsregler zu untersuchen. Der gewählte Ansatz ist sowohl motiviert durch Ergebnisse aus der Biologie als auch durch die Vorteile moderner stochastischer Methoden. In dieser Thesis präsentieren wir einen Gleichgewichtsregler, einen einfachen vorkoppelnden Gang und erste Ergebnisse eines Systems, welches beide Komponenten miteinender kombiniert. Die Grundkomponenten können in weiterführender Forschung modifiziert werden.

Der entwickelte Gleichgewichtsregler basiert auf einem verbreiteten Kriterium für statische Stabilität, dem aktuellen Druckmittelpunkt des Oncillas. Dieser wird aus Kraftdaten, welche aus den montierten OptoForce-Sensoren stammen, und den Positionen der Endeffektoren mittels eines vereinfachten kinematischen Modells bestimmt. Lokal Gewichtete Regression wird benutzt um die Motorbefehle zu berechnen, welche den Druckmittelpunkt des Oncillas näher an einen gewünschten Punkt heranbringen.

Das statische Laufen basiert auf Analysen vom Laufverhalten vierbeiniger Tiere. Ferner ist die Trajektorie eines jeden Beines sorgfältig von Hand abgestimmt und als Kombination von Sinuskurven parametrisiert. Zusätzlich wurden rhythmische Bewegungsprimitiven eingesetzt, welche den per Hand erstellten Gang modulieren.

Abschließend haben wir erforscht, wie der finale Gang durch das Kombinieren von vorwärtskoppelndem Gang und Gleichgewichtsregler beeinflusst wird. Der zu bestimmende Parameter war die Menge an hinzugefügter Rüchkkopplung, welche bestimmt, wie sehr der Oncilla sich auf den Gang oder auf den Rückkopplungsregler verlässt. Mit der aktuellen Einstellung konnte der Gleichgewichtsregler den Gang nicht sichtbar verbessern.

Diese Thesis liefert erste Ergebnisse in Richtung einer vielseitigen Plattform und erlaubt weitere Experimente bezüglich des Vorteils einer Rückkopplung beim Lernen eines Ganges.

Contents

Figures and Tables

List of Figures

List of Tables

1 Introduction

1.1 Motivation, Goals and Ideas

Animals with legs can access almost any part of the surface of the earth. In doing so, they show a wide range of flexible movements: jumping, climbing, running, and in some cases swimming and even gliding. This ability allows them to pass difficult and rough terrains such as deserts or mountain areas. Also, they can move incredibly fast and easily adapt to obstacles and disturbances.
A robot with these abilities would be a much more versatile autonomous system than today's common wheeled vehicles. Those are typically limited to easily accessible areas or even depend on paved streets. Thus, the benefit of being able to handle challenges more flexibly provides a lot of possible applications for such a system: It allows for smooth locomotion, as holes or obstacles can be avoided or compensated for with ease, without greatly affecting the environment. Therefore, such a robot could work as a transport system for delicate goods as well as a more flexible and less destructive vehicle in agriculture or forestry. It could also support humans during the exploration of unknown, hard to access areas or in terrain rescue help, and would especially allow remote inspection of areas that are dangerous for humans, like disaster recovery in polluted areas, construction sites or military areas. Another field of support would be everyday human life, as service robots in household and health care have to deal with a human-built world that includes stairs and cluttered rooms, which both serve as great challenges to robots and would hence demand accordingly appropriate movement capabilities. Hence, a walking robot could function as support for humans in various situations and as a replacement in dangerous tasks.
Another application domain is the study of locomotion itself. Robots can provide insights into how evolution shaped bodies and motion patterns, and they allow the testing of hypotheses about why animals move the way they do, for example by comparing gaits optimized towards low consumption of energy with gaits optimized towards low attrition.
The field of animal locomotion has attracted interest for a long time already, at the forefront of which is the study of walking gaits such as the one shown in Figure 1.1. So far, there exist incidences for vertebrates in which simple rhythmic movements such as walking are generated by central pattern generators (CPGs) from a single input signal. The CPGs are located in the spinal cord, and with never-changing ground conditions, they work without requiring addditional information from the brain, as has been shown on decerebrated cats by Whelan [1].
The locomotion patterns produced by the CPGs are classified into different gaits according to their properties. For example, one of these gait properties is the ground reaction time, which is the fraction of time in a walking cycle during which a foot is in contact with the ground. A related property is the number of feet having ground contact at the same time. Based on those two properties, one can separate between dynamic and static gaits. Static gaits are constantly statically balanced; thus, for a quadruped, at least three legs have to be on the ground the whole time. Such gaits are usually called crawling or creeping gaits, and their stability has been analyzed by McGhee in 1968 [2]. The constant balanced state allows animals to perform the gaits arbitrarily slowly, which can be seen when, for instance, a cat creeps up on its prey. In contrast, dynamic gaits of quadrupeds have phases where only two, one, or even no foot at all has ground contact. Balance is kept by performing the next step, which means the animal is actually constantly in flux and rebalancing itself. Thus, the intrinsic balance of the gait itself can be maintained in different ways, either by keeping a constantly statical equilibrium, or by having compensating movements already included in the gait, which both seem to be done without additional information from the brain.

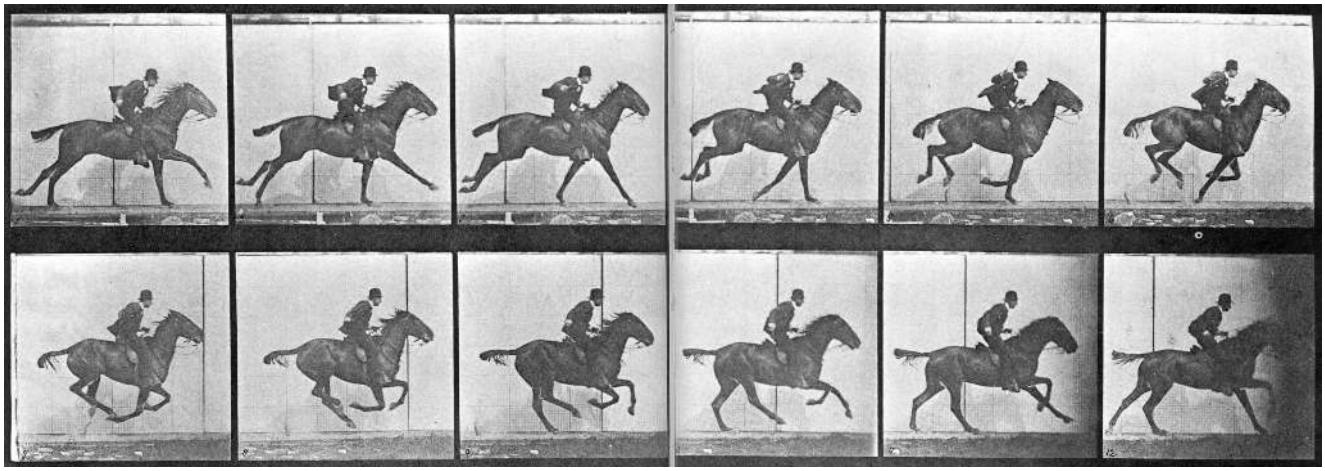


Figure 1.1.: Early photographs that show a sequence of a galloping horse[1]. The pictures were taken by Eadweard Muybridge to ascertain whether a horse has all four hooves off the ground while galloping, which makes them one of the first documents of gait study.

However, since the environment has many irregularities, the simple rhythmic gait patterns can be modified by neural information from the motor cortex. For example, when an obstacle appears, the visual cortex sends commands to the motor cortex, which initiates modulating patterns for CPGs in the spine. These adaptions are often taken to restore balance after facing irregularities in the surroundings. In mammals, the task of (re-)balancing is solved by combining the information provided by several different senses, such as vision, touch, proprioceptive sensors and vestibular sensors. Despite vision often being an important sense for orientation in everyday life and in areas with large obstacles, animals do not necessarily need their vision for locomotion, as they can balance and walk even if visual information is not available. For instance, humans are able of balancing and walking with closed eyes, as long as they do not interfere with large obstacles.

The neural movement generation of animals is strongly intertwined with the bio-mechanical structure of their bodies. For example, the bodies of most vertebrates are compliant, soft and flexible because they consist of muscles and skin. This anatomy allows animals to load and then reuse energy, especially during dynamic walking when larger forces are acting on their bodies, but also to easily cope with stochasticity and small disturbances, like irregularities in the ground.

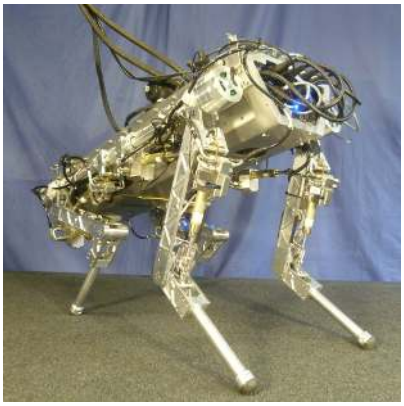
To this day, bringing a robot to life with the same qualities that animals possess has not been fully achieved. One of the difficult problems researchers face is the underactuation of a freely walking system. This means that one cannot directly control position and orientation of the robot, only indirectly by adjusting its joints. This manipulation influences the endeffector positions, usually the feet, and changes the whole system's frame relative to the world frame. In the worst case scenario, it can make the robot end up in a dead lock position by making it fall. A second hard factor for real life application is the many degrees of freedom usually found within a quadrupedal platform, as well as the huge amount of data to process, which includes noise from the system and stochasticity of the environment, coming from many disturbances and variations in the outdoor world, as, for example, through uneven terrain. Thus, a robot that is supposed to be versatile requires elaborated hardware and control.

As animals apparently deal with those problems effortlessly and have already optimized their locomotion over years of evolution, their way of facing those problems is a source of inspiration for robotics. Biological mechanics, sensors and computational power of the brain are until today unmatched by artificial constructions. Thus, we construct components that take inspiration for functionality from biology, but work differently, mostly simpler, than their actual counter parts do in nature.

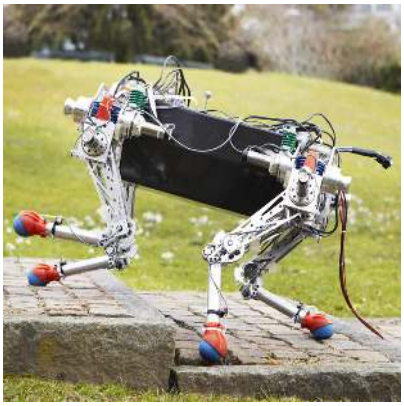
[1] taken from `https://commons.wikimedia.org/wiki/Eadweard_Muybridge#/media/File:Muybridge_horse_gallop.jpg`

(a) RHex[2]

(b) Little Dog[3]

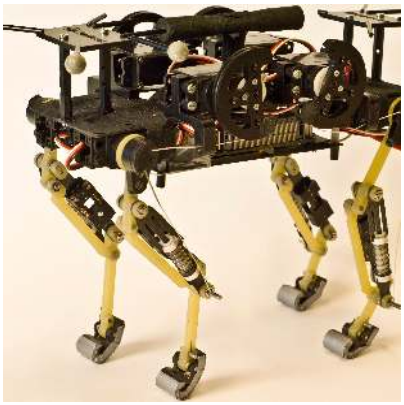


(c) HyQ[4]



(d) StarlETH[5]

(e) Spot[6]



(f) Cheetah-cub[7]

Figure 1.2.: Examples that depict the variety of hardware platforms in locomotion. The leg structures range from simple, one segmented legs to biologically inspired, three segmented pantographic leg constructions.

There is a wide variety of approaches for constructing robots; some are huge simplifications as they, for example, lack knee joints, while others have more degrees of freedom per leg than animals do in nature, allowing the exploration of new movements. Often, the hardware of common robots lack the flexibility and compliance of animals' body structure; they contain parts made of plastic or metal resulting in a rigid body.

An alternative branch of robots has been started in the 80's by Raibert [3], using springs to obtain passive compliance, allowing for dynamic movements such as hopping and the saving of energy as in animals. The idea of compliance is not only important for walking itself, but in the long term, also for interacting with humans and the environment. The research has been continued through the company Boston Dynamics, leading to some of today's most impressive walking robots, Big Dog [4] and Spot [5]. Those robots also make use of another technology for compliance: Hydraulics. To obtain a perceived compliance in rigid constructed systems, one can simulate compliant behavior by, for instance, actively contracting legs while running. This solution called active compliance is, of course, not as energy efficient

[2] taken from `http://www.pro-linux.de/images/NB3/imgdb/x-rhex-in-der-wueste.jpg`
[3] taken from `http://www.bostondynamics.com/img/LittleDog_Terrain.png`
[4] taken from `http://new.semini.ch/research/hyq-robot/`
[5] taken from `http://robohub.org/four-legged-robot-that-efficiently-handles-challenging-terrain/`
[6] taken from `http://icdn1.digitaltrends.com/image/bdspot1-1500x1000.jpg`
[7] taken from `http://biorob.epfl.ch/files/content/sites/biorob/files/users/207295/public/Pictures/CheetahCubMarkers.jpg`

as real compliances and also rather distant from biology. Still, it offers some benefits, as it allows for variable stiffness.
The drawback of compliant hardware is that it makes precise control harder in comparison to well actuated rigid robots. Thus, there is a tradeoff between flexibility and precision for the hardware, having large influence on the control level for robots. This may be the reason why often times only limited types of gaits exist for each robot, either dynamic or static ones. Some research has been done by Takamuku et al. [6] and Blickhan et al. [7] which study the abilities that the shape of the body gives to a robot or an animal. Related to these studies is the idea to develop hardware that is close to the animal movement range, for example by resembling a cat's or a dog's leg behavior as has been done for the Cheetah [8] and the Oncilla [9].
For obtaining the information that animals draw from their senses, robots are equipped with sensors in joints for proprioception, force sensors for tactile sensing, cameras for vision and an IMU replacing the vestibular system, as well as many more sensors, among others, to measure the level of the battery charging. In the implementations, the information from tactile sensors is often used to calculate common stability criterions for keeping the statical balance of a moving system. Options are the center of gravity (CoG), the zero moment point (ZMP) discussed by Vukobratovic [10] or the center of pressure (CoP). Another approach is to use neural networks, which find useful information in data without explicitely modeling it.
For the control model, the approaches for walking range from biology-related approaches resembling the neurological relations to more abstract statistical ones which allow efficient solutions and exploit new possibilities. Many researchers took inspiration from the central pattern generators observed in animals. Different CPG models are developed for rhythmic gait generation, some examples of which are described in [11]. They can be used directly as simple feedforward commands without requiring any information about the hardware and thus being model-free. Often, however, CPGs are modulated by reflexes or improved by adding feedback to allow them to react to disturbances. For example, reflexes can be used to lengthen the swing phase in case of a hole or to shorten it in case of an early ground contact to prevent the robot from falling. CPGs have been applied to several robots, for example to a simple quadruped without knees, allowing to learn different gaits depending on the desired velocity, like in the work done by Owaki [12]. Impressive results regarding real world application are the hexapod Rhex [13] and the quadrupedal Tekken II [14], which are both able to walk outdoors.
Another approach to generate walking patterns is, for example, a force threshold based state machine, as had been utilized by Palankar and Palmer [15] for a hexapod.
As there is a large variety in the hardware of walking robots (some examples are shown in Figure 1.2), it is often difficult to compare different control approaches in locomotion, as the hardware tends to already be tuned to the specific task the robot is supposed to work for. For instance, the quite specific foot structure of the Tekken II could hardly be adopted for other settings.
Still, researchers who are working on endowing robots with the similar movement abilities as animals usually furthermore want practicability and simplicity for their approaches and therefore aim at the following features: Smooth movements of the robot, robustness with respect to interruptions, changing of the speed, adaption of gait details such as step height or length as well as switching between gaits or even movements such as jumping, walking or swimming. These features should be encodable in a model with just a few parameters, such that they are easy to tune.
We want to use a platform that has been determined for and already be proven to work for dynamic walking and use it for static walking. Such slow movements may be desired in the future for accompanying humans and for crossing rough or unpredictable terrain. Also, they are supposed to be energy efficient for animals which raises the hope of gaining the same effect for robot locomotion.

1.2 Related Work

In this section we reference related work on walking robots and tactile sensors. Remarkable success has been achieved in the field of hexapods. Palankar and Palmer [15] proposed a simple but effective approach of using force sensors. A state machine generates the gait by switching states according to hand-tuned force thresholds. So the measured ground forces modulate the swing and stance phase, as the phase length depends on the strength of the ground contact. The hexapod is able to cross artificial rough terrain. Another interesting hexapod is the robot RHex [16]. It has simple constructed legs without knees that move in a circular fashion which is different from biologic six-legged counterparts such as insects and instead rather resembles the way wheels work. They enable the robot to walk forward, backward, to turn, and also to cross a large variation of terrain such as stones and sand. The robot is constructed robustly enough such that mudd and water do not pose a problem.

For quadrupeds, RoboSimian [17] is a novel approach, as it has four extremities but is not limited to quadrupedal walking. It is thought to perform a mixture between bipedal and quadrupedal locomotion like an ape, an idea that recently has been explored for the Honda ASIMO as well [18]. The construction of RoboSimian's legs and arms is not bio-inspired as it is intended to serve in space. Therefore it has high (statical) stability that prevents it from falling, and it has no passive compliance which allows precise movements. It has participated in the DARPA challenge [19], where it used legs and wheels allowing for a wide range of movements.

Owaki et al. [20] use a robot without knee joints. CPGs generate the shoulder movements. They are modulated only by intraleg force feedback, which means that each leg receives force feedback solely from its own force sensor. It explores how different gaits from animals are learned depending on speed or on an added pendulum. This approach distincts from most others, as it does not contain a predefined leg trajectory coupling defining the gait; the coordination results from the applied feedback alone.

A large project with several researchers using the same platform was done within the DARPA project "Learning Locomotion (L2)". This project resulted in more comparative work achieved by [21], [22], [23], [24] and [25], all using a Little Dog [26] robot from Boston Dynamics for their research. The main focus was to surpass very rough terrain, requiring pre-planning of sensible trajectories and foothold placement search. Therefore, the participants used detailed information about the landscape, which had been provided beforehand, as well as motion capture equipment. The resulting gaits were closer to human climbing than to performing a rhythmic gait pattern, as each single step was planned carefully. Thus, despite optimizing toward speed, the results were rather static with slow movements. The legs were usually bent a lot to have a low center of mass, making it easy to keep balance. The robot includes ground contact sensors which were used by Righetti et al. [27] to avoid slipping by optimizing to the least possible tangential ground forces. Also, a form of impedance control has been attempted by Buchli et al. [28] to simulate compliance, but the hardware impeded more dynamic behavior.

BigDog [4] and the newly introduced Spot [5], both also from Boston Dynamics, were used to produce some of the most impressive current results. They are both rather large quadrupeds, about the size of mules, and are supposed to accompany humans. Videos show how the robots are able to walk autonomously outdoors, cross slopes of hills and rough terrain and even handle slippery ice. Their appearance seems to be biology inspired, and they perform a trotting gait. To achieve that, the hardware includes springs in the lowest leg segment. Also, a high amount of sensors and well developed hardware such as hydraulic actuators are used. However, the robots' exact way of function is unclear, as no detailed report is available. So far, no other gait has been shown, so very slow movements do not seem to be possible yet.

HyQ, [29] and recently [30], is a slightly smaller robot with a weight between 70kg and 90kg. It works with hydraulic actuators and electric motors. HyQ uses active impedance, which is purely simulated on the control level and models the compliance of springs whose stiffness can be adapted as required during execution. This feature allows HyQ a wide movement range from precise, stiff movements to trotting and even fly trotting. It also can absorb the shock after being dropped and is able to jump with all feet

off the ground by itself. For control, foot trajectories are generated in task space based on CPGs. They can be modulated by foot contact information. Inverse kinematics and inverse dynamics are required for transforming the trajectories to torque control commands. Trunk control and push recovery stabilize the robot. Hence, HyQ delivers remarkable results but it requires good models of the hardware and cannot exploit energy efficient movements. The MIT Cheetah [31] uses impedance control as well and it is able to gallop, detect obstacles and cross them by jumping like a horse as can be seen in [32].
A smaller quadruped which is as well designed for dynamic movements is the StarlETH [33]. It has force sensors on each foot and it includes springs to allow passive compliance while it uses active impedance as well. A wide range of abilities have been shown on this platform: Like HyQ, it can cope with slipping and dropping and is able to trot as well as to walk, even when it is disturbed meanwhile. Contact forces of the feet are optimized by minimizing tangential forces.
Another robot which has been designed for more dynamic movements is the Cheetah-cub [34]. Its leg design is oriented on that of mammals: It has three segments and is thus more complex than most of the other introduced robots which have two ore one segmented legs. The legs also include springs for passive compliance, allowing to perform fast trotting gaits that work with simple feedforward CPG control. The passive compliance leads to entrainment when the robot has ground contact, changing the actual performed leg trajectory.
For the Oncilla [9], which is used in the present work and described in detail in Subsection 2.1, a similar leg structure has been used. Degrave [35] lets the Oncilla turn and trot by using different strategies such as sine curves and trajectories generated in task space. In [36], a CPG based on Hopf-oscillators is used to generate a trotting gait. The generated gait trajectories are modulated by the output of a neural network to stabilize them. The modulation depends on information about roll and pitch gained from an IMU or a camera; the parameters were optimized using Particle Swarm Optimization (PSO). Ajallooeian [37] improves the balance by using Virtual Model Control (VMC). This model-based posture control generates forces to rebalance the robot through virtual springs attached from parts of the actual dislocated robot to a desired pose. This approach can already be seen as some kind of active compliance. The control system is good enough such that according to [38], the Oncilla is able to walk outdoors.
For most of the current dynamic gaits it appears that they have a rather high frequency. This property results in very short swing phases of the legs which often do not cover much distance each, in order to keep the balance of the robot. We do not see such a walking behavior when we look at animals.

1.3 Outlook

In this thesis, we contribute to solving the task of locomotion with a bio-inspired idea:
In order to obtain a robust walking gait, we provide tactile feedback that in future will faciliate safe policy exploration, whose search space we narrow down by avoiding unstable positions and decisions. For that, we develop a balancing controller that depends on the center of pressure, a transformation of force measurements from tactile sensors. The controller is described in Subsection 3.1 and evaluated on our platform in Subsection 4.1.
Additionally, we use a carefully tuned handcrafted feedforward gait, which in future will be modulated by a statistical approach for gait generation that provides mean and variance over a set of executed movements. The background about gaits is provided in Subsection 3.2, the evaluation of the actual gait is described in Subsubsection 4.2.1, as well as a first learned modulation giving an insight into the parameter complexity of future approaches is shown in Subsubsection 4.2.2.
As a first approach, we simply add feedback generated from the balancing controller. Some first results can be found in Subsection 4.3. In the future, we want to refine the approach by using adaptive feedback that varies according to the reliability of the feedforward command. The goal of this approach is to work without using any information about the structure of the environment or (pre-)planning a long run trajectory to negotiate obstacles.

2 The Oncilla Platform

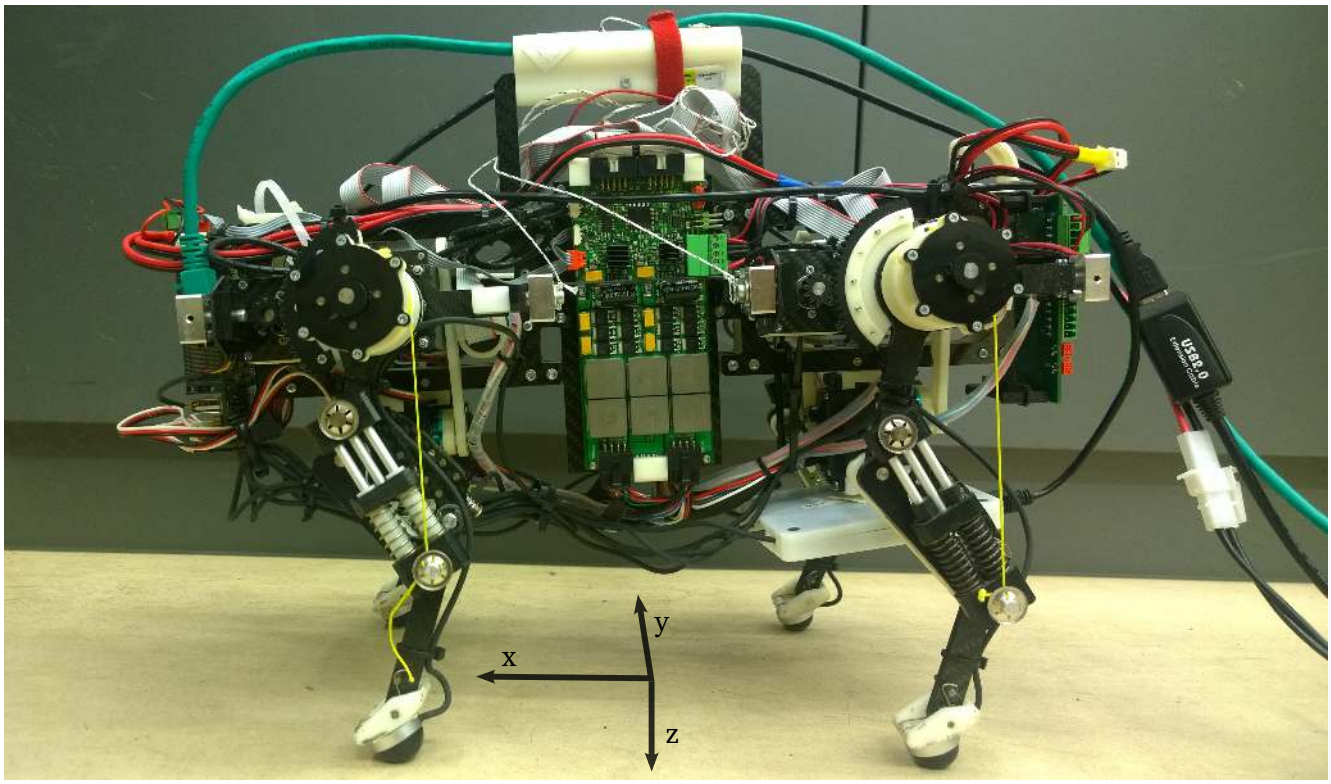
2.1 Oncilla Hardware

The Oncilla is a quadrupedal robot from the AMARSi project and a successor of the Cheetah [8], pictures can be seen in Figure 2.1. It has the size and weight of a cat and a rigid trunk. The legs have five joints each: a lateral hip joint, a sagittal hip joint, a knee, an ankle and a joint where the foot is attached. The first and the second joint, q_0 and q_1, are both actuated. The next two joints, q_2 and q_3, are coupled with each other and are actuated by proximally mounted actuators working via a cable mechanism, flexing the knee and thus shortening the leg length. A spring works as the antagonist, stretching the leg when the cable is loosened. The last joint, q_4, has no actuator.
Hence, the Oncilla has 12 active degrees of freedom in total. For an overview see Table 2.1. Throughout the whole work presented here, we use the legs only in the parasagittal plane, reducing the amount of DOFs to eight. The legs themselves are referred to as the left front leg being L1 or LF, the right front leg L2 or RF, the left hind leg L3 or LH and the right hind leg L4 or RH.
The creators of the Oncilla were inspired by biology, leading to a robot having similar movement behavior as a mammal. Therefore it has three segmented pantographic legs, with the first and the last segment keeping the same angle to each other for most of the movements, as there is a coupling between knee and ankle joint. Less biologically inspired is the leg configuration of the Oncilla. The natural leg configuration of animals lets the knees point forward, while the elbows point backwards. The legs of the Oncilla are arranged such that knees and elbows all point forwards.
Another biological feature is compliance, which allows loading of energy and coping with small irregularities and disturbances. Passive compliance is included in the Oncilla in form of springs in the leg segments and in the lowest joint q_4. The goal is to allow the benefit of energy efficient solutions as well as requiring less precision in control. For more technical details on the Oncilla, see work of Sproewitz et al. [9] and especially the one of Rutishauser et al. [8] for the leg construction.
In return, these biological features result in highly non-linear dynamics. For instance, because of the compliance, in-air leg trajectories differ from on-ground trajectories, since ground contact gives non-linear perturbations according to Sproewitz et al. [39]. Due to friction, the current springs need noticeable time to stretch. Furthermore, they are too stiff to contract passively while walking, but especially the front legs have too little resilience to stretch the leg when it bears about 2/5 of the robots weight. The

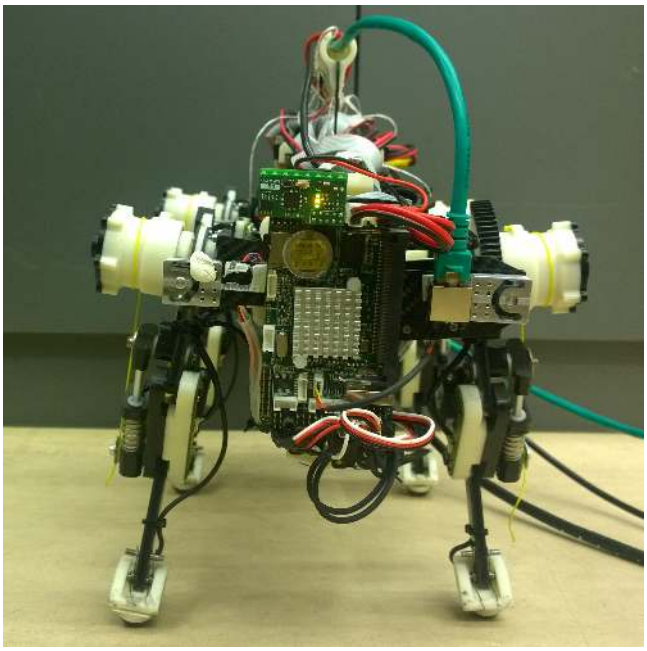
joint	actuated	motor position information	magnetic encoder	symbol
lateral shoulder / hip joint	yes	MP0	-	q_0
sagittal shoulder / hip joint	yes	MP1	ME1	q_1
elbow / knee joint	yes	MP2	ME2	q_2
wrist / ankle joint			ME3	q_3
metacarpophalangeal / metatarsophalangeal joint	no	-	-	q_4

Table 2.1.: Overview of naming, actuation and available sensors of the Oncilla joints.

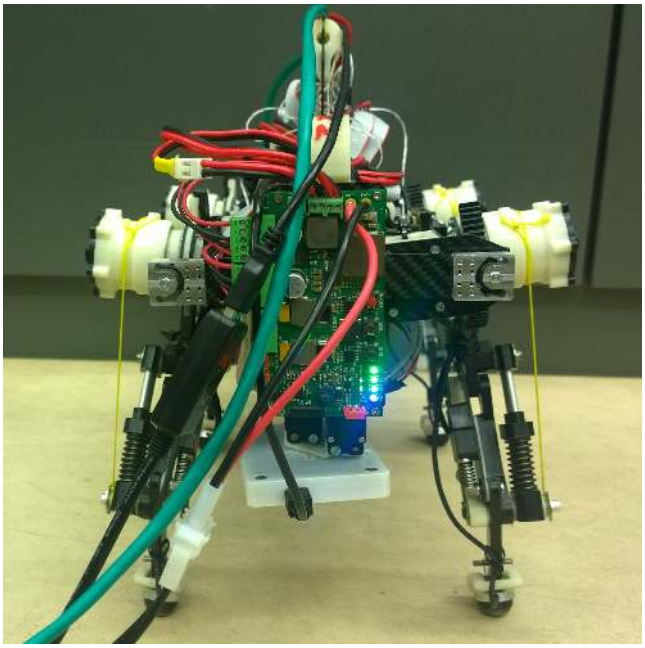
leg construction also does not allow a linear retraction, the leg moves backwards when retracted and to the front when stretched, making it harder to plan movement trajectories. Hence, the Oncilla is a noisy system with difficulties providing accurate joint and end effector positions. For all experiments, the Oncilla is powered externally and thus needs a cable connection.



(a) left side



(b) front



(c) back

Figure 2.1.: The pictures show different views of the Oncilla. The side view includes the robot's reference frame. We can see the ethernet cable, the OptoForce cable and the energy connection.

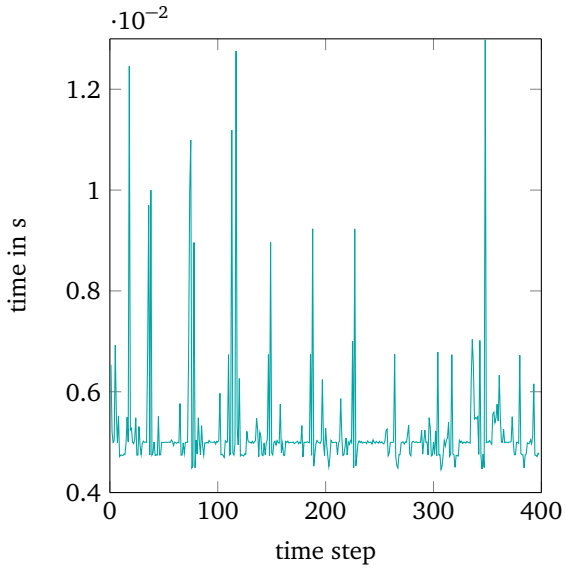


Figure 2.2.: Exemplary faulty communication between the remote computer and the Oncilla due to slow computations in Matlab, creating shaky movements of the robot.

2.2 Software for Robot Control

All computations for controlling the Oncilla are done on a remote computer using Matlab. Communication with the robot works with a `C++` server on the Oncilla and a `Java` client on the external computer via an ethernet cable. Sending a command to the Oncilla and receiving the answer is implemented as one single command for the Matlab interface and can take up to 3ms, the average value being about 2ms.
A command contains the desired motor positions for the two shoulder joints and the knee joint, thus it is a 12×1 vector. The return value delivers the actual motor positions as well as all available magnetic encoder values for the joint positions, resulting in a 24×1 vector.
The communication works on 200Hz, presenting a high challenge for Matlab, as working in real time is required to avoid irregularities in communication (see Figure 2.2 as example), which affects the smoothness of robot movements. To match the timing constraint, the model learning algorithm is written as `C++` `mex`-file and the receiving of force sensor data is modeled as `Java`-object which allows multithreading. This setup enables parallelization, of which Matlab alone is not capable.

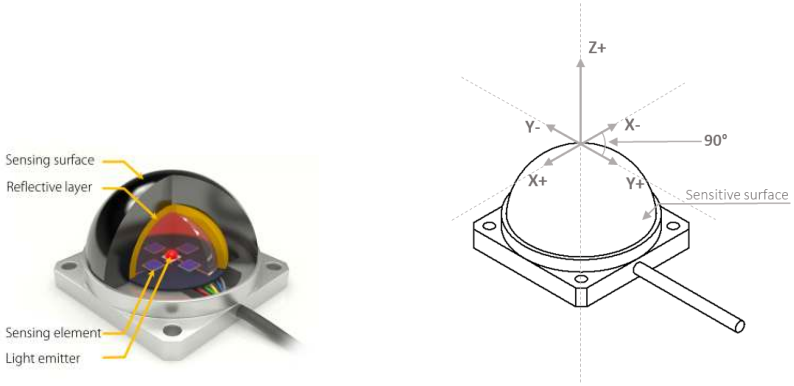
2.3 Additional Force Sensors

To measure the ground reaction force and thus the weight distribution among the feet, we worked with additional force sensors that were not originally included in the Oncilla. To validate their recordings, we further constructed a force plate.

2.3.1 OptoForce Sensors

We mounted OptoForce sensors [40] under each foot of the Oncilla. The diameter is about 2cm and the surface is a silicone half-dome, pictures can be seen in Figure 2.3. The form of the sensors approximates the Oncilla's feet as points. In each dome, four sensors measure the dome's deformation. Raw data is received on the remote computer via USB connection with average rate of 260Hz.
To record data, we first compute an offset from raw data when the sensors have no ground contact. The offset is then used to calibrate all further incoming data by subtracting it from the subsequent stream.

[8] both taken from `http://optoforce.com/3dsensor/`



(a) Construction **(b)** Coordinate system

Figure 2.3.: OptoForce sensor[8] views that show the functionality and the coordinate system of the OptoForce sensors. The sensors measure the deformation of the surface by using the reflections of emitted light.

As the sensors measure deformation rather than force, the values are very sensitive to the angle between the sensor and the ground: If the angle is not perpendicular, the F_z value is lower, despite having the same weight on the foot. Since we can neither control the lowest joint q_4 nor measure its angle, we cannot learn a mapping to obtain the actual force operating on the leg. So we combine the measured force values F_x, F_y and F_z into a single force F by using the Euclidean distance:

$$F = \sqrt{F_x^2 + F_y^2 + F_z^2}.$$

The sensors also have a high definition which makes them sensitive to noise. Thus, we applied smoothing to capture only the essential information. More details are described in Subsection A.5 in the appendix.

2.3.2 Force Plate

The loose joint q_4 and the sensitiviy of the OptoForce sensors let us doubt the reliability of the recorded data. To obtain more reliable data, we constructed a force plate using a precision force torque sensor, a picture is shown in Figure 2.4. It allowed us to validate the OptoForce sensor measurements and see how reliable their recordings are for our purpose. So the plate data serves as our ground truth data to which we compared the OptoForce sensor recordings. The force plate delivers data with 40Hz which we also smoothen as described in the appendix Subsection A.5.

2.4 Measuring the Center of Pressure

The sum of all forces that act between an object and its supporting surface works on a point called the center of pressure. Hence, for a standing quadruped it is a point between the ground contacts of the feet, determined by the amount of force applied on each contact point. So for a resting Oncilla, the CoP can be calculated as follows:

$$\mathbf{c} = \sum_{j=1}^{4} \frac{F_j \mathbf{e}_j}{\sum_{i=1}^{4} F_i},$$

where $\mathbf{e}_j = [x_j, y_j]^T$ are the endeffector coordinates of foot j and F_j the respective measured force. Calculating the CoP from the OptoForce data is straightforward, the sensors directly measure the force F_j (with $j = 1 \ldots 4$) applied at the contact points between each foot and the ground. The position of

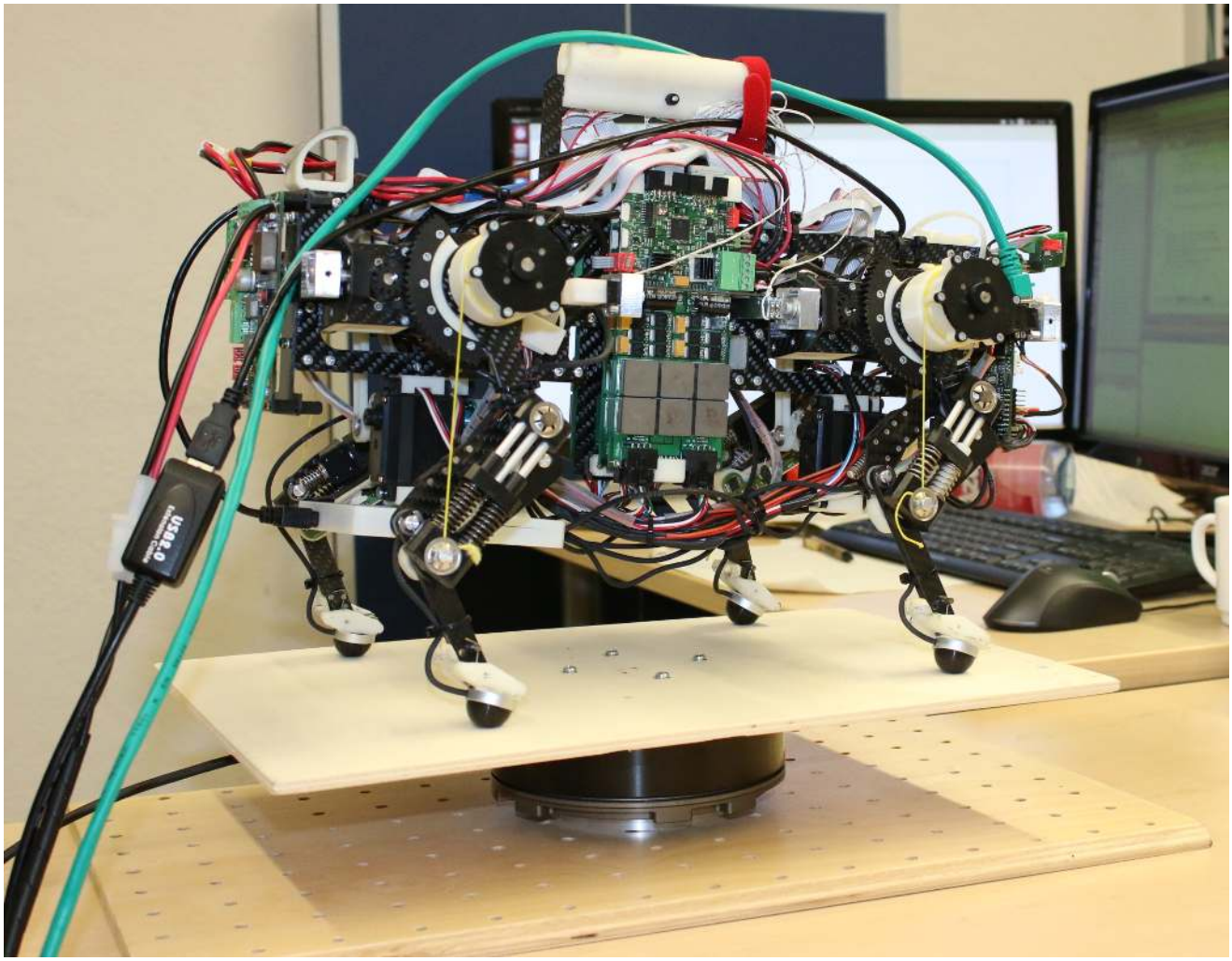


Figure 2.4.: The Oncilla stands on the force plate with OptoForce sensors mounted to the tips. The force plate uses a precision force torque sensor.

the feet, i.e., the endeffectors $\mathbf{e}_j$, are calculated by using simplified forward kinematics. We neglect compliance and double connection in the knee, the actual calculations can be found in the appendix Subsection A.1. The forward kinematics need magnetic encoder values for the joint positions $\mathbf{q}$, as the motor positions $\mathbf{p}$ do not capture enough information about the joints actuated by the cable mechanism (q_2 and q_3). We use this approach to compute the CoP throughout the whole presented work, which means as soon as the Oncilla moves, the CoP is only a rough estimation since we neglect forces created by moments and torques.

For the CoP calculation of the force plate, we always calculate the CoP relative to the center of the plate, thus the endeffector positions are assumed to be constant over the whole time. This assumption highlights a drawback of the force plate which can deliver reliable data only when the Oncilla is keeping its feet on the same position relative to the plate center, so we cannot use it for actual walking.

Figure 2.5 shows how the CoP trajectories generated from the force plate data and the OptoForce sensors look for different recordings. The shape of force plate and OptoForce trajectories are rather different and the OptoForce trajectories vary a lot from one recording to the next. In Figure 2.6 we can see that direction changes of the different trajectories happen analogously, such that both sensor modalities contain enough information to allow the computation of useful feedback.

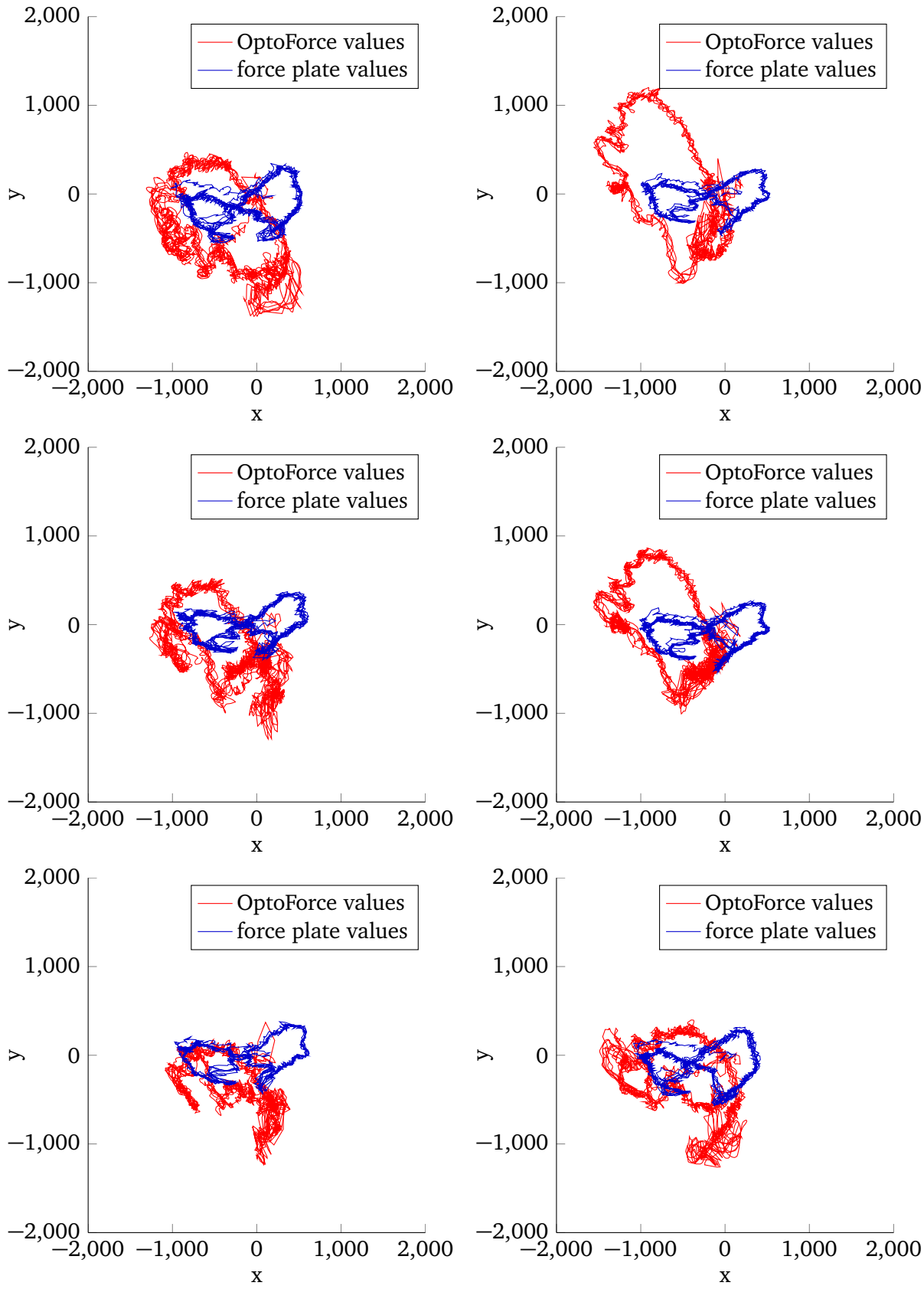


Figure 2.5.: The plots show the variations between CoP trajectories of the OptoForce sensors and the force plate. The OptoForce recordings differ from each other as each new ground placement changes the angles between force sensors and ground. The recordings also show that the weight of the Oncilla is not equally distributed, since despite using symmetric movement trajectories, the CoP trajectories are not symmetrical. The Oncilla's head points to the right.

2.5 Training Data Generation

To record training data, the Oncilla performs simple sideway swaying movements. They are symmetric movements as both knees on the same side receive the same command. First one side bends down, lifts up again, then the other side performs the same. We repeat this movement until enough data points are collected, which takes about 15s. One can find a performance of training data generation in the video [41].
For both sensors, OptoForce and force plate, the CoP trajectories vary over different placements of the robot (as can be seen in Figure 2.5). Every time we place the Oncilla on the ground, we have to record training data again: For the OptoForce sensors, the angle between sensor and ground changes every time and the legs lock in different positions, both because of the included compliance. Regarding the force plate, the position of the feet relative to the center of the plate changes with every replacement and therefore the recorded trajectory varies as well. Still, the force plate records more reliable inter trial data.

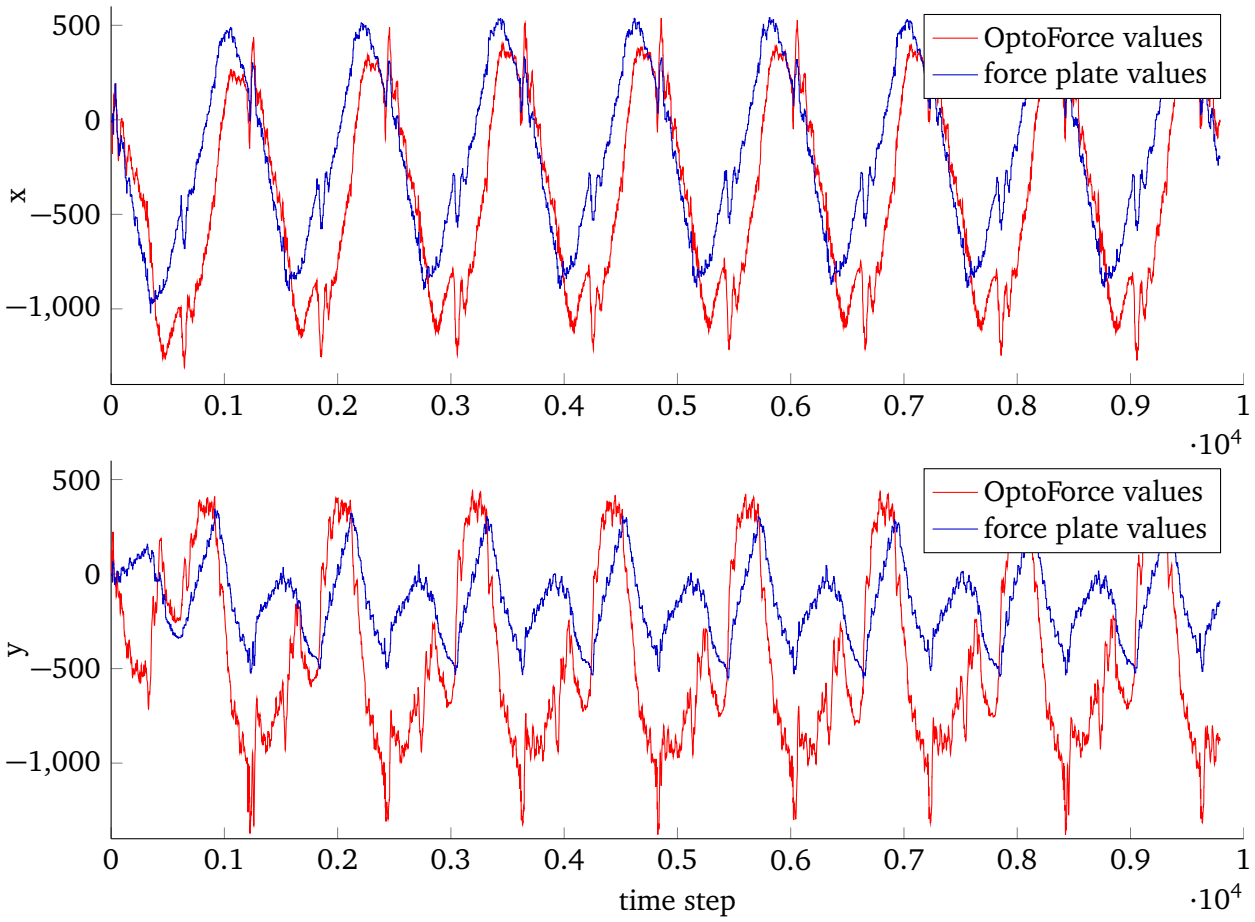


Figure 2.6.: The depicted CoP trajectories were recorded over time for both dimensions of the CoP. The comparison of the OptoForce sensor trajectories with the force plate trajectories shows that direction changes of the different trajectories happen analogously.

3 Control Strategies for Locomotion

For the feedforward command, we will use the idea of central pattern generators in the way that we create a rhythmic movement generator and modulate it using movement primitives. The feedback will rely on information about the center of pressure, a commonly used criterion for stability.

3.1 Balancing Controller

We want a simple balancing controller as feedback whose output works additive to the feedforward commands, which means it does not modulate the feedforward component. The feedback shall be based on the information gained from force sensors which we cumulate in the center of pressure. These thoughts lead to the general setup of

$$\mathbf{u}_{FB} = f_{FB}(\mathbf{c}).$$

As $\dim(\mathbf{u}_{FB}) > \dim(\mathbf{c})$, f_{FB} is unfortunately not a unique function, as (infinite) many joint positions result in the same CoP. Thus, we have the same problem as usually encountered in inverse kinematics, leading us to use the same approach to resolve it: We will learn a forward model f_c matching joints to CoP values. The Jacobian of this forward model will allow us to calculate approximate solutions for the inverse model. In more detail: Given the recorded training data, we want to learn a model that consumes the input vector $\Delta\mathbf{c} = [\Delta c_x, \Delta c_y]^T$ with $\Delta c_x = c_x^* - c_x$ and $\Delta c_y = c_y^* - c_y$ and predicts changes of the motor position command ($\Delta\mathbf{u}$).
To implement this idea, we learn a forward model f_c that uses the actual motor positions $\mathbf{p}$ and velocities $\dot{\mathbf{p}}$ to predict CoP coordinates $\tilde{\mathbf{c}}$, i.e. $\tilde{\mathbf{c}} = f_c([\mathbf{p}, \dot{\mathbf{p}}]^T)$. Subsequently, we use the Jacobian $\mathbf{J}$ of this forward model f_c to approximate the inverse function f_{FB}, similar to inverse kinematics solutions. Given the difference $\Delta\mathbf{c}$ between a desired CoP $\mathbf{c}^*$ and the actual current CoP $\mathbf{c}$, the inverse approximation calculates $\Delta\mathbf{u}$, a change for the feedforward command. A depiction is shown in Figure 3.1.

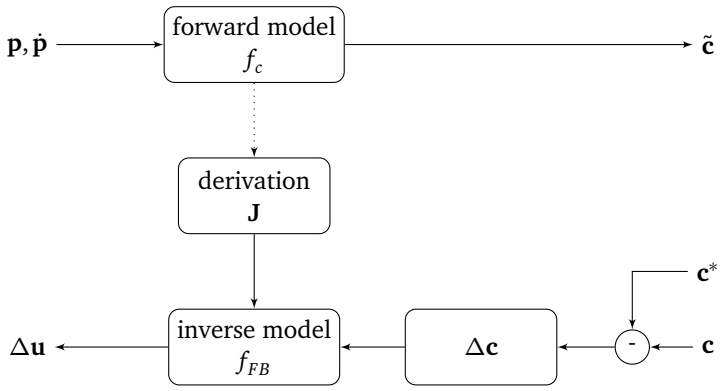


Figure 3.1.: To calculate the feedback update, we use a linear forward model f_c that maps the motor position $\mathbf{p}$ and velocity $\dot{\mathbf{p}}$ to the CoP $\tilde{\mathbf{c}}$. The derivation of this forward model is used for an inverse model f_{FB}. It takes the difference between the actual CoP $\mathbf{c}$ and a desired CoP $\mathbf{c}^*$ and predicts an update for the feedback command $\Delta\mathbf{u}$.

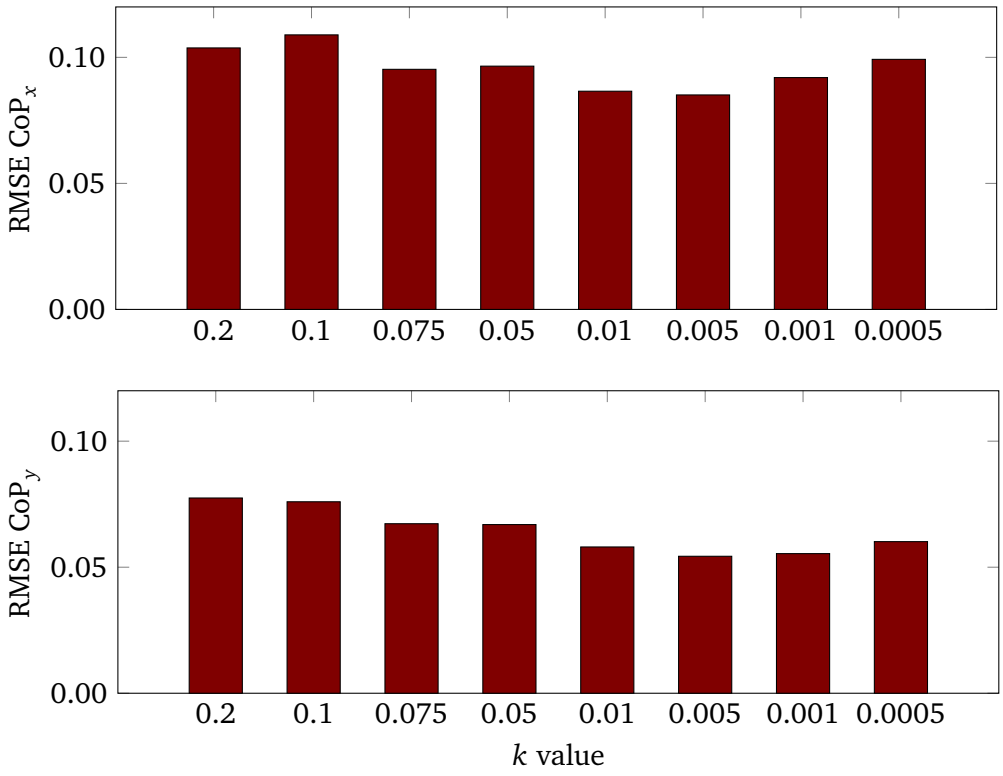


Figure 3.2.: RMSE as percentage of data range is computed depending on the chosen k. As the RMSE is lowest at $k = 0.005$ for both CoP dimensions, this value seems to work best for our model. For all depicted results, $n = 2000$.

3.1.1 Learning a Forward Model

In model learning, we approximate a function f_c that maps input data $\mathbf{X}$ to output data $\mathbf{Y}$. In our case, input parameters for the forward direction are the motor positions $\mathbf{p}$ and output labels are the CoP coordinates c_x and c_y. Recordings of random Oncilla movements indicated a non trivial relation within the data, thus we decided to use Locally Weighted Regression (LWR).
LWR has the advantage of not fitting a global solution but a linear local one to data points close to the current input $\mathbf{r}$. According to Atkeson (1997)[42], a general setup for LWR is to calculate the scalar weight w_i for each data point $\mathbf{x}_i$ by applying a kernel function $K(\cdot)$ to the weighted distance d between input query $\mathbf{r}$ and data point $\mathbf{x}_i$

$$w_i = K(d(\mathbf{x}_i, \mathbf{r})).$$

Weighting the training data leads to $\mathbf{z}_i = w_i \mathbf{x}_i$ as entries of $\mathbf{Z}$ and $\mathbf{v}_i = w_i \mathbf{y}_i$ as entries of $\mathbf{V}$. To predict the output label $y(\mathbf{r})$, linear regression is used on the weighted training data

$$y(\mathbf{r}) = \mathbf{r}^T (\mathbf{Z}^T \mathbf{Z})^{-1} \mathbf{Z}^T \mathbf{v}.$$

In our LWR implementation we use a squared exponential kernel for $K(\cdot)$ and simple subtraction for the distance function d. So for calculating the weights we yield

$$w_i = \exp((\mathbf{x}_i - \mathbf{r}) h (\mathbf{x}_i - \mathbf{r})^T).$$

Two open parameters have to be set, i.e. the bandwidth k and the number of training samples n. For evaluating the forward model and tuning the parameters, we tried a set of different reasonable parameters and calculated the Root Mean Squared Error (RMSE) to see how well the predictions matched the

actual label. Also, we experimented with the approximated velocity $\dot{\mathbf{p}}$ as additional input, which showed to be useful for creating a better LWR model (see Appendix A.3), probably as it indicates direction of movement and the CoP trajectory varies depending on that, especially for the OptoForce recordings as can be seen in Figure 2.5.
In Figure A.3, one can see the tendency for an increasing parameter n reducing the RMSE, thus a value as high as possible is preferred. Unfortunately, Figure A.4 shows the calculation time of LWR for a single two dimensional CoP prediction and thus a backdraw of LWR: There is a tradeoff between the amount of training data and precision in predictions. As we control the Oncilla with 200Hz, all computations and the communication have to be done within 5ms, limiting us to $n = 2000$ training data points.
With n fixed, we simply evaluated parameters in the range from 0.2 to 0.0005 for the bandwidth k. The results, depicted in Figure 3.2, let us set $k = 0.005$.

3.1.2 Jacobian

Many approaches for inverse kinematics require the derivation of the forward model. As LWR provides a local linear forward model

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b},$$

the derivation of the Jacobian is simply $\mathbf{A}$. To evaluate the LWR parameters regarding the quality of the Jacobian, we use the previous CoP $\mathbf{c}_{t-1}$, the current Jacobian $\mathbf{J}_t$ and the difference between current and previous state to calculate the current CoP $\tilde{\mathbf{c}}_t$ and compare it with the actual current CoP $\mathbf{c}$ to asses the quality of the prediction:

$$\tilde{\mathbf{c}}_t = \mathbf{c}_{t-1} + \mathbf{J}_t([\mathbf{p}_t, \dot{\mathbf{p}}_t]^T - [\mathbf{p}_{t-1}, \dot{\mathbf{p}}_{t-1}]^T).$$

Some additional smoothing gives satisfying results for the Jacobian predictions, as can be seen in Figure 3.3.

3.1.3 Inverse of Forward Model

Two common approximations for inverse kinematics are Jacobian transpose and pseudoinverse. The Jacobian transpose approach with feedback update

$$\Delta\mathbf{u} = \beta\mathbf{J}^T\Delta\mathbf{c}$$

is a rough approximation. Still, for a sufficient small stepsize β it reduces the difference between the desired CoP $\mathbf{c}^*$ and the current CoP $\mathbf{c}$, $\Delta\mathbf{c} = \mathbf{c}^* - \mathbf{c}$, thus it brings us closer to our desired position $\mathbf{c}^*$.
Using the pseudoinverse to calculate the feedback update

$$\Delta\mathbf{u} = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\Delta\mathbf{c}$$

is a more precise and hence the preferred approach, but it tends to have stability problems. To avoid singularities, we use a damped pseudoinverse:

$$\Delta\mathbf{u} = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T + \lambda\mathbf{I})^{-1}\Delta\mathbf{c}.$$

We can benefit from the higher precision as we calculate only one step of the inverse kinematics approach before we execute feedback directly on Oncilla. Additionally, the pseudoinverse allows for null space constraints which allow to specify e.g. joint limits in the future.
The two inverse possibilities are compared to each other in the simulated balancing controller evaluation in Subsubsection 4.1.1.

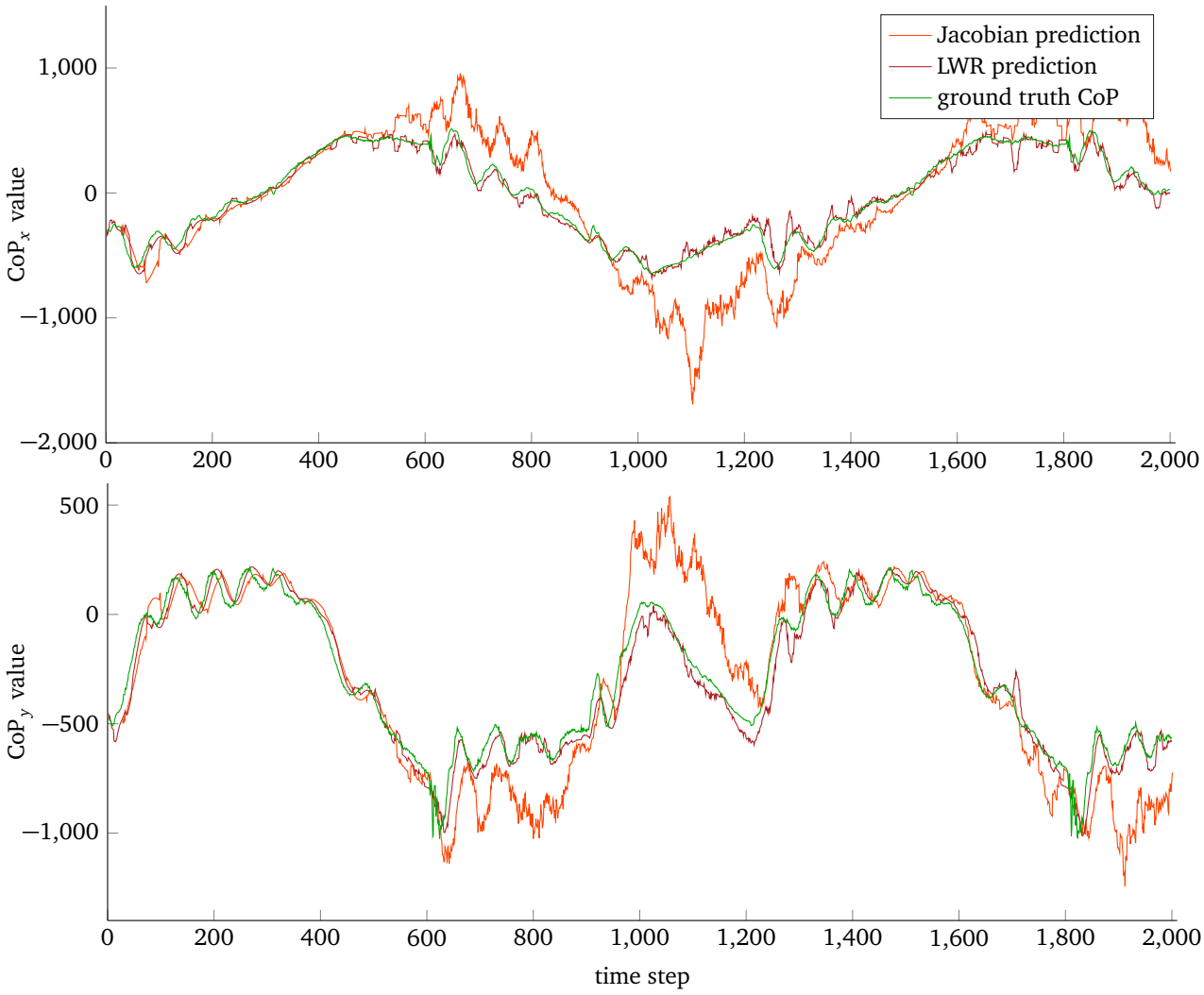


Figure 3.3.: Results for Jacobian prediction with $k = 0.005$, compared with the actual CoP and the predictions of the forward model. Smoothing is required to yield satisfying results when using the Jacobian.

3.2 Walking Gait Generator

Current research indicates that in animals, CPGs create a rhythmic movement from a simple linear input. For our control system, we want to use such a CPG for generating the feedforward command.
Thus, we need to create a rhythmic gait pattern. A desirable feature of such a gait is its dependency on as few parameters as possible.
Approaches for gait generation range from using simple sinusodial functions to more elaborated approaches like a mixture of oscillating functions, which exist in various forms, such as a mixture of Hopf-oscillators discussed by Degallier [11].
We use different theoretical approaches on generating a forward gait: a hand-tuned version with sinusodials and a learned movement primitive based on the hand-tuned gait. In the final discussion (Subsubsection 5.3.3), the movement primitive approach is extended to a stochastic model, where only theoretical ideas are presented. Real robot experiments are part of future work.

3.2.1 Gait Properties

The different gaits used by animals to locomote, for example trot, walk and gallop of horses, can be classified into different categories: Gaits are called symmetrical when right and left leg of a pair move alternatingly, while they move at the same time in antisymmetrical gaits. The categories of static and dynamic gaits have alreade been introduced.
To describe the actual pattern of a gait, one usually distincts between legs being in stance phase or in swing phase. One also denotes the events switching the phase, for instance, lifting and placing the legs. A full walking cycle includes the two phases of all legs, thus a quadruped gait covers eight events, lifting and placement of all four feet.
Hildebrand [43] uses the duty factor and a phase offset for each leg to mathematically describe gait patterns, reducing the amount of specification parameters. Some examples are illustrated in Figure 3.4. The duty factor is calculated as the percentage of stance phase of the whole walking cycle by

$$\text{duty factor} = \text{stance}/(\text{stance} + \text{swing}).$$

If a quadruped's gait has a duty factor larger than 50%, it is called walk, and run in the other case. For a static stable gait of a quadruped, we need at least 75% stance phase, as at all times during the whole cycle at least three legs have to be on the ground to give static stabilization. Such a gait is called crawling or creeping gait.
The phase offset denotes when either the foot placement or lifting happens relative to that event of a chosen reference leg (usually the left hind leg, LH). Only one of the two possible events is required, as the other one can be calculated given the phase offset and the duty factor.
The difficulty of a creeping gait is that all legs have a different phase, while for example in trot, always two legs (the diagonal opposite ones) share the same phase, reducing the amount of distinct timepoints with occuring events.
McGhee [2] examined the stability of the theoretically possible gaits resulting from permuting the leg event order of creeping gaits, as not all leg event orders result in stable gaits. A common leg movement order that is actually observed in animals (for example in horses) is LH, LF, RH and then RF, which is the one we used for the implemented gaits.

3.2.2 Hand-tuned Mixture of Sinusodials

One of the simplest approaches to create a gait is a mixture of different sine curves. Each of those sinusodial curves has at least two parameters, center point and amplitude. To allow a larger variation of

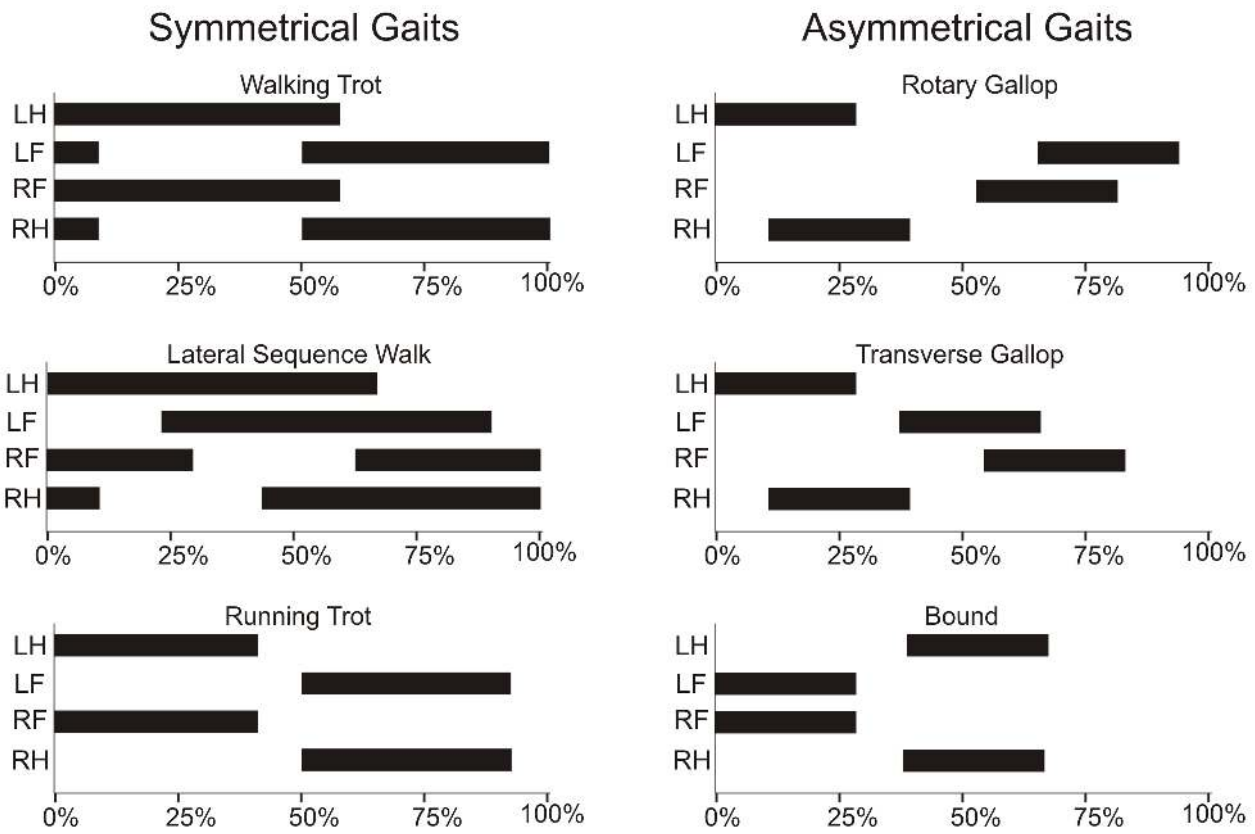


Figure 3.4.: The figure shows several gaits described according to Hildebrand's mathematical gait definition[9]. The black parts mark the respective foot being in contact with the ground. The depicted gaits are not statically stable.

shapes, we also introduced a start and an end point for each sine function, allowing the scaling of the rise and the decline phase independently.
The shape of the resulting trajectory τ_h is based on results from Sproewitz et al. [39], as they were obtained using a similar leg construction as the one of the Oncilla. Sproewitz et al. show that gaits having a double knee-peak allow faster locomotion, with one peak during swing to shorten the leg such that it lifts off the ground, and another one during stance, supporting the weight balancing. To use it as a creeping gait, we re-scaled the phases such that each leg has 75% stance phase, as we are aiming at a static walking gait. The parameters for this approach were hand-tuned and the results are discussed in Subsubsection 4.2.1, including further adjustments and refinements.

3.2.3 Mixture of Von Mises Basis Functions

The chosen sine approach resulted in a large amount of nonlinear parameters. This nonlinearity turns parameter tuning into a hard task. Thus, a linear representation of the handcrafted trajectory is preferred.
Movement primitives (MPs) have this appealing feature, as well as being a compact parametric representation of a (movement) trajectory τ satisfying our other requirement for our gait generator. MPs have been introduced by Ijspeert [44]. A walking gait trajectory τ is represented by values y_z for each phase step z along a walking cycle, thus $\tau = y_{1:Z}$. Each trajectory point y_z is specified by a linear basis function model $\mathbf{\Psi}$, i.e. $y_z = \mathbf{\Psi}_z^T \mathbf{w}$ with $\mathbf{w}$ weighting the basis functions and thus being the compact representation of the trajectory.

[9] taken from `https://upload.wikimedia.org/wikipedia/commons/c/cf/Gait_graphs.jpg`

Depending on the chosen basis functions, MPs can model stroke based or rhythmic movements. Since for a gait we desire the latter, we define $\mathbf{\Psi}$ as a matrix of von Mises basis functions ψ_i^{VM} with

$$\psi_i^{VM}(z) = \exp\left(\frac{\cos(2\pi(z - c_i))}{h}\right).$$

The Oncilla is a position controlled robot and as we directly model a position trajectory, we do not need differential equations allowing us to integrate positions from velocities or acceleration. Thus, we have a simple von Mises mixture model representing the rhythmic pattern with only a few parameters, namely the weights $\mathbf{w}$.
Those model parameters are calculated using the handcrafted trajectory τ_h from Subsubsection 3.2.2 as labels $Y = y_{1:Z}$ for a damped regression such that

$$\mathbf{w} = (\mathbf{\Psi}^T\mathbf{\Psi} + \alpha\mathbf{I})^{-1}\mathbf{\Psi}^T Y.$$

The linearity of the parameters is another benefit of MPs. The output is less sensitive to parameter changes compared to nonlinear models and more stable, a useful feature when experimenting directly on hardware.
In this representation, two hyperparameters have to be tuned manually: the number of basis functions n and the bandwidth h. The evaluation of parameter tuning is discussed in Subsubsection 4.2.2.

3.3 Feedforward Controller with Tactile Feedback Error Correction

To create a more robust and thus a more versatile controller, we combine the balancing controller and the gait generator. The gait pattern serves as straight feedforward command, while the balancing controller will add feedback to stabilize the movements. The percentage of added feedback is weighted by ω. So each command $\mathbf{u}$ is calculated as

$$\mathbf{u} = (1 - \omega)\mathbf{u}_{FF} + \omega\mathbf{u}_{FB},$$

with the feedforward command $\mathbf{u}_{FF}$, the feedback command $\mathbf{u}_{FB}$ and a scalar weighting ω. The theoretical setup is depticted in Figure 3.5.
To set the weight, a first step is the hand-tuning of ω to a stable solution. An alternative approach is learning the weights together with the gait parameters. The most advanced feedback weighting is an adaptive ω that changes according to the variance of a stochastic feedforward command. This stochastic feedforward command is encoded in a distribution over trajectories. A small variance suggests that we can have a strong belief in the command and thus need little or no feedback. The weighting ω may also depend on the reliability of the feedback itself, as for example a bad prediction of the CoP by LWR indicates that the Jacobian is also less reliable, thus the predicted feedback may not be helpful. Those ideas are also discussed in Subsubsection 5.3.1.

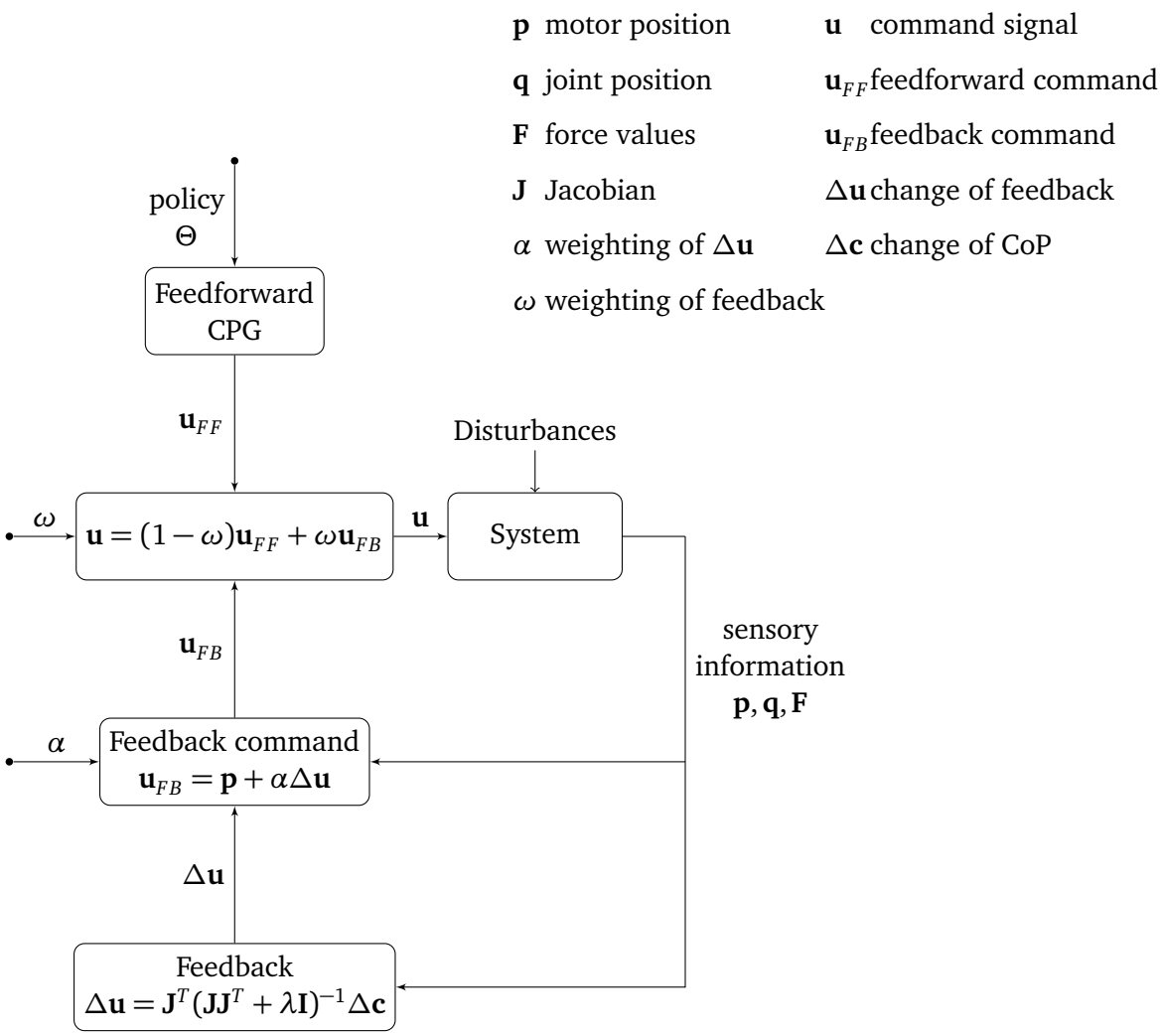


Figure 3.5.: The diagram shows our concept of a feedforward controller with tactile feedback error correction. The parameters of the gait Θ determine the commands $\mathbf{u}_{FF}$ generated from the feedforward component (the gait pattern). Based on the current joint positions $\mathbf{q}$ and the measured force values $\mathbf{F}$ of the system, the required motor position change $\Delta\mathbf{u}$ is computed using the inverse model. This feedback is needed to bring the CoP of the system to a desired position. It is further weighted by a stepsize α and added to the current motor positions $\mathbf{p}$ of the system. The resulting feedback command $\mathbf{u}_{FB}$ of the balancing controller is combined with the feedforward command $\mathbf{u}_{FF}$ using a weighted sum controlled by factor ω.

4 Experiments and Results

Both of the components introduced in the previous section, the feedforward walking gait and the balancing controller, were tested individually before we finally combined them to achieve a full system setup.

4.1 Feedback Generation and Model Learning

Before applying feedback directly to the robot, we compared the two possible approaches, pseudoinverse and Jacobian transpose, by evaluating them on recorded data. Afterwards we only used the pseudoinverse on the Oncilla.
As the recorded data includes only symmetric movements of the knees, we simplified the feedback calculation as well. We calculate only two feedback values, one for the left and one for the right side, and apply them to the respective knees.

4.1.1 Evaluation of Inverse Model in Simulation

For a simple simulated evaluation of the feedback controller, we used the calculated command as regularization. Thus, we start with a point from the training trajectory where the Oncilla stands in a one-sided bent position. As we have no force data in simulation, we use the prediction of LWR as real CoP. Also we have no feedforward command, so the feedback weight ω is set to 1, thus, the next command is simply $\mathbf{u}_{FB}$, i.e.,

$$\mathbf{u} = \mathbf{u}_{FB} = \mathbf{p} + \Delta\mathbf{u},$$

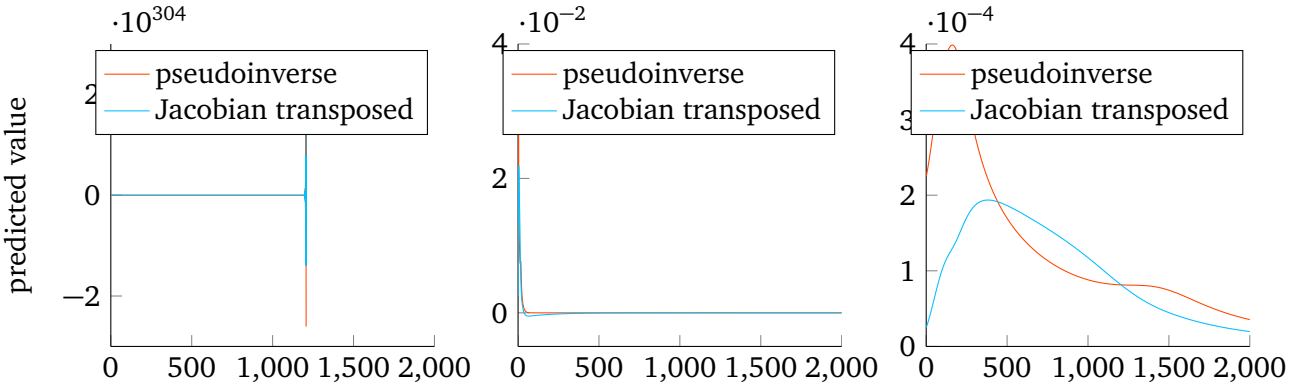
thus it is completely determined by feedback. As we only use recorded training data, $\mathbf{p}$ is the command $\mathbf{u}$ from the previous time step in this case.
For such a setup, the values for CoP prediction and feedback simply diverge up to infinity for the Jacobian transpose as well as for the pseudoinverse. The pseudoinverse also has stability issues, as $\mathbf{p}$ reaches values far away from training data points of LWR. This circumstance makes the weights very small, which results in matrix entries close to zeros when calculating the pseudoinverse.
So we regularize the feedback by interpolating linearly to $\mathbf{c}^*$ and taking only a small step with size σ along this direction when computing the feedback update

$$\Delta\mathbf{u} = \mathbf{J}^{\ddagger}\sigma(\mathbf{c}^* - \mathbf{c}).$$

Due to the linearity, weighting the interpolation has the same effect as weighting the amount of feedback change applied to the current joints with α when calculating the command

$$\mathbf{u} = \mathbf{p} + \mathbf{J}^{\ddagger}\sigma\Delta\mathbf{c} = \mathbf{p} + \alpha\mathbf{J}^{\ddagger}\Delta\mathbf{c}.$$

With a small stepsize $\sigma = 1/10$, both methods converge, and no stability problems with the pseudoinverse are observed.
For a similar behavior of the pseudoinverse and the Jacobian transpose approaches, the weight β for the Jacobian transpose has to be set to the small value of 0.00001. Examples for diverging, converging and slowly converging results can be seen in Figure 4.1.
Both methods show oscillations appearing at the beginning of feedback application. The oscillations decline with a smaller stepsize σ, but a smaller stepsize also lowers the rate of convergence. For example, with a stepsize of $1/500$ the feedback does not converge within 2000 test steps, which is equal to 10 seconds. As both methods show similar behavior and the pseudoinverse has no stability problems, we used the pseudoinverse for the remaining experiments.



(a) stepsize $\sigma = 1/3$ **(b)** stepsize $\sigma = 1/10$ **(c)** stepsize $\sigma = 1/5000$

Figure 4.1.: The divergence and convergence of feedback depends on the stepsize σ. Figure a) gives an example for divergence of the two inverse kinematics approaches due to a too large stepsize σ. As it can be seen in the other two examples, both approaches converge with a smaller stepsize. The smaller the stepsize, the more time steps the convergence takes.

4.1.2 Practical Evaluation of the Balancing Controller

For a practical evaluation of the feedback controller the real robot is used. Initially, the Oncilla is in a one-sided bent position and when the feedback controller starts. As in the simulation, no feedforward commands are applied. Thus the next command is always the previous one plus the calculated feedback change. The feedback is not filtered and but is bounded to protect the joints (it still allows jumps up to 30% of the joint movement range). As we stated already, we use only the pseudoinverse $\mathbf{J}^{\dagger}$ for practical evaluation.
With such a setup, the Oncilla shows two different typical behaviors: In case the stepsize σ is too large, the movements start to oscillate. In case the stepsize is too small, the Oncilla gets stuck and barely moves itself.

Using the Force Plate for Feedback Generation

Using the self constructed force plate worked quite directly out of the box: Hand-tuning of the stepsize σ showed stable solutions with values from $1/500$ to $1/700$, which means that for this parameter range, the Oncilla brings its trunk back to a horizontal position. As the values are rather small compared to the ones used in simulation, the small oscillation at the beginning is not noticeable anymore.
Additionally it is possible to push the Oncilla slightly, getting a response by the Oncilla creating contra tension. Also one can force reactions by applying force directly to the force plate (See video [45] for a short demonstration). As the applied feedback is symmetric, the Oncilla can only correct forces working along the y-dimension of the Oncilla frame.
Thus, a proof of concept is done for the feedback controller, as the Oncilla can (re-)balance itself.

Using the OptoForce Sensors for Feedback Generation

Switching from the force plate to the OptoForce sensors worked less intuitively. One problem was that the force plate started to vibrate when the Oncilla moved, causing too much noise in the OptoForce sensors. Thus, we performed experiments using only one of the two sensors, either force plate or OptoForce sensors.
Another problem was rooted in filtering: As the CoP values calculated from OptoForce sensors are quite noisy, we tended to filter them stronger than the force plate. Such strong filtering created a delay in

the CoP position that was large enough to let the Oncilla oscillate, as it overshot the desired position constantly.
After solving these two problems through extensive parameter tuning, the Oncilla was able to lift itself up to a balanced position and also reacted to pushes. As the OptoForce sensors work sufficiently and also deliver data that allows us to validate our feedback controller concept, we will solely use the OptoForce sensors for all following experiments. They allow for actual locomotion and are closer to the idea of an autonomous walking robot.
The next step in evaluating the balancing controller was to add a constant feedforward signal $\mathbf{u}_{FF}$. We lowered ω to values between 0.5 and 0.1 to have at least 50% feedforward and tried to find a stable parameter setting in this range for computing the commands

$$\mathbf{u} = (1-\omega)\mathbf{u}_{FF} + \omega\mathbf{u}_{FB} = (1-\omega)\mathbf{u}_{FF} + \omega(\mathbf{p} + \alpha\mathbf{J}^{\dagger}\Delta c).$$

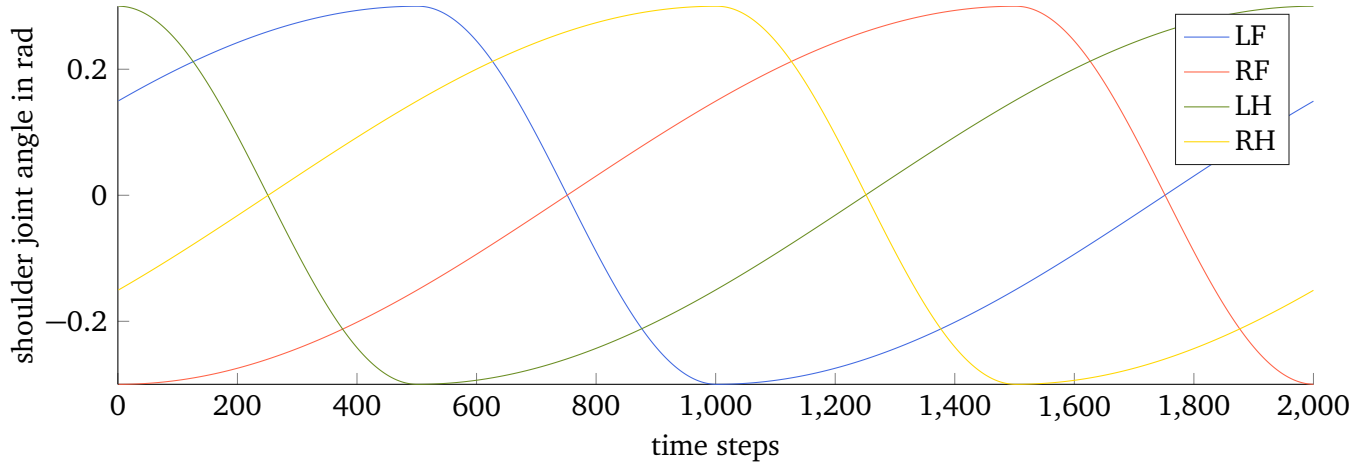
In this setup, it was not possible to tune the amount of added feedback controlled by α to gain a stable solution. The knee motors switched from no reaction to oscillation immediately without a working solution inbetween (in the range of $\alpha = 1/15$ to $1/100$). During the oscillation, we also noticed some friction in the leg springs, holding the Oncilla back from quick leg stretching as well as making small changes induced by feedback less precise, leading to delayed and thus irregular changes of the knee joint position.

4.2 Gait Generation

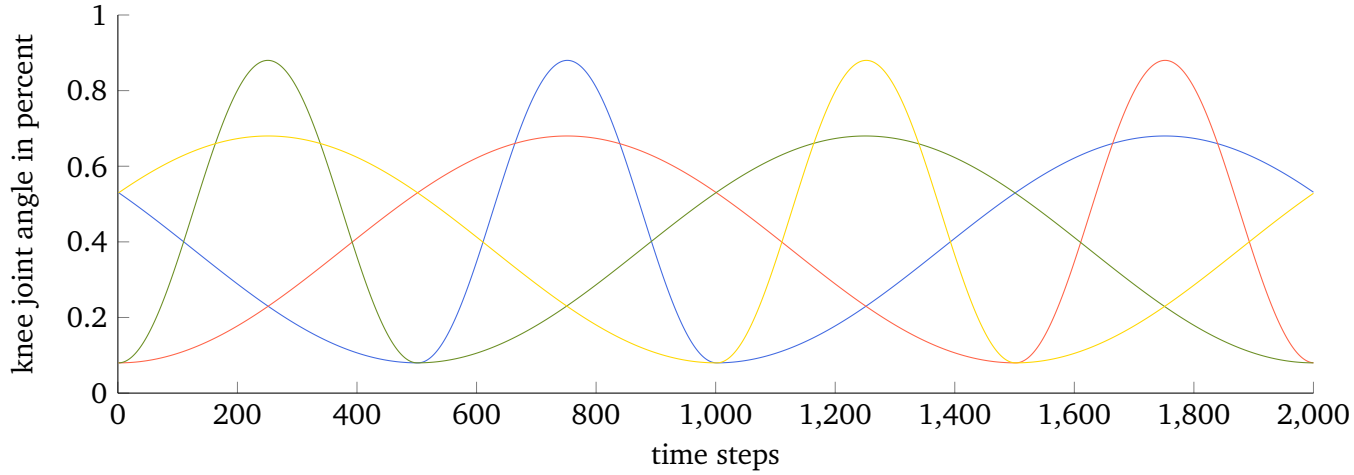
Tuning gaits has to be done very carefully, as the Oncilla cannot stumble and fall. In case of the leg movements not being perfectly adjusted to each other, they exert twist movements to the Oncilla. Such motions are dangerous for the plastic parts and the joints.

4.2.1 Handcrafting a Stable Gait

The basic ideas for a handcrafted gait were that the shoulder joint in the leg has to move backwards in the stance phase and forwards during the swing phase. The implemented trajectories can be seen in Figure 4.2a.
The knee has to flex during the swing phase to lift the foot off the ground and then again to flex during the stance phase for balancing the robot. So we used the example trajectory presented in [39], Figure 3, as reference. We changed the shape of the gait to the one of a creeping gait by stretching the stance and shortening percentage of the swing phase (25% swing and 75% stance phase to be able to constantly rely on static stability). The resulting knee trajectory is shown in Figure 4.2b.
This simple approach does not provide satisfying results, despite trying different ways of modulating the contraction during stance phase. The hind legs were able to move to the front, but without ever actually leaving the ground. The front legs were stuck to the ground and blocked forward movement, which sometimes even resulted in shifting the Oncilla backwards. An example recording can be seen in video [46].
Taking a closer look at how the trajectories of the legs are combined, one can see that when the hind legs lift, the diagonal front leg is bent and the other two legs have the same amount of contraction, leading to a weight shift to the contracted front leg, which makes it possible to actually move the hind leg. For the front legs, the hind leg on the same side is contracted and both legs on the other side have the same length, resulting in too much weight on the front leg such that it is not able to move forwards.
The next, more elaborated control variant varies sine functions more precisely and uses them only partially. The different sine curve parts can be specified by setting their start points, peak points, end points, as well as their amplitudes, which allows to specify the gait events such as foot lifting. As we have a large



(a) shoulder trajectory



(b) knee trajectory

Figure 4.2.: In the first simple sine approach, each shoulder trajectory is a sine curve that is clinched such that bringing the foot to the front (negative joint angle values) in the swing phase takes 25% of the walking cycle. The knee trajectories contain two different sine function parts, one that contracts the leg to 90% of what is possible for the swing phase and one that contracts the leg to 65% during the stance phase. All knees have the same trajectory.

number of parameters, the used number of sine curves was difficult to tune. Setting the values had to be done carefully, as inadequate trajectories easily result in very unstable and for the Oncilla dangerous situations such as falling, which endangers the hardware.
With this approach, the Oncilla was at first still not able to lift its feet visibly off the ground. To solve the problem, we investigated the actual static behavior of the weight shifting by letting a stable standing Oncilla move slowly to distinct positions. The specifical movements revealed that a leg has to contract 90% to allow the lifting of the diagonal opposite front leg. Additionally, it is important to stretch the leg that shall be lifted before knee contraction, to actually shift the weight to the contracted diagonal opposite stance leg. Also the shoulder angle of the swing leg is important: If it is already too forward when lifting the leg, the foot shuffels on the ground, if it is too far behind when the knee stretches, the foot has early ground contact and pushes the Oncilla backwards.
So we changed the trajectories to have the required 90% contraction during stance phase, having this contraction already before the swing leg starts to lift and stretching the leg before the swing leg touches the ground again. We also adapted the shoulder trajectories, such that the forward movement in the swing phase is shorter than the actual swing phase, as flexing and stretching of the leg should happen when the leg is not moving to the front. The resulting trajectories are depicted in Figure 4.3. In contrast to the simple first approach, not all legs have the same trajectory: The trajectories of the hind legs differ from the ones of the front legs which comes from the order in which the legs enter the swing phase.
The resulting walking gait brings the Oncilla forward and three of the four legs have an actual in-air swing trajectory. Unfortunately, the huge contraction of the knees during stance phase results in a rather shaky gait with extreme movements. A demonstration can be seen in video [47] and some pictures are shown in Figure 4.4.
In the video, one still observes a few problems which are further discussed at the end of this thesis. Despite symmetric trajectories, only three legs have a continuous swing phase, the left hind leg falls back on itself when it is lifted. Also, the Oncilla often does not walk in a straight line, but tends to have a slight curve when walking several steps. Another observed problem is a backwards sliding of the feet due to which the Oncilla moves forward less than it actually should. Nevertheless we constructed a basic gait that actually brings the Oncilla forward and enables further experiments regarding gaits.

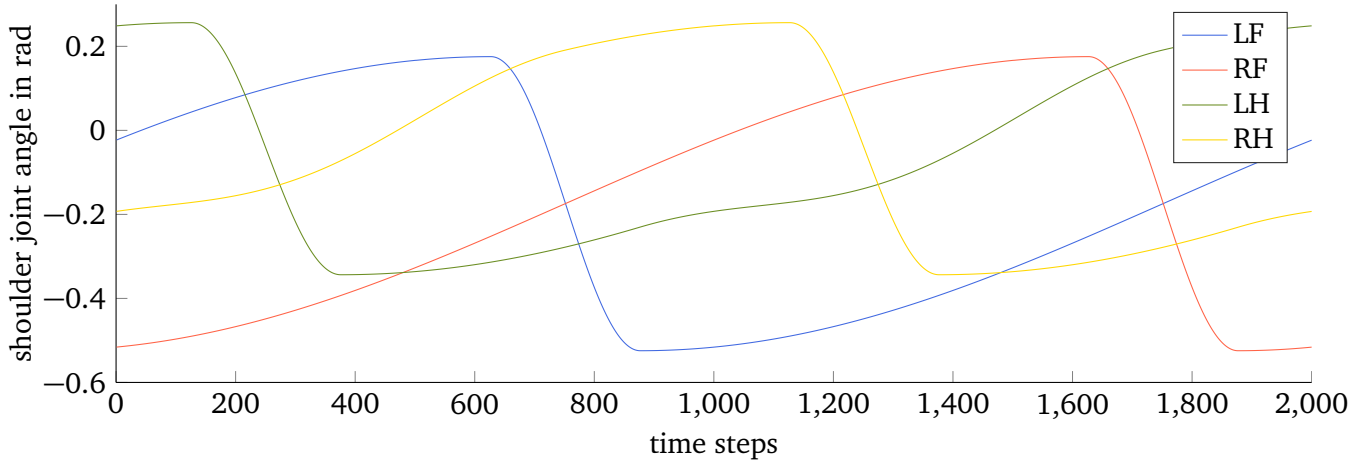
4.2.2 Modulating a Stable Gait

To have a linear and also a more compact representation of the handcrafted model, we used von Mises basis functions and learned the weights of the basis functions as described in Subsubsection 3.2.3. The number of basis functions was hand-tuned. For the solutions depicted in Figure 4.5, the shoulder trajectory is encoded by 12 basis functions and for the knee trajectories we used 16 basis functions. As it mimics the hand-tuned trajectory, the mixture model results in almost the same walking behavior as can be seen in the video [48].

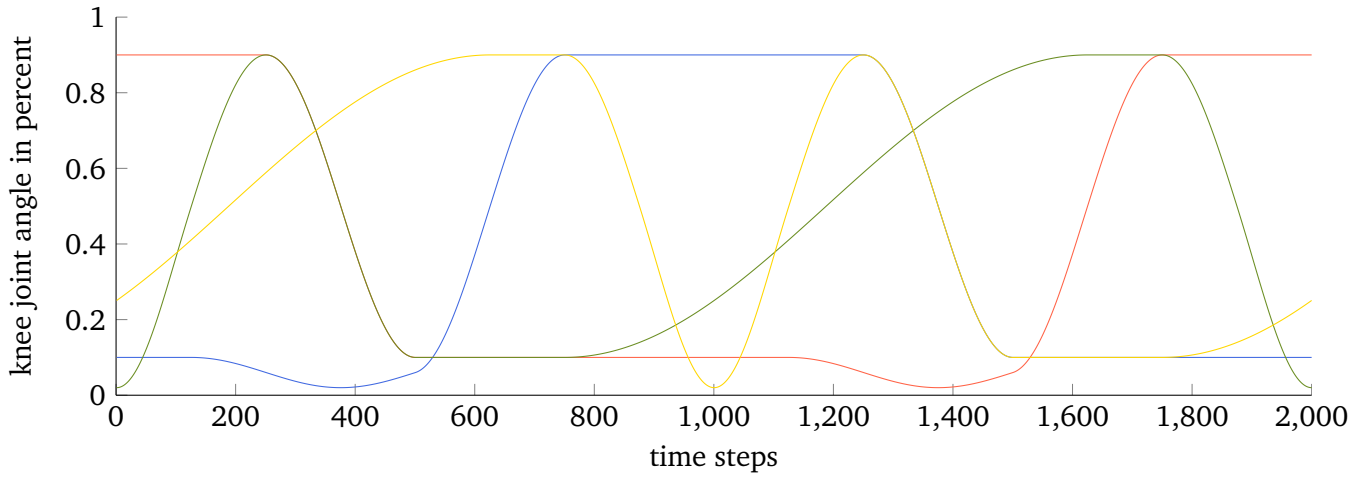
4.3 Combining Feedback Controller and Feedforward Gait

With a working gait and balancing controller, we now want to evaluate if the balancing controller can improve the stability of the gait or simplify the parameter tuning. As the walking gait itself is already a stable gait, the evaluation of the effect of the feedback controller is done qualitatively by observing the gait and quantitatively by evaluating the recorded CoP trajectories and joint trajectories.
The first very naive approach is to simply combine the constructed balancing controller and the feedforward walking gait. Figure 4.6 shows how the recordings of CoP trajectory differ from the CoPs that were recorded during actual walking.
The feedback applied in the beginning was very shaky and thus dangerous for the joints, an example of which can be see in the video [49]. Hence, we filtered the predicted commands by using up to 100

previous time steps. Otherwise, an evaluation of the stepsize α would not have been possible for the range where the feedback actually had an influence on the gait.
The problem with additive feedback is that we have to lower the actual feedforward commands, because feedforward and feedback both give us valid positions, a sum of those values does not result in a valid position anymore. So we weighted the feedback and the feedforward commands with ω and $1-\omega$, respectively. This means that effectively, we do not execute 100% of the feedforward gait. For example, with an ω set to 1/10, only 90% of the gait are executed. The execution of this example is shown in the video [50]. The feet barely leave the ground during the swing phase. The video should be seen as a comparison to the video [51] where we add feedback on top of the 90% executed gait. For learning the LWR model, the question is which of those two possbilities should be used as training data. We tried both possibilities in a short evaluation and could not find significant differences regarding the resulting gait.
Applying the feedback with the current setup does not yet give any noticeable improvement to the gait when observing the performed gait itself. But the recorded joint data shows that the created feedback actually changed the joint positions of the gait. Figure 4.7 shows how the trajectories of the original hand-tuned gait (purple) look in comparison to the feedforward component (turquoise), which is the same trajectory performed with only 90%, and two recordings from a full walking system including feedforward and feedback (red). The latter is depicted twice to give a notion of how much the feedback can differ from one recording to the next. So we can see that in general, applied feedback leads to the knees being more flexed during the whole time. The data also shows that the data recorded with 100% feedforward gait is closer to the one seen during feedback application, and should thus give better training data for the LWR model. To improve the performance of the executed gait visibly, the feedback should be modified, which is discussed in the Subsubsection 5.3.1.



(a) shoulder trajectory



(b) knee trajectory

Figure 4.3.: For the improved sine function approach, several overlapping, scaled sine curves form trajectories for shoulders and knees. In comparison to the first approach, the shoulder uses a smaller fraction of the cycle to move to the front (negative joint angle values). The hind leg trajectories also have small bumps during the stance phase, an attempt to counteract the backwards movement of the foot which results from contracting the knee. The knees are nearly fully contracted during their own swing phase as well as during the swing phase of the diagonal opposite leg. Before its own swing phase, a leg stretches fully to shift the weight to the diagonal opposite leg. The trajectories of the hind legs differ from the trajectories of the front legs due to the leg order within the walking cycle.

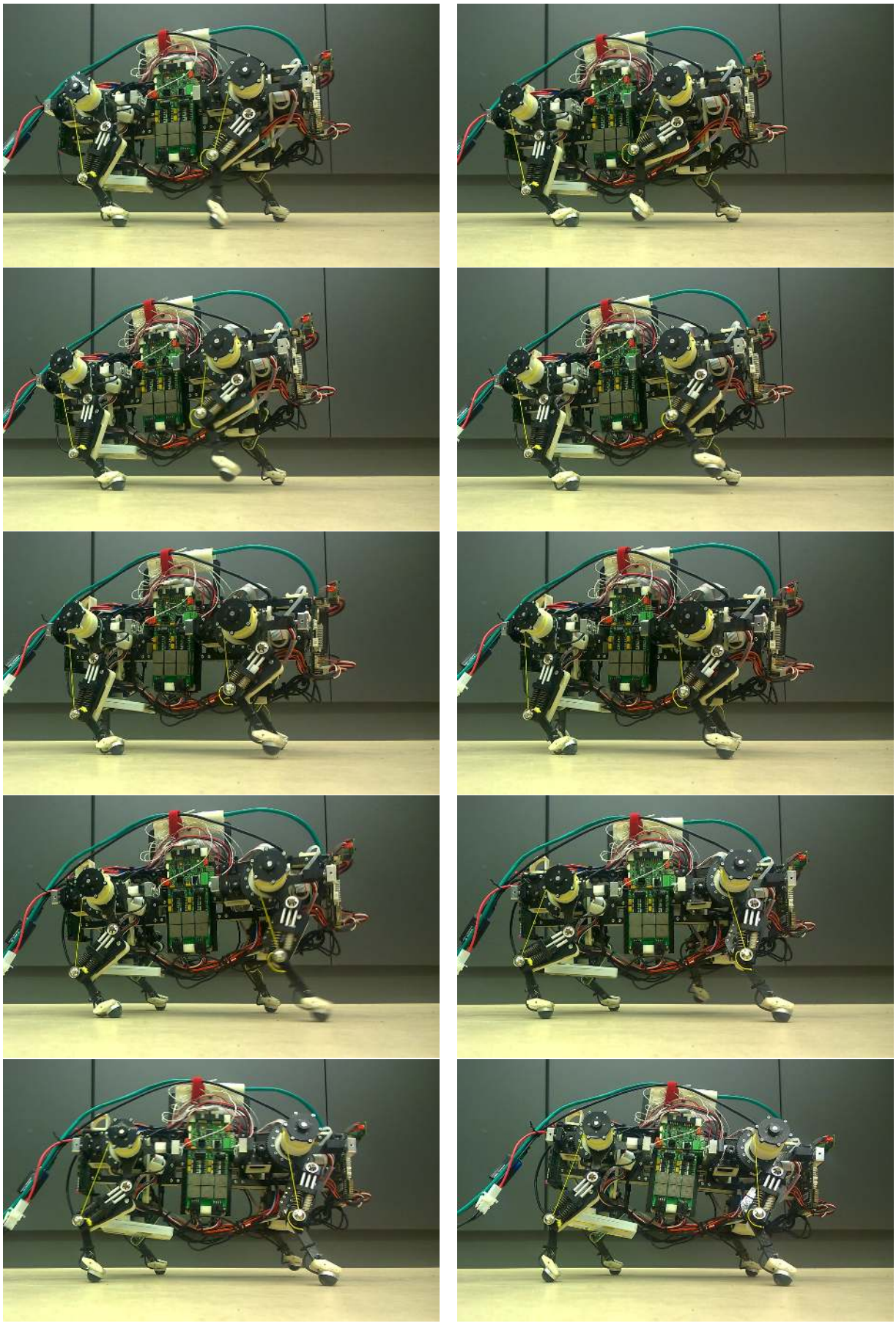
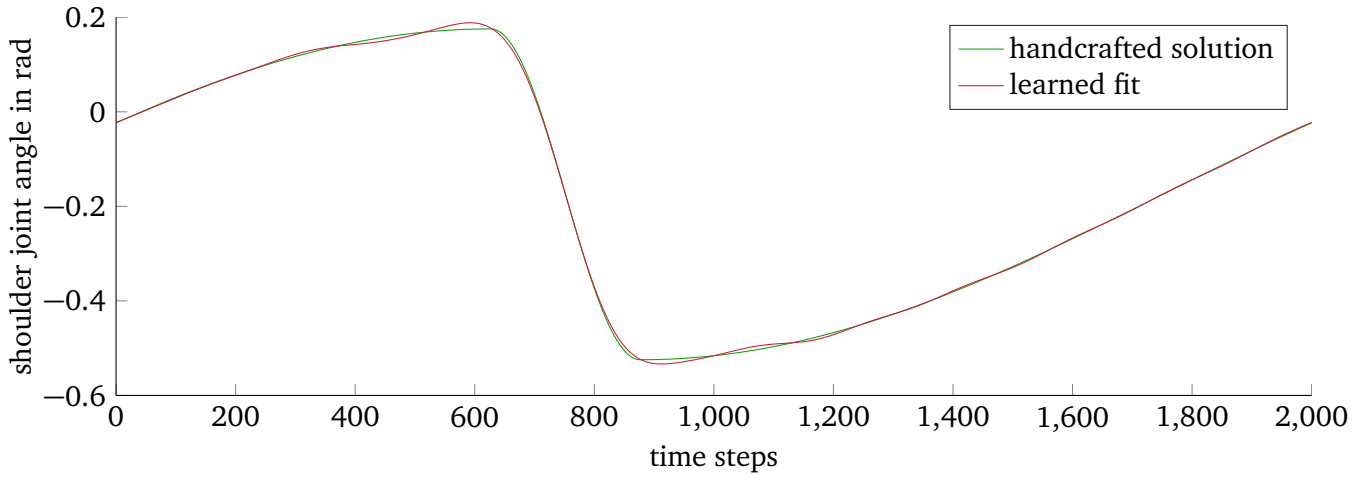
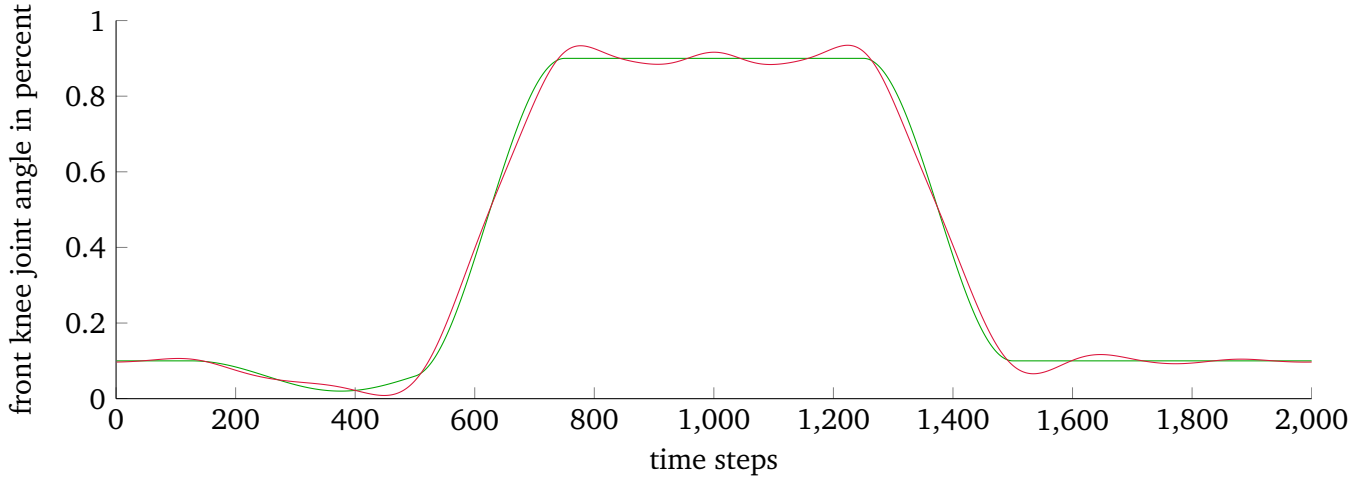


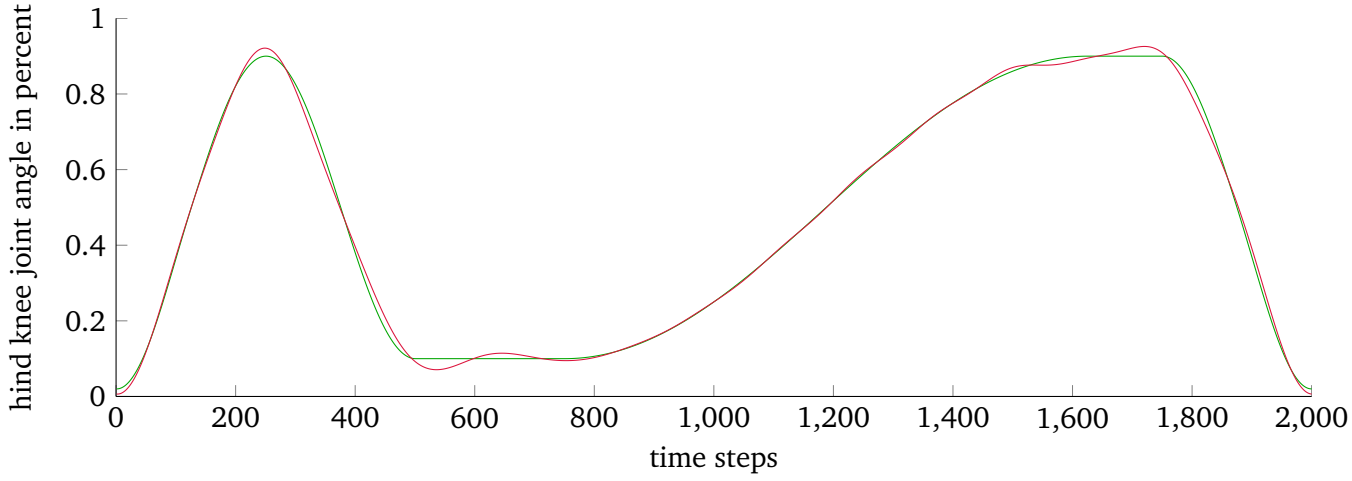
Figure 4.4.: Snapshots showing the hand-tuned gait



(a) shoulder trajectory



(b) front knee trajectory



(c) hind knee trajectory

Figure 4.5.: The modulation done with von Mises basis functions is compared with the handcrafted trajectory. 12 von Mises basis functions encode the shoulder trajectory, both of the knee trajectories use 16 basis functions each. The number of basis functions is hand-tuned, the weights are learned by using regression. The mixture model using von Mises basis functions is able to modulate the hand-tuned gait appropriately and is thus preferred, as it is linear in the parameters in contrast .

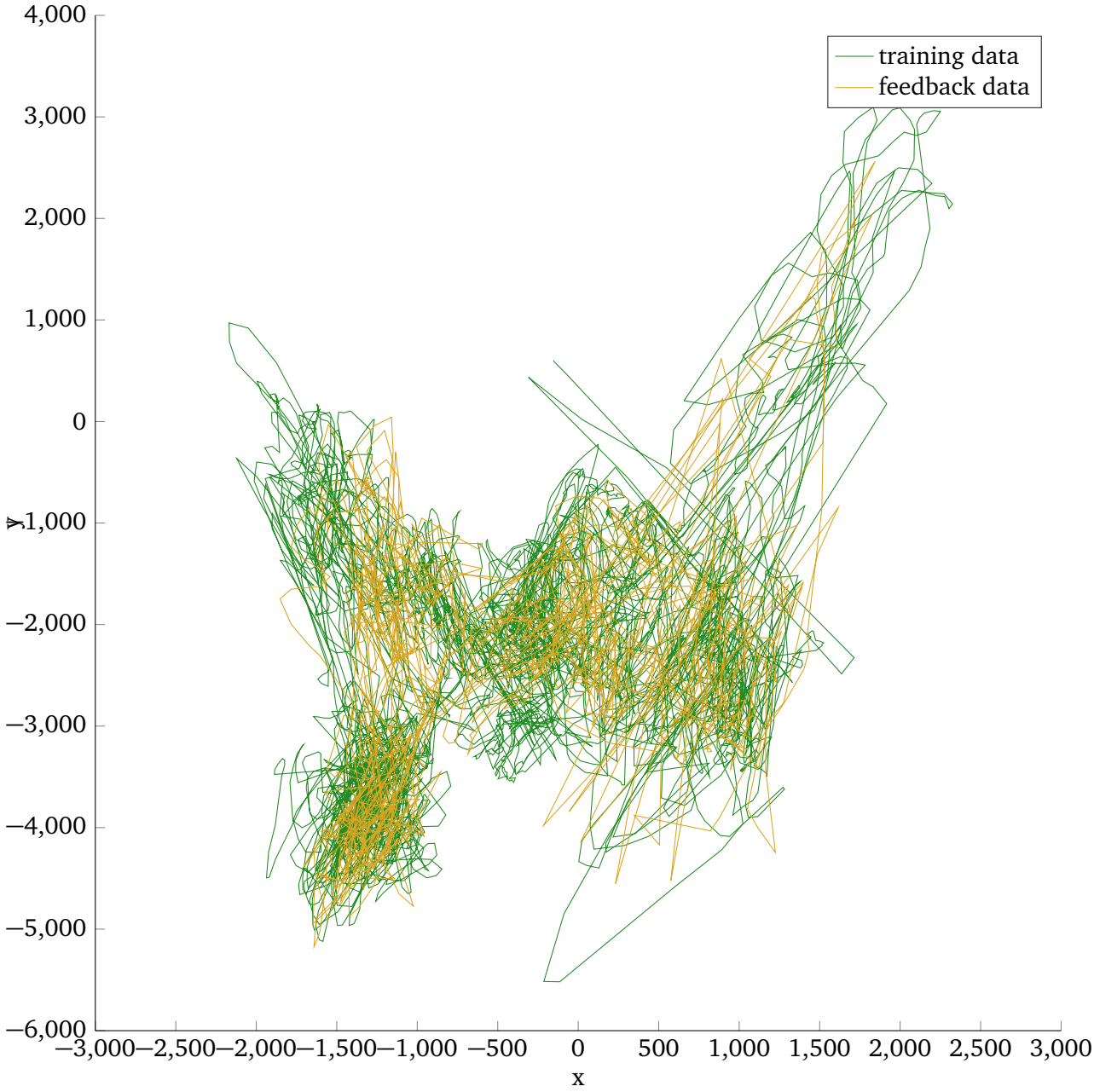
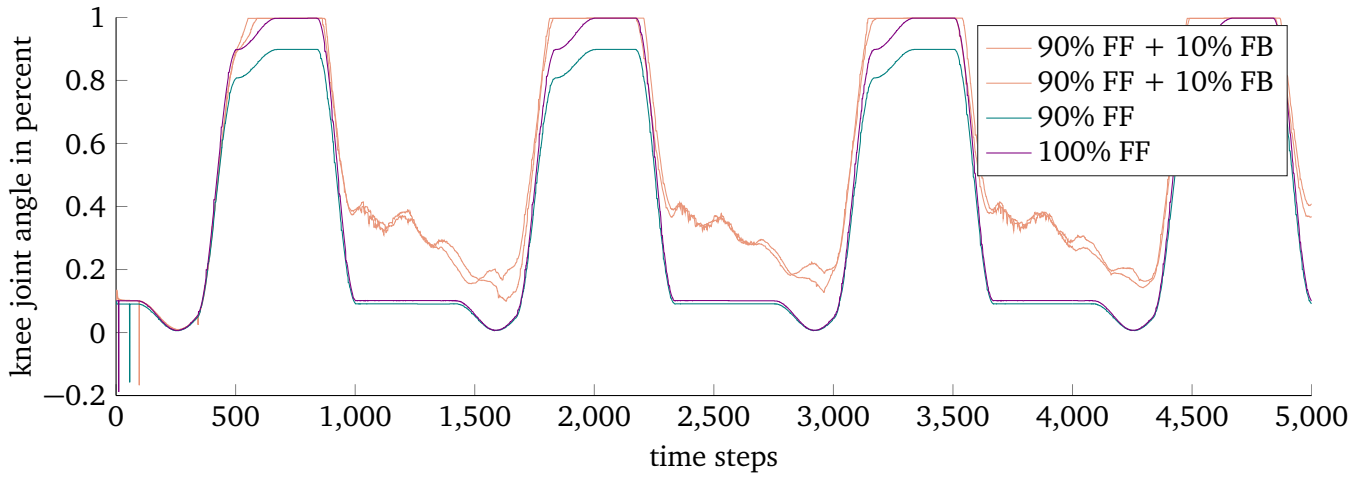
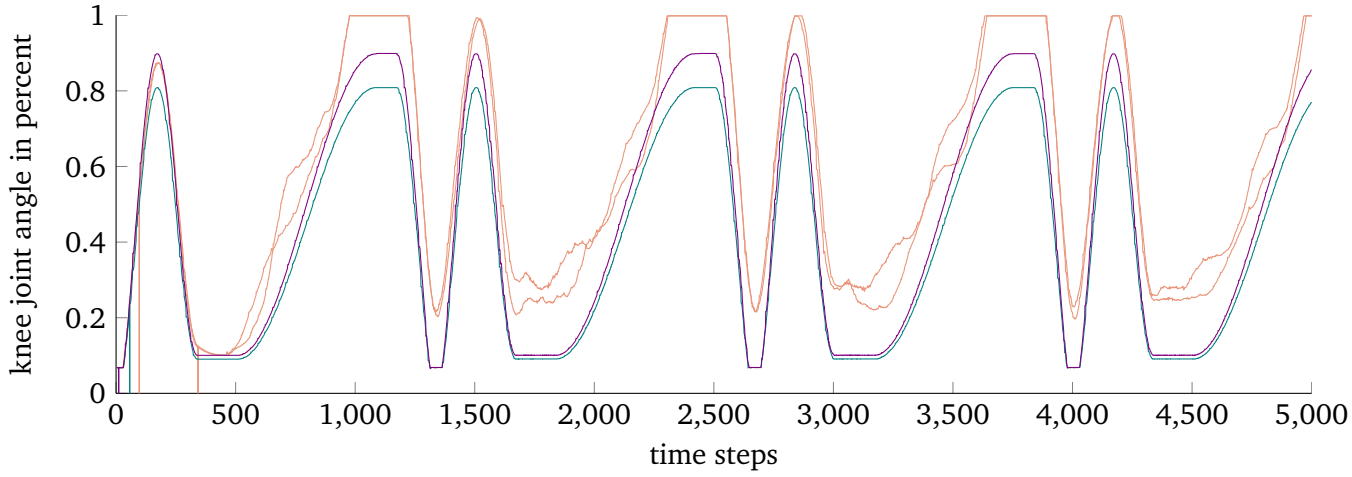


Figure 4.6.: The CoP data recorded while walking for training data generation is compared with the recordings from walking with applied feedback. The training data is generated by using the feedforward gait. In this plot, the parameter values are for the stepsize $\alpha = 1/5$ and for the feedback weighting $\omega = 1/10$ during the recording with applied feedback. The feedback does not seem to visibly influence the gait. The head of the Oncilla points to the right.



(a) left front knee trajectory



(b) left hind knee trajectory

Figure 4.7.: Recordings from the gait with applied feedback (red) are compared to the hand-tuned gait when executed alone, once as full gait (purple) and once with only 90% of the feedforward command (turquoise). Two different trajectories of the combined system are plotted to give a notion of how much recordings differ from each other due to stochasticity of the system. The feedback decreases the stretching of the knees. Hence, the Oncilla is in a more crouched position compared to the original hand-tuned gait.

5 Discussion

5.1 Conclusion

We evaluated stand-alone approaches for balancing and walking and tested the effects of a combination of the two. The balancing controller worked quite well for such a minimalistic approach. It uses a small training data set and predicts a reduced set of knee joints and yet it rebalances the robot after disturbances from the environment. Using the center of pressure as criterion for feedback is quite limited and thus will not be sufficient anymore when using dynamic gaits, where there may not exist a CoP at all during flight phases. Conceptually, the CoP is a criterion for statical balance and thus may be working against balance even in cases where dynamic gaits have a defined CoP.
The feedforward gait was hand-constructed and required a lot of tuning and experimenting until resulting in a movement actually bringing the Oncilla forward. One of the results was that the hind and front legs need different kind of trajectories. Also, not all flaws of the gait could be removed despite extensive adapting of the trajectory, as for example, the Oncilla's weight still does not rest on the right front leg when the left hind leg is supposed to be lifted, resulting in the Oncilla tilting backwards temporarily. We also showed that it is possible to learn parameters such that the gait is modulated by rhythmic movement primitives.
When combining those two components, some specific details that made the balancing controller work were not sensible anymore, as for example the contact angle of the force sensors switched constantly during walking. Also the way we combined feedforward and feedback worsened the constructed gait, as it was not fully executed anymore.
In the initial results we demonstrated that a combined approach results in suitable walking gaits, which have to be further improved in the future.

5.2 Technical Challenges of the Oncilla as Platform

We noticed some problems originated in the hardware system, making it difficult to control the robot.
A general problem is the passive compliance of the system, it makes precise control harder, especially when we have to use a kinematic model. The inverse kinematics can only be approximated. For a static stabilized gait, exact control is preferred.
The most interfering passive compliance is the one located in joint q_4 in the foot. As the joint has no sensor, it is not possible to figure out the angle of the foot, on which the measurements of the OptoForce sensors depend, resulting in sensor data that varies. A possible solution could be to fix this joint, which would result in a similar structure as seen on the HyQ or the StarlETH, but remove some of the biologically inspired compliance.
Due to their constant stiffness, springs may, overall, not be the best solution of including compliance to a robot, at least not if it is supposed to fulfill various tasks with different requirements on precision and compensating abilities, as for choosing springs, one has a trade-off between those two options. Active impedance as an alternative is discussed and shown to be useful in [30].
Another problem is the actuation of joints q_2 and q_3 by a cable driven leg retraction. Ideally, loosing the cable lets the springs extend the leg. However, the current springs are not strong enough to fulfill their task in all situations where the cable is loosened. Already about 2/5 of the Oncilla's weight impedes a full stretching of the leg, leading to a loose cable and another joint configuration than expected. Thus, in case the following commands require a flexion of the knee to balance, the command will not directly

result in the desired response, as first the loose cable will be rolled up, leading to no change of the actual knee flexion.
A demanding property is the weight distribution within the robot. The weight is not distributed equally, and not even symmetrically. This fact is quite obvious when taking a look at the walking gait, where the swing phases of the right legs can perform quite well, while the hind left leg has an interrupted swing phase due to the weight of the Oncilla not being shifted enough to the front right leg. Also, the weight distribution is not very beneficial regarding the lifting of the legs themselves, as the diagonal opposite leg has to bend at least 90% to allow the foot to leave the ground. This is even more difficult during walking, as with a momentum working in another direction, even more contraction of the diagonal opposite leg is necessary. These required extreme movements which increase the danger of twisting the joints of the Oncilla.
Due to the bio-inspired leg construction, flexing the knee does not retract the leg linearly. When working in joint space, this makes it difficult to plan and hand-tune a gait trajectory that does not include unwanted propelling or acceleration, as flexing the knee always moves the foot backwards.
An additional drawback of using the robot as an autonomous walking system is the lack of an IMU in the current setup. Thus, we cannot determine the robot's position relative to the world coordinates.

5.3 Future Work on Feedforward and Feedback Control

As the current setup is only a first and very basic approach to a walking Oncilla, there are much opportunities for all components to be improved. Some ideas and suggestions are presented in this section.

5.3.1 More Elaborated Feedback

The first step to improved feedback should be to try different movement trajectories for the generation of training data; including the walking trajectory of the Oncilla, as it will produce the training data being closer to what LWR receives during application. Another attempt could be simply lifting the legs, since it provides the information of how the CoP changes according to lifted legs. More elaborated training data could be gained when using online learning.
Another idea is to refine the model predictions. Right now, we use a quite limited set of predictions: We only correct the knee motor position und we use the same prediction for both knees on one side. So one of the next steps should be to expand the LWR training data and prediction dimension such that each knee is treated independently. Further steps could include both shoulder joints as well. Especially the q_0 joints could be interesting for walking on uneven terrain, because small steps sideways can help to rebalance the robot.
As we use the pseudoinverse for calculating the feedback, we could make use of the constraints that can be included in this method. A constraint handling the joint limits of the Oncilla could force other solutions as response, which could lead to smarter balancing adjustments of the Oncilla. For example, currently the setup often tries to contract a fully contracted leg even more. Instead, lengthening the other legs should be an alternative solution.
Using Gaussian processes (GPs) instead of LWR could also lead to improved feedback. A well tuned GP should yield predictions faster than LWR, since a GP prediction requires less mathematical calculations. Additionally it provides uncertainty that could also be used for adapting the weighting ω of the feedback. However, for the previously mentioned online learning, GPs could be a less suitable solution regarding their longer training time compared to LWR.
A more elaborated approach could be to use the balancing controller as a real controller that tracks a given CoP-trajectory. For this approach, we would have to create a CoP-trajectory that shifts the weight towards the diagonal opposite leg of the next swing leg. Such an approach would be similar to [52]. In this case, the training data could include each leg bending separately, giving data of how these movements influence the position of the CoP. A well functioning CoP tracker can also allow the steering of the

Oncilla towards desired directions which can be done by defining a respective CoP trajectory. Thus, for such application cases the feedback component would have a higher weighting ω than it does currently. With such enhanced feedback compenents, we could also experiment with adding the feedback only to legs in stance phase or only to legs in swing phase versus constantly adding feedback to all legs. A more elaborated approach to set the weighting for the feedback would be to learn ω together with the gait parameters which is discussed in the next section.

5.3.2 Learning Gait Parameters

Presently, the gait needs some improvements. The sliding of the feet shows that the leg trajectories are not coordinated well enough with each other, so the shoulders could make a compensating movement when the knees are contracted or stretched. Also, a good trajectory would have brought the Oncilla into another position where the weight is better distributed, which means shifted more onto the leg such that the resulting higher friction would reduce sliding.
As hand-tuning the gait is quite hard and has not led to satisfying results thus far, we could now make use of having a compact representation in form of movement primitives which use von Mises basis functions. The reduced linear parameter set could be learned by using Reinforcement Learning.
After letting the robot walk for some time, we could calculate the reward for the current gait parameter set depending on how long the robot can walk, how much distance it covered and maybe, by using an Optitrack system or an IMU, also the roll and pitch movements which indicate the stability and smoothness of the gait. Using contact times, which can directly be derived from foot sensor recordings, would also allow to relate recorded gait data to the duty factor. For example, one could compare stance phase duration for each cycle and see if it is consistent, as consistency gives a hint of how regular the current gait is. Further we can regard all contact times and see how closely we actually achieve the type of gait we intended. Both of these options could help to evaluate the gait and thus generate feedback. It would be interesting to see if the results can handle the unequal weight distribution of the Oncilla, thus resulting in a gait with four continuous swing phases.

5.3.3 Probabilistic CPGs (PCPG)

Our general vision is to improve the control system by making use of statistical benefits. This goal can be achieved by extending the movement primitives to probabilistic movement primitives (ProMPs), where we do not have a single trajectory τ anymore but a distribution over trajectories. Modulating the gait as ProMPs results in a probabilistic CPG that delivers uncertainties regarding the current prediction.

Probabilistic Movement Primitives

A probabilistic version of MPs has already been introduced as probabilistic movement primitives by Paraschos [53]. ProMPs model a distribution over trajectories $p(\tau)$, with a trajectory τ being specified by its values y_t for each time step t: $\tau = y_{1:T}$. Like for the movement primitives, each trajectory point y_t is specified by a linear basis function model $\mathbf{\Psi}$, i.e. $y_t = \mathbf{\Psi}_t^T \mathbf{w} + \epsilon_y$ with $\mathbf{w}$ being the compact representation of the trajectory.
Given such a weight vector w, the probability for a trajectory τ is a Gaussian trajectory model

$$p(\tau|\mathbf{w}) = \prod_t \mathcal{N}(y_t|\mathbf{\Psi}_t^T \mathbf{w}, \mathbf{\Sigma}_y).$$

As for movement primitives, the handcrafted gait can provide the required training data.
Besides stating variance, ProMPs provide some additional features that could allow a more complex walking system: In case a multi-dimensional trajectory is used, ProMPs include a co-variance matrix.

Its entries can be changed, which adapts the co-activation of trajectories. This feature could allow gait transition, as the phase offset between the different legs could be adjusted.
A more advanced variation could include planning of trajectories in task space and then use via points to modulate the trajectory such that exact foot placement or step height can be planned.

5.3.4 Probabilistic Feedforward Controller with Adaptive Tactile Feedback Error Correction

The use of ProMPs to modulate the gait brings the advantage of having a variance or better the uncertainty of the feedforward prediction at each time step. This information can be used to adapt the weighting ω of the feedback according to the current uncertainty of the feedforward command: When there is high uncertainty, more feedback may be required than when uncertainty is very low.
Additionally, the quality of LWR predictions can be evaluated as well, for example by comparing the prediction to the actual CoP. The evaluation can be used to set the weight of the feedback command, as a bad prediction may indicate a bad LWR model, resulting in a bad Jacobian and thus a less reliable pseudoinverse. So in case the LWR prediction was far off from the actual CoP, we should use less amount of feedback.

5.4 More General Ideas for Future Work

Besides the next steps that could be done with the Oncilla, there are many features of other quadrupeds that seem to be useful for future autonomous systems and hence should be exploited, even though they are not directly emerging from the current setup.
We could extend or change the current system such that it allows to modulate the transition from swing phase to stance phase and vice versa. This extension would help in cases of bumps and holes along the way just like they occur in the outside world. It also could bring our approach closer to biology than strictly feedforward rhythmic gait patterns. There seems to be a switch between stance and swing phase induced by feedback from tactile sensing as suggested by [1]. One possibility to implement this feature would be to use a state machine, as it was done by [15]. Another way, which would be closer to our current setup, would be the modulation of the rhythmic pattern. For example, Gay [36] achieved modulation for CPGs by using neural network feedback generated from information about rolling of the robot which was inferred from a camera or an IMU. Those modulations do not only adapt the phase length, but also the step height. Such modulations are also done for the Tekken II robot [14].
The CoP is a very mechanical description of balance and it is not very close to how animals make use of tactile information. A quite simple approach would be to directly use the force measurements of the feet without including the endeffector position, but instead use joint positions as additional input. This approach is independent of the Oncilla frame position relative to the world's frame, thus it does not require an Optitrack system.
On the hardware side, a flexible trunk would compensate twisting and rotational movements which emerge from leg movements that are not perfectly coordinated with each other. It could also allow for bending of the spine which is helpful for running gaits with a flight phase as shown by the Boston Dynamics Cheetah [54]. Another way to avoid the problem is shown on the fast sprinting robot Raptor from KAIST [55] which has nearly no displacement between front and hind shoulders, and thus nearly no trunk at all. Here the question is how a spine would influence the control of the robot. Other features which current bio-inspired platforms still lack are the ability to adjust joint stiffness by increasing or lowering body tension. Such an improvement could be useful for instance for the ankle joint when placing or lifting the foot, because a fixed ankle allows the application of forces that push off the ground. One of the few robots with such a feature is the hydraulic quadruped HyQ [29], which uses active compliance and thus is able to vary the compliance during execution.

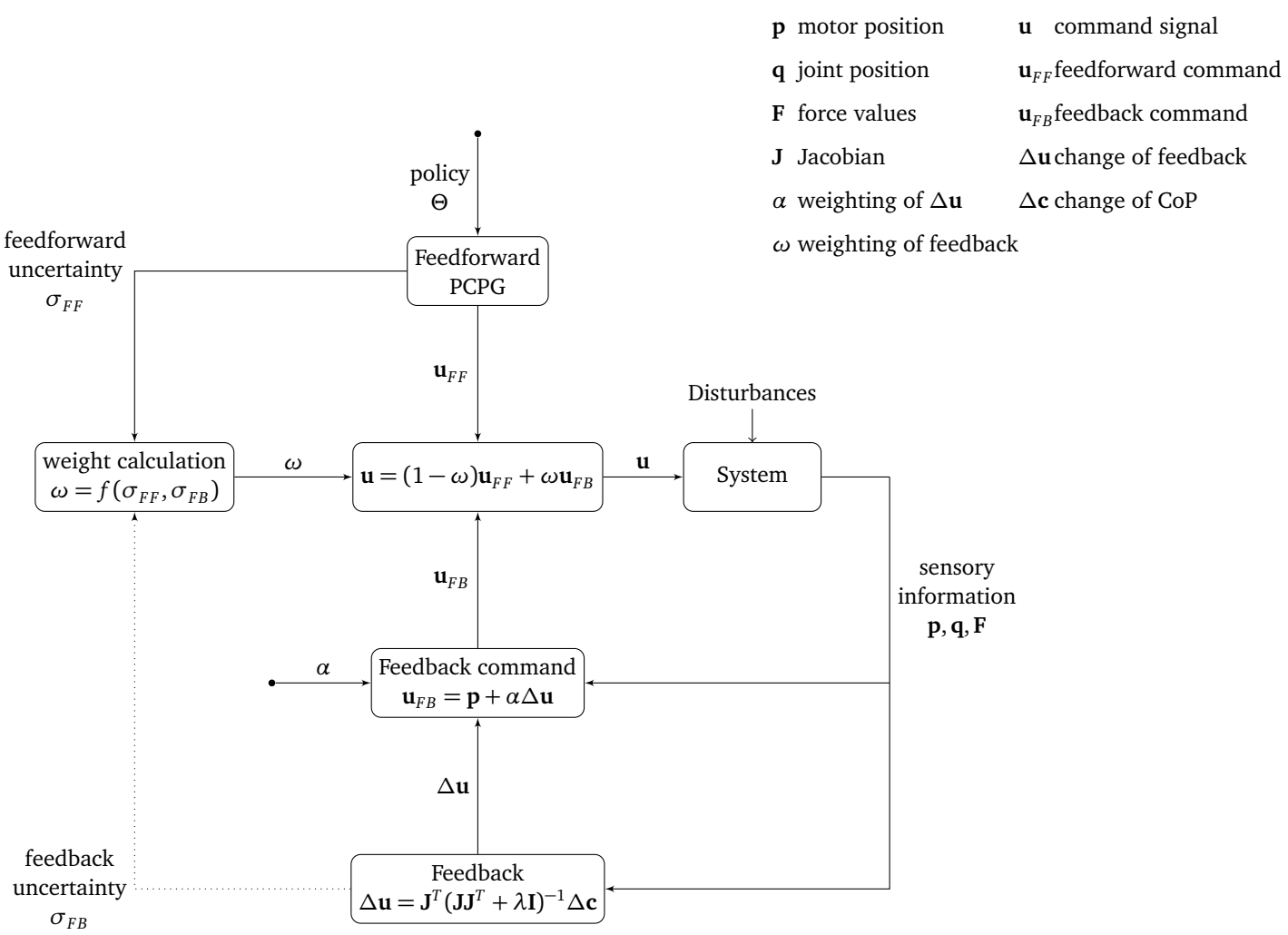


Figure 5.1.: This diagram of a probabilistic feedforward controller with adaptive tactile feedback error correction extends the setup that was already presented in Figure 3.5 by the new block *weight calculation* on the left. A function $\omega = f(\sigma_{FF})$ that calculates the weight ω depending on the variance of the feedforward $\sigma_F F$ gait could be part of future work. A further possible extension which additionally takes the uncertainty or the error of the feedback model into account is indicated by a dotted line. In this case, ω will depend on σ_{FB} as well, as denoted in the function: $\omega = f(\sigma_{FF}, \sigma_{FB})$.

Bibliography

[1] P. Whelan, "Control of Locomotion in the Decerebrate Cat," *Progress in Neurobiology*, vol. 49, no. 5, pp. 481–515, 1996.

[2] R. McGhee and a.a. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, pp. 331–351, 1968.

[3] M. H. Raibert, H. B. Brown, M. Chepponis, E. Hastings, J. Koechling, K. N. Murphy, S. S. Murthy, and A. J. Stentz, "Dynamically Stable Legged Locomotion," Tech. Rep. 4148, 1983.

[4] M. Raibert, "BigDog, the rough-terrain quadruped robot," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 17, pp. 6–9, 2008.

[5] Boston Dynamics, "Introducing Spot." `https://www.youtube.com/watch?v=M8YjvHYbZ9w`.

[6] S. Takamuku, A. Fukuda, and K. Hosoda, "Repetitive grasping with anthropomorphic skin-covered hand enables robust haptic recognition," *Proceedings of IROS*, 2008.

[7] R. Blickhan, A. Seyfarth, H. Geyer, S. Grimmer, H. Wagner, and M. Günther, "Intelligence by mechanics," *Philosophical Transactions A*, vol. 365, no. 1850, pp. 199–220, 2007.

[8] S. Rutishauser, A. Sproewitz, L. Righetti, and A. J. Ijspeert, "Passive compliant quadruped robot using Central Pattern Generators for locomotion control," *2008 2nd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, no. c, pp. 710–715, 2008.

[9] A. Sproewitz, L. Kuechler, A. Tuleu, M. Ajallooeian, M. D'Haene, R. Moeckel, and A. J. Ijspeert, "Oncilla Robot, A Light-weight Bio-inspired Quadruped Robot for Fast Locomotion in Rough Terrain," in *Symposium on Adaptive Motion of Animals and Machines (AMAM2011)*, pp. 63–64, 2011.

[10] M. Vukobratović and B. Borovac, "Zero-Moment Point — Thirty Five Years of Its Life," *International Journal of Humanoid Robotics*, vol. 01, no. 01, pp. 157–173, 2004.

[11] S. Degallier, L. Righetti, S. Gay, and A. Ijspeert, "Toward simple control for complex, autonomous robotic applications: Combining discrete and rhythmic motor primitives," *Autonomous Robots*, vol. 31, no. 2-3, pp. 155–181, 2011.

[12] A. Tero, M. Akiyama, D. Owaki, T. Kano, A. Ishiguro, and R. Kobayashi, "Interlimb neural connection is not required for gait transition in quadruped locomotion," 2013.

[13] Boston Dynamics, "RHex Rough-Terrain Robot." `https://www.youtube.com/watch?v=ISznqY3kESI`.

[14] Y. Fukuoka, H. Kimura, and A. H. Cohen, "Adaptive Dynamic Walking of a Quadruped Robot on Irregular Terrain Based on Biological Concepts," *Proceedings of ICRA*, vol. 2, pp. 2037 – 2042, 2003.

[15] M. Palankar and L. Palmer, "A force threshold-based position controller for legged locomotion," *Autonomous Robots*, vol. 38, no. 3, pp. 301–316, 2014.

[16] U. Saranli, M. Buehler, and D. E. Koditschek, "RHex: A Simple and Highly Mobile Hexapod Robot," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 616–631, 2001.

[17] P. Ha and K. Byl, "Feasibility and Optimization of Fast Quadruped Walking with One- Versus Two-at-a-Time Swing Leg Motions for RoboSimian," 2014.

[18] T. Kamioka, T. Watabe, M. Kanazawa, H. Kaneko, and T. Yoshiike, "Dynamic Gait Transition between Bipedal and Quadrupedal Locomotion," *International Conference on Intelligent Robots and Systems*, pp. 2195–2201, 2015.

[19] DARPA, "DARPA Challenge."

[20] D. Owaki, L. Morikawa, and A. Ishiguro, "Listen to body's message: Quadruped robot that fully exploits physical interaction between legs," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1950–1955, 2012.

[21] Z. J. Kolter and A. Y. Ng, "The Stanford LittleDog: A learning and rapid replanning approach to quadruped locomotion," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 150–174, 2011.

[22] M. Zucker, N. Ratliff, M. Stolle, J. Chestnutt, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "Optimization and learning for rough terrain legged locomotion," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 175–191, 2011.

[23] A. Shkolnik, M. Levashov, I. R. Manchester, and R. Tedrake, "Bounding on rough terrain with the LittleDog robot," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 192–215, 2011.

[24] P. D. Neuhaus, J. E. Pratt, and M. J. Johnson, "Comprehensive summary of the Institute for Human and Machine Cognition's experience with LittleDog," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 216–235, 2011.

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

[26] M. P. Murphy, A. Saunders, C. Moreira, A. A. Rizzi, and M. Raibert, "The LittleDog robot," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 145–149, 2011.

[27] L. Righetti, J. Buchli, M. Mistry, and S. Schaal, "Control of legged robots with optimal distribution of contact forces," *International Journal of Robotics Research*, vol. 32, no. 3, pp. 280–298, 2013.

[28] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, "Compliant Quadruped Locomotion Over Rough Terrain," *International Conference on Intelligent Robots and Systems*, pp. 814–820, 2009.

[29] C. Semini, *HyQ - Design and Development of a Hydraulically Actuated Quadruped Robot*. PhD thesis, 2010.

[30] C. Semini, V. Barasuol, T. Boaventura, M. Frigerio, M. Focchi, D. G. Caldwell, and J. Buchli, "Towards versatile legged robots through active impedance control," *The International Journal of Robotics Research*, vol. 34, no. 7, 2015.

[31] D. J. Hyun, S. Seok, J. Lee, and S. Kim, "High Speed Trot-running : Implementation of a Hierarchical Controller using Proprioceptive Impedance Control on the MIT Cheetah," *The International Journal of Robotics Research*, vol. 33, no. 11, pp. 1417–1445, 2014.

[32] Massachusetts Institute of Technology (MIT), "MIT Cheetah lands the running jump." `https://www.youtube.com/watch?v=_luhn7TLfWU`.

[33] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, "Control of Dynamic Gaits for a Quadrupedal Robot," *International Conference on Robotics and Automation*, pp. 3287–3292, 2013.

[34] A. Sproewitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with Cheetah-cub, a compliant quadruped robot," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 932–950, 2013.

[35] J. Degrave, T. Waegeman, and B. Schrauwen, "Comparing Trotting and Turning Strategies on the Quadrupedal Oncilla Robot," *International Conference on Robotics and Biomimetics*, pp. 228–233, 2013.

[36] S. Gay, J. Santos-Victor, and A. Ijspeert, "Learning robot gait stability using neural networks as sensory feedback function for Central Pattern Generators," *International Conference on Intelligent Robots and Systems*, no. July 2015, pp. 194–201, 2013.

[37] M. Ajallooeian, S. Pouya, A. Sproewitz, and A. J. Ijspeert, "Central Pattern Generators augmented with virtual model control for quadruped rough terrain locomotion," *Proceedings of International Conference on Robotics and Automation*, pp. 3321–3328, 2013.

[38] M. Ajallooeian, A. Tuleu, A. Sproewitz, P. Eckert, M. Vespignani, and A. J. Ijspeert, "Rich Locomotion Skills with the Oncilla Robot," 2014.

[39] A. T. Sproewitz, M. Ajallooeian, A. Tuleu, and A. J. Ijspeert, "Kinematic primitives for walking and trotting gaits of a quadruped robot with compliant legs.," *Frontiers in computational neuroscience*, vol. 8, no. 27, 2014.

[40] OPTOFORCE, "3D Force Sensor OMD-20-SA-60N." `http://optoforce.com/wp-content/uploads/2015/07/OptoForce_Datasheet_OMD-20-SE-40N.pdf`, 2014.

[41] Oncilla Robot, "trainingDataGeneration." `https://www.youtube.com/watch?v=vG7hXRulzao`.

[42] C. Atkeson, A. W. Moore, and S. Schaal, "Locally Weighted Learning," *Artificial Intelligence Review*, vol. 11, pp. 11–73, 1997.

[43] M. Hildebrand, "Symmetrical Gaits of Horses," *Science*, vol. 150, pp. 701–708, 1965.

[44] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors.," *Neural computation*, vol. 25, no. 2, pp. 328–73, 2013.

[45] Oncilla Robot, "balancing." `https://www.youtube.com/watch?v=iKXC1lo3wq4`.

[46] Oncilla Robot, "walkingSimpleHandcrafted." `https://www.youtube.com/watch?v=RehZNUlgVL4`.

[47] Oncilla Robot, "walkingElaboratedHandcrafted." `https://www.youtube.com/watch?v=pyjQ3_yXtKU`.

[48] Oncilla Robot, "walkingMixtures." `https://www.youtube.com/watch?v=xR6MjCaJFao`.

[49] Oncilla Robot, "shakyWalkingOmega20Alpha2." `https://www.youtube.com/watch?v=uB1VBR3TLA4`.

[50] Oncilla Robot, "walkingOmega10NoFeedback." `https://www.youtube.com/watch?v=_x6b-eSI3Ps`.

[51] Oncilla Robot, “walkingOmega10Alpha5StrongSmoothing.” https://www.youtube.com/watch?v=_zjWxG4Z9gU.

[52] J. Z. Kolter and A. Y. Ng, “Learning Omnidirectional Path Following Using Dimensionality Reduction,” *Proceedings of Robotics: Science and Systems*, 2007.

[53] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, “Probabilistic Movement Primitives,” *Advances in Neural Information Processing Systems*, pp. 2616–2624, 2013.

[54] Boston Dynamics, “Cheetah robot runs 28.3 mph.” https://www.youtube.com/watch?v=chPanW0QWhA.

[55] BMG KAIST, “KAIST Raptor robot runs at 46 km/h.” https://www.youtube.com/watch?v=lPEg83vF_Tw.

A Appendix

A.1 Forward Kinematics of the Oncilla

We derived simplified forward kinematics of the Oncilla by assuming a rigid body. This means that we neglect the compliance in leg segments and in the unactuated q_4. Such a simplification is reasonable as we operate only with slow speed, not compressing the springs passively, and in static stable situations. Also we do not take the parallel mechanism in the knees into account. The endeffector is calculated relatively to shoulder or hip coordinates as follows.
For leg 1 and 2, we obtained

$$x = 6.2\sin(\theta_1) + 5.5\sin(\theta_1 + \theta_2) + 5.8\sin(\theta_1 + \theta_2 + \theta_3),$$

$$z = 6.2\cos(\theta_1) + 5.5\cos(\theta_1 + \theta_2) + 5.8\cos(\theta_1 + \theta_2 + \theta_3).$$

For leg 3 and 4, we obtained

$$x = 8.0\sin(\theta_1) + 6.5\sin(\theta_1 + \theta_2) + 5.0\sin(\theta_1 + \theta_2 + \theta_3),$$

$$z = 8.0\cos(\theta_1) + 6.5\cos(\theta_1 + \theta_2) + 5.0\cos(\theta_1 + \theta_2 + \theta_3).$$

The endeffector coordinate for y is always constant, as joint q_0 is not used and, thus, the legs do not move along y direction.

A.2 Influence of Ground Placement on Recorded OptoForce Data

Depending on the angle α of joint q_4 (a depiction can be seen in Figure A.1), the OptoForce sensors give different measurements despite actually having the same force F_z^* acting on them. As neither the angle α nor the actual force F_z^* is known, it is not possible to calculate the actual force or to map the measured force F_z to the actual one.

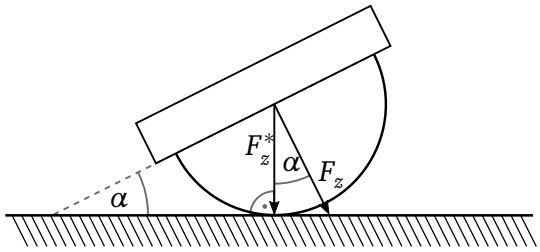


Figure A.1.: The measurements of the OptoForce sensors F_z depend on the angle α between the foot and the ground. Since this angle is unknown and the respective joint is not actuated, the actual weight on the foot F_z^* cannot be computed.

A.3 LWR Benefit of Adding Velocity

Adding velocity to the LWR input increases its performance, as the results show in Figure A.3 .

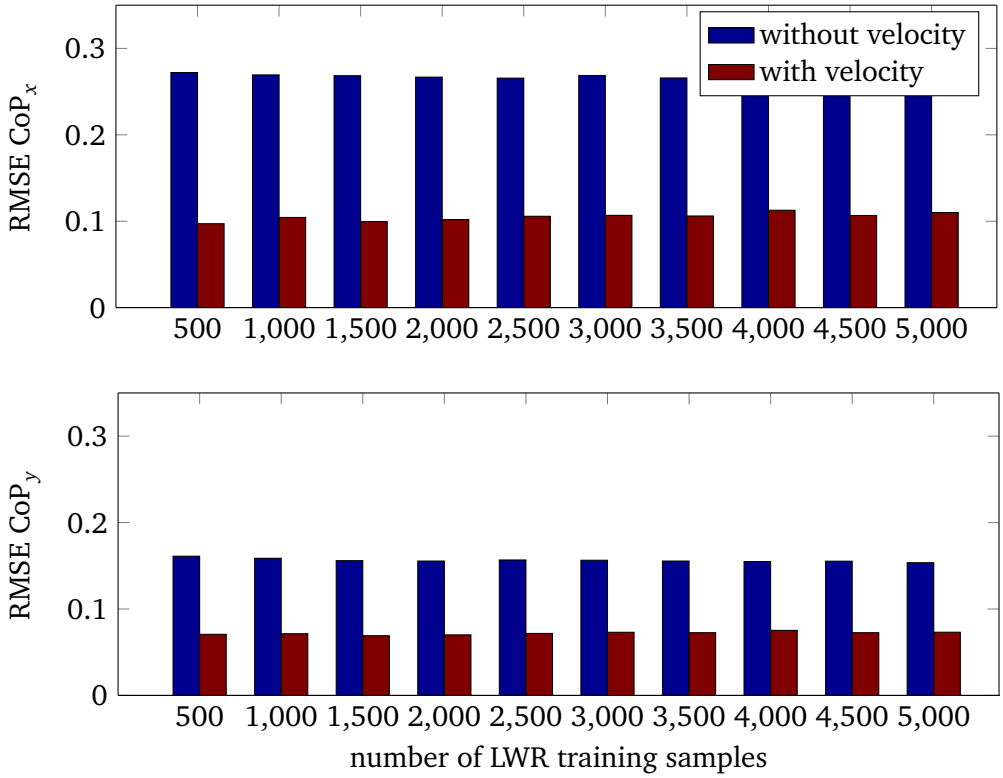


Figure A.2.: RMSE as percentage of data range depends on the amount of LWR training samples. Velocity as additional input parameter of the LWR model decreases the RMSE by about 50%.

A.4 LWR Tradeoff between Precision and Speed

While more training data allows LWR to make better predictions, as the results in Figure A.3 show, a large amount of training data slows the prediction time down, which can be seen in Figure A.4. Thus, we are limited to about 2000 training data points due to time constraints.

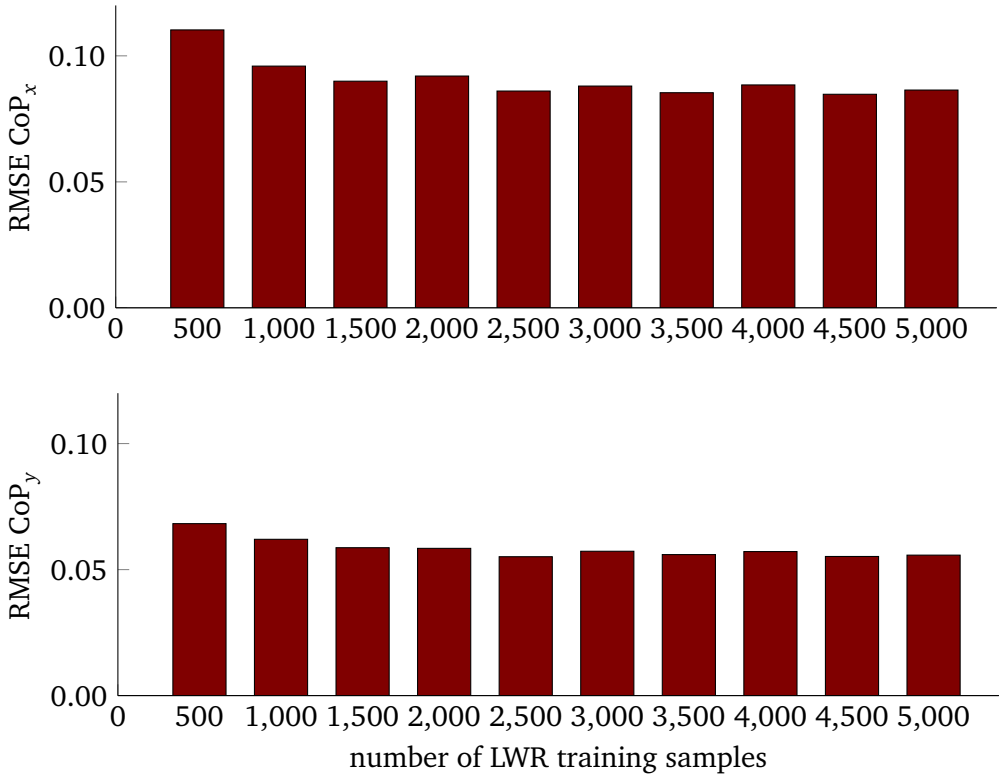


Figure A.3.: RMSE as percentage of data range depends on the amount of LWR training samples. A larger amount of training data allows preciser predictions.

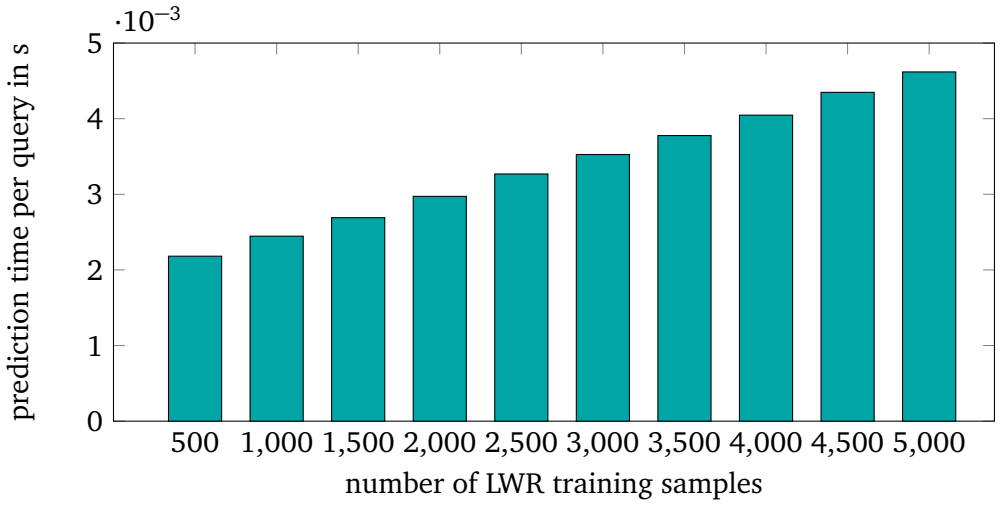


Figure A.4.: LWR prediction time as it depends on the number of training samples. A larger amount of training data requires more prediction time.

A.5 Filtering

As we can use only few training data points for LWR and we have a noisy system, smoothing is a useful tool to recover the trend of the data. When applying the learned forward model, the data from the different data streams is noisy as well and filtering improves the outcome. The feedback predictions are still often rather noisy, such that they need filtering again to improve the safety of the robot. Our experiments showed that finding a suitable filter and tuning it correctly is quite crucial.
One of the simplest approaches for smoothing serial data is the moving average filter with some weighting c, which calculates the value v_n as follows:

$$v_n = (1-c)\frac{\sum_{i=1}^{m} v_{\text{raw},n-i}}{m} + c\, v_{\text{raw},n}.$$

While heavily filtered CoP data leads to good results for LWR prediction, the application for a real time system must be more deliberated: Smoothing the CoP too much leads to a time delay for the position of the CoP, resulting in a delayed reaction which keeps the Oncilla from balancing properly. Due to the CoP delay it overshoots the desired position, tries to steer back and overshoots again. So for the CoP smoothing, we finally used an already available version of the butterworth filter.
Regarding the input data for the LWR model, the velocity signals of the Oncilla seem to be step functions, having only three different values: velocity when bending a knee, velocity when retracting it, and zero when the other side is in action. The other filters tend to smooth the edges too much, so we tried a simple variation of the moving average, the so called moving median, to preserve the sharp edges when computing the filtered value

$$v_n = \text{median}(v_{\text{raw},n-i}, .., v_{\text{raw},n}).$$

Evaluations on the Oncilla showed that smoothing the velocity data towards a clear step function actually had a negative influence on LWR. The Oncilla gets stuck in several positions and then reacts slowly to gentle pushing, as rarer input data points are not known to the model.
The LWR feedback was smoothed using a moving average filter again. To test only the balancing controller itself, 15 previously seen data points were enough. For the combined approach, we had to use up to 100 previous data points, which is also due to the usage of a larger α, as it gives more influence to irregularities in the predicted command change.
So smoothing in general helps to improve the LWR performance on the sparse data we have, but it has to be applied carefully as it may limit the range of training data and thus the space where LWR can work properly.