CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Department of Computer Science

Master's thesis

Learnable state estimator for multi-legged robot

Jiří Kubík

August 2021

Supervisor: Ing. Petr Čížek

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Kubík**	Jméno: **Jiří**	Osobní číslo: **457189**
Fakulta/ústav:	**Fakulta elektrotechnická**		
Zadávající katedra/ústav:	**Katedra počítačů**		
Studijní program:	**Otevřená informatika**		
Specializace:	**Umělá inteligence**		

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Learnable state estimator for multi-legged robot

Název diplomové práce anglicky:

Learnable state estimator for multi-legged robot

Pokyny pro vypracování:

Seznam doporučené literatury:

[1] J. Faigl, P. Čížek. Adaptive locomotion control of hexapod walking robot for traversing rough terrains with position feedback only, Robotics and Autonomous Systems, 116:136-147, 2019, https://doi.org/10.1016/j.robot.2019.03.008
[2] J. Hwangbo, C. D. Bellicoso, P. Fankhauser, and M. Hutter. Probabilistic foot contact estimation by fusing information from dynamics and differential/forward kinematics. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 3872-3878, 2016, https://doi.org/10.1109/IROS.2016.7759570
[3] S. Nobili, M. Camurri, V. Barasuol, M. Focchi, D. Caldwell, C. Semini, and M.Fallon. Heterogeneous Sensor Fusion for Accurate State Estimation of Dynamic Legged Robots. Proc. of Robotics: Science and Systems XIII. 2017. https://doi.org/10.15607/RSS.2017.XIII.007
[4] E. Lubbe, D. Withey, and K. R. Uren. State estimation for a hexapod robot. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 6286-6291, 2015, https://doi.org/10.1109/IROS.2015.7354274
[5] S. Haddadin, A. D. Luca, and A. Albu-Schaffer. Robot Collisions: A Survey on Detection,Isolation, and Identification. IEEE Transactions on Robotics, 33(6):1292-1312, 2017, https://doi.org/10.1109/TRO.2017.2723903

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Petr Čížek, centrum umělé inteligence FEL

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

Datum zadání diplomové práce: **21.02.2021** Termín odevzdání diplomové práce: **13.08.2021**

Platnost zadání diplomové práce: **19.02.2023**

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

III. PŘEVZETÍ ZADÁNÍ

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

.	
Datum převzetí zadání	Podpis studenta

Declaration

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

Prague, August 13, 2021

. . . .

Jiří Kubík

Acknowledgement

I would like to thank my family for both material and mental support, my supervisor for the necessary guidance, my friends and colleagues for support during these challenging pandemic times, namely David V. for transportation to the lab during the pandemic, Miloš P. for listening to my problems and Jindřiška D. for emotional support.

Abstrakt

Vícenozí kráčející roboti mají v porovnání se svými kolovými nebo pásovými protějšky relativně komplexní morfologii, která jim poskytuje výhodu při překonávání náročného terénu. Nicméně efektivní lokomoce v náročném terénu vyžaduje spolehlivou detekci došlapů, jedině tak je možné zajistit, aby robot přizpůsobil své chování okolnímu prostředí. Pro cenově dostupného robota vybaveného pouze poziční zpětnou vazbou byl v rámci této diplomové práce vyvinut odhadce kontaktu nohy, který jsa založen na metodách strojového učení jest podroben experimentálnímu ohodnocení. Tři metody strojového učení jsou použity k předpovědi polohy nohy robotu na základě posloupnosti posledních nastavených a změřených pozic nohy. Konkrétně byla použita regrese nejmenších čtverců (Ordinary Least Squares regression), regrese nejmenších čtverců s druhořádovými polynomiálními příznaky a třívrstvá dopředná neuronová síť využívající ReLU aktivační funkci. Navržené metody jsou experimentálně vyšetřovány s ohledem na přesnost navržených modelů, robustnost vůči změnám parametrů systému, velikosti souboru nasbíraných dat použitého k učení, využívaným výpočetním prostředkům a na závěr nasazeny na šestinohou robotickou platformu SCARAB za účelem detekce kontaktu nohy. Navíc, statistické vlastnosti souboru nasbíraných dat použitých k učení, citlivost regresorů vůči rychlosti pohybu nohy v souboru nasbíraných dat, v neposlední řadě vliv změřených a nastavených pozic ve vstupních datech regresorů na jejich předpovědi je detailně zkoumán za účelem objasnění neúspěšného nasazení regresorů na robotickou platformu. Na závěr byl uměle vytvořen scénář kontaktu nohy s cílem prozkoumat jak regresory reagují na nahodilé kolize. Výsledky této práce ukazují, že navzdory slibnému výkonu regresorů v prvních ohodnocovacích scénářích, nejsou navržené regresory vhodné k detekci kontaktu, jelikož se buď přeučí a nebo neposkytují dostatečně spolehlivé predikce.

Klíčová slova: dynamika systémů, vícenozí kráčející roboty, řízení pohybu

Abstract

The enhanced rough terrain traversability of the multilegged robots is directly connected to their relative complex morphology in comparison to the wheeled or tracked robots. However, the efficient rough terrain locomotion requires reliable contact sensing necessary to adapt robot behaviour and cope with the terrain irregularities. In this thesis, the learnable leg contact estimator for an affordable hexapod robot with positional feedback-only has been developed and experimentally evaluated. In particular, three light-weight machine learning approaches, namely Ordinary Least Squares regression, Ordinary Least Squares regression with second-order polynomial features and three-layer feed-forward neural network with the Rectified Linear Unit activation function, are used to predict leg position based on the sequence of the measured and set positions of a particular leg. The proposed methods are investigated experimentally w.r.t. the model precision, robustness to the parameter changes, size of the training set and computational requirements and experimentally deployed to the SCARAB hexapod platform to detect foot-contact. Additionally, the statistical properties of training datasets, the regressors sensitivity to the leg movement speed, the effect of the measured and the set positions on the prediction are examined to explain the unsuccessful deployment. Finally, the artificial foot contact scenarios have been designed to examine how the regressors react to arbitrary collisions. The collected results show that despite the promising performance in the initial scenarios, proposed regressors are not suitable for the contact detections since the regressors either overfit or provide unreliable predictions.

Keywords: system dynamics, multi-legged robots, locomotion control

Contents

Chapter 1

Introduction

Robotics is a rapidly developing engineering branch aiming to substitute human factors with automation, nowadays further accelerated by the ongoing pandemic. While industry mainly focuses on stationary manipulators extensively used for automated manufacturing, mobile robots can be deployed for a broader range of tasks outside of the well-structured factories to the harsh unstructured conditions addressed by field robotics. A few examples of recent mobile robotics deployments include but are not limited to, NASA Perseverance rover and Ingenuity drone successfully landed on the Mars surface [1] depicted in Fig 1a, New York City Police Department deployed Boston Dynamics Spot walking robot to crime scenes reconnaissance [2], or increasing demand for autonomous food delivery robotic solution during the pandemic situation [3] depicted in Fig. 1c, are only a few examples of recent mobile robotics deployments.

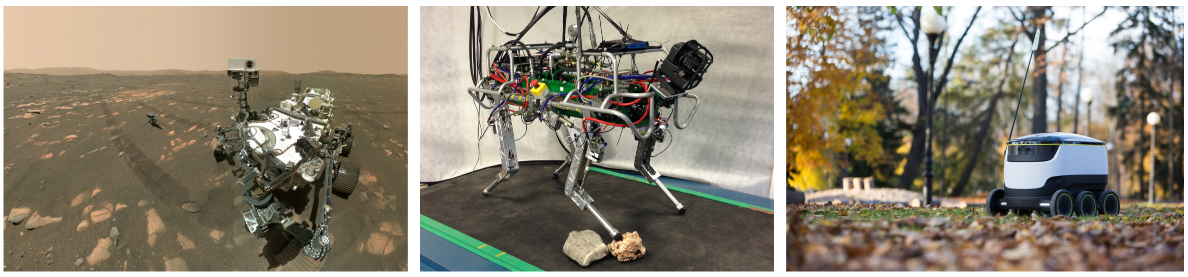

(a) NASA Mars roover.[1] (b) Hydraulic Qadruped robot.[2] (c) Food delivery robot[3]

Figure 1: Mobile robotic platform examples.

Aside from specialized platforms, such as extraterrestrial rovers, most of the mobile robot types require infrastructure. For example, roads are necessary for autonomous cars, magnetic lines and localization markers are used for navigation of logistic robots in storage houses, or runways are required for takeoff and landing of fixed-wing unmanned aerial vehicles (UAVs) such as autonomous planes. Even though tracked robots are not as infrastructure-demanding, terrain traversed by them is usually heavily damaged. On the other hand, multi-legged robots require no specialized infrastructure to traverse challenging terrain easily as they mimic the natural way of transportation. Their relatively complex morphology enables their enhanced rough terrain locomotion in comparison to the wheeled or tracked robots see Fig. 2. Because of their morphology with a high number of degrees of freedom, specialized and more complex methods are necessary for their control. On the other hand, system redundancy allows controllers to choose particular foothold and leg trajectories, enabling precise terrain negotiation utilized, for example, in humanitarian demining [5].

In this work, we will focus on developing a learnable leg state estimator for an in-house constructed hexapedal platform SCARAB (Slow-Crawling Autonomous Reconnaissance All-terrain Bot) depicted in Fig. 3. The robot represents an affordable research platform built from off-the-shelf components. It comprises the trunk that hosts the main control computer, batteries and exteroceptive sensors, and

[1]courtesy of NASA: `https://mars.nasa.gov/resources/25790/`

[2]courtesy of Nobili et al. [4]

[3]courtesy of Starship Technologies: `https://www.starship.xyz/follow/`

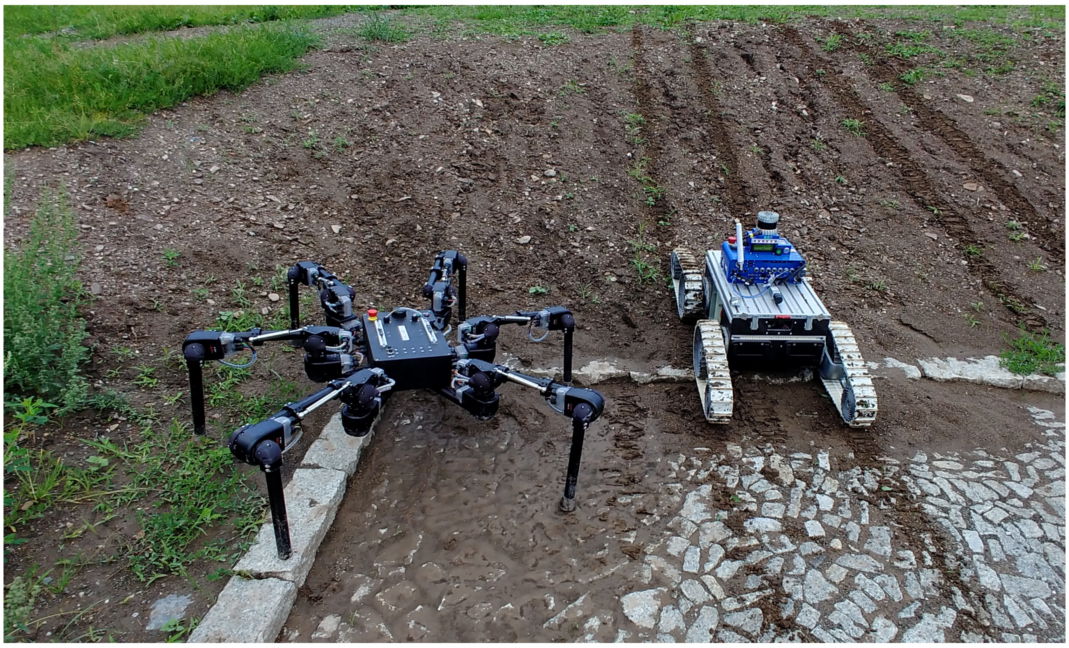


Figure 2: Differences between the impact of a legged robot (Lily the hexapod robot by HEBI Robotics) and tracked robot (MARV by Jettivision) on the sloped muddy terrain.

six legs, each with three joints actuated by the affordable Dynamixel AX-12 servomotors. However, selected actuators have only the P-type position controller and provide only positional feedback. The positional feedback only limits the locomotion control of the robot to monitoring virtual elasticity in joints since this is the only way how to traverse harsh terrain without additional sensors [6] and without excessive torques in the joints that may damage the actuators. Used locomotion controller [7] is based on an analytical dynamics model of the robot.

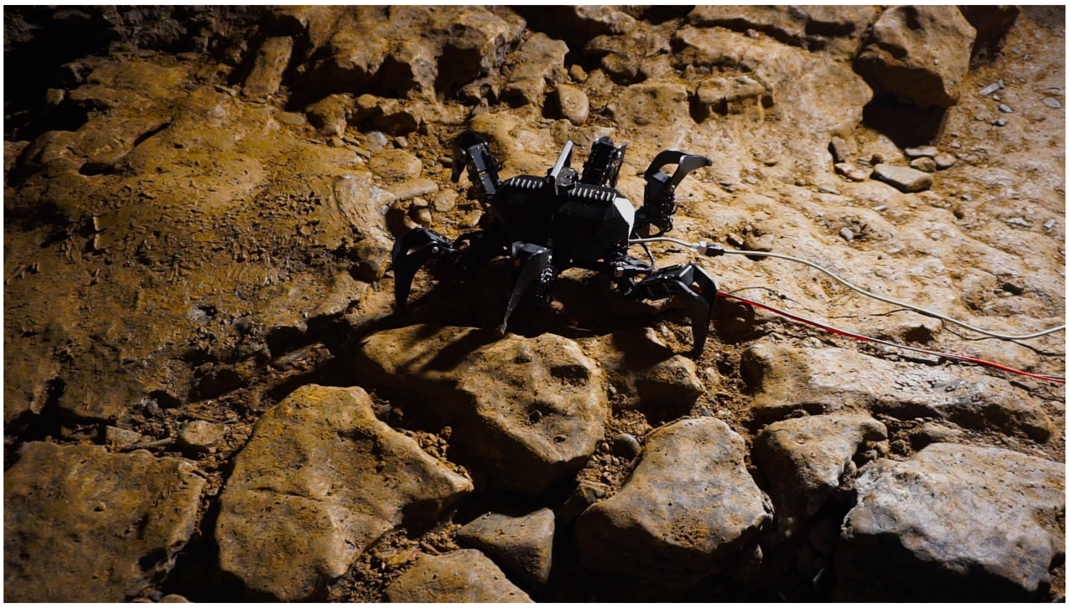


Figure 3: Hexapod walking robot SCARAB (Slow-Crawling Autonomous Reconnaissance All-terrain Bot) in the underground environment.

Although the process is laborious, one can identify the parameters of the dynamic model, but the model itself can not take into account all the possible environmental factors and other processes affecting the robot. Until we participated in DARPA SubTerrenian Challenge [8], our robots rarely left a controlled habitat of the robotic laboratory. Consequently, we did not expect nor focused our research on harsh underground conditions. Mud deposits increasing weight of the robot legs, dust or fine mud increasing servo friction hand in hand with possible leg weight reduction as a result of leg damage are only a few examples of otherwise hard to predict environment factors we have encountered. These conditions are expected to significantly change the system dynamics during the deployment; therefore, we cannot rely on the analytical model previously identified in a controlled laboratory environment.

Further, our SCARAB fleet currently consists of four robots with additional units on the way; hence the time spent identifying the dynamics model will further lengthen with the size of the robot fleet. Additionally, fused filament fabrication (FFF) 3D printing technology was used to speed up the development process and unlock instant platform improvements based on experience from deployments. Even though all robots in the fleet share similar morphology and leg structure, there are slight differences in robot parameters caused by constantly improving settings of the fabrication method, materials used, and robot design itself.

Hardly predictable real environment and variation in robot parameters emphasize the importance of solution robustness to maintain robot performance during deployment and poses engineering challenges motivated by real-life scenarios. Therefore we suggest exchanging analytical model by model based on machine learning that will cope better with changes in robot parameters and can be re-learned online. This learnable system can be easily deployed on multiple hexapod walking robots without any modifications other than re-learning. Aside from removing laborious system identification and accelerating the production process, machine learning approaches look for relations between input and output data; therefore, these approaches model all relations, including those neglected by analytical models.

Moreover, we aim to obtain additional information from the robot as we have found out that the state of the robot leg is essential to ensure reliable locomotion over rough terrain [7, 9]. This information shows to be very important for other legged platforms [10–12] but force-sensing utilized by others is not suitable for our platform because we lack required sensor inputs [6]. Nevertheless, we believe that information needed to reveal the leg state is encoded in positional data and the robot's attitude. Therefore, we want to create a learnable model that is capable of detecting leg contacts and estimating leg state, assessing whether the leg is supporting the body or not. Reliable contact assessment is essential to maintain the attitude of the robot in challenging terrains.

This work presents the learnable state estimator developed using machine learning methods while focusing on the foot-contact detection capabilities. Three lightweight machine learning methods are experimentally evaluated and compared to the analytical baseline model [7]. Based on the evaluation results, the most suitable method is deployed on the SCARAB robot. The performance of the regressors deployed on the robot in the rough-terrain traversal scenario exhibit behaviours that were not seen during the evaluation of the regressors. Therefore, we have put further effort into explaining the inferior behaviour of the regressors by thoroughly examining the statistical properties of training datasets, the regressors sensitivity to the leg movement speed and the effect of the measured and the set positions on the prediction accuracy.

The rest of this thesis is organized as follows; Chapter 2 surveys the state of the art methods used for robot dynamics modelling and state estimation, Chapter 3 outlines the necessary background, including platform and locomotion control description, underlying dynamic model and problem statement, proposed methods are described in Chapter 4, Chapter 5 reports results of numerous experimental evaluation and deployment scenarios, and Chapter 6 conclude this work.

Chapter 2
State of the Art

This chapter presents the state of the art methods relevant for the state estimation and robot control of the multi-legged platforms. It is divided into two main parts. The first part introduces methods used for robot control with emphasis on machine learning-based approaches. The second one overviews state estimation in mobile robotics with a focus on multi-legged platforms. We chose this division since, to our best knowledge, there are no approaches including both model learning and state estimation.

2.1 Robot Control and Machine Learning

Nguyen-Tuong and Peters [13] describe the model as a collection of essential information about the robot system and a description of the agent's influence on the system, both kinematic and dynamic level essential for planning, control and many other applications.

Based on the amount of prior information about the system, we can distinguish white-box, grey-box and black-box control methods. *White-box* (also referred by some authors as clear-box) *controls methods* have complete information about the system and its properties. Conventional control methods such as differential-algebraic equations [14], state-space models [15] or modular controller design [16] are a few representative examples of white-box control methods. *Black-box control methods* have no prior information about the system at all and deduce relations between system inputs and outputs based on provided data. Therefore machine learning methods can be deployed to approximate this relation. To name a few, artificial neural networks (NN) [17], reinforcement learning (RL), gaussian process regression (GPR) [18], Local Weighted Projection Regression (LWPR) [18] can be used for such tasks. *Gray-box control methods* combines white-box and black-box, commonly exploiting knowledge about systems to design white-box and adding black-box to compensate for unknown non-linearities [19–25].

The profound difference between the white-box and black-box model is that the white-box exploits prior information about the system, whereas the black-box model has no assumption about system structure. To successfully design a white-box model, one must understand the system and its underlying physics; on the other hand, the black-box design needs almost no information about the system itself. The White-box model incorporates only relations explicitly defined in the design process, whereas the black-box methods attempt to reflect all relations between input and output. These attempts may fail because of overfitting to data since black-box models can imitate system behaviour at best [13] and the reason for failure might not be easy to reveal since the explainability of black-boxes is limited or none, and it is nowadays actively researched topic [26]. Grey-box control methods combine white-box and black-box models such that the advantages of both approaches are exploited (prior information about the system in the case of white-box and the ability to compensate for unknown non-linearities in the case of black-box). In the following subsections, a detailed description of selected methods is provided.

2.1.1 White-Box Control Methods

White-box control methods are centred around a model based on physical principles. Creating a precise model requires identifying relationships between relevant variables of the plant. This requires both understanding underlying physics and advanced knowledge about the system itself [27]. These requirements are the main downside of the white-box models; their accuracy is directly connected to their complexity. Therefore unknown non-linearities are never included in the model.

Eventhough it is possible to use genetic algorithms (GA) described in Section 2.1.3 to identify model parameters [19, 22], white-box control methods cannot be learnable. However, we will describe selected methods for the sake of completeness since we believe that they help us broaden our insight into the control theory. On top of that, white-box are an integral part of the grey-box methods. White-box can be described by a set of an ordinary differential equation, a set of differential-algebraic equations [27] or analytically using Euler-Lagrange formulation [28]. The Euler-Lagrange formulation is used by baseline locomotion controller [7] to describe the leg dynamics. Regardless of the used method, general parameters of these equations must be identified using system identification methods, some of which can be automated using adaptive control [29], machine learning such as neural network-based SANARX [30] or previously mentioned genetic algorithms.

In the context of controlling multi-legged robots, Hwangbo et al. [31] mention two common approaches: modular controller design and trajectory optimization. *Modular controller design* breaks control problem into individual modules. These modules are assumed to be independent of each other, and therefore, each module can be designed and tuned individually. Individual modules are limited in accuracy since approximation of operational state is valid within a small region; therefore, significant compromises have to be made, such as slow acceleration, fixed body pose, or limited velocities of limbs. On top of that, designing modular controllers is laborious and advanced knowledge of the robotic system is required to hand-tune each new module, new robot, or even new manoeuvre [31].

Trajectory optimization approaches divide the controlling process into two steps: trajectory planning and trajectory tracking. Trajectory planning uses rigid body dynamics and numerical optimization to compute an optimal path, while trajectory tracking follows the optimal trajectory. While being more automated, trajectory optimization approaches perform worse than modular controllers. These methods remain computationally demanding or beyond the capabilities of current optimization techniques due to the complexity of the multi-legged robots domain. Even though it is possible to reduce their precision or run them on powerful external machines, they still require tuning and can produce suboptimal solutions [31].

2.1.2 Black-Box Control Methods

Black-box control methods represent a helpful alternative to the conventional white-box approach to the system control models. The main feature common to these methods is estimating the model directly from the data using system input and output. This direct approach allows taking into account all relations between input and output, including unknown non-linearities that are usually omitted by standard physical-based modelling [13]. However, new challenges arise if one considers using black-box control methods. Since these methods are data-oriented, the quality of data affects the accuracy of the resulting model. Data used to train black-box models have to cover as many regions of a model state space as possible, even though covering complete state space is not possible [13]. Though covering a representative portion of the state space is usually sufficient for black-box methods to generalize from provided data, and keeping the data-rich and balanced decrease the chance of model overfitting [13,22]. Black-box models tend to have higher model complexity than the white-box approaches [22]; therefore, the dimensional reduction is applied. The core idea about dimensional reduction is based on the assumption that helpful information often lies in the low-dimensional manifold of the original input space. Using dimensional reduction decreases both the complexity of the model and, more importantly, a chance of overfitting [13] because it removes dimensions that are potentially correlated with output but not causally connected. Black-box learnable models are used in a variety of robotic engineering tasks such as inverse dynamic control [18], inverse kinematics of redundant manipulator [32], robot manipulation of 7-DoF redundant manipulator [33], slippage prediction and control of multi-legged robot [34] or legged robot locomotion over rough terrain [35].

Based on inner structure and the way how black-boxes are fitted to the data, we can further distinguish the following: off-line and online methods, parametric and non-parametric methods, global and local methods.

In *off-line learning*, data are collected on the platform, trained and then deployed on the system. Any training occurs separately from deployment. On the other hand, in *online learning* scenario, either the black-box model is trained during deployment on the system, or the model is pre-trained off-line and then improved during the deployment. Online learning generalizes better and adapts the model to time-dependent changes since these approaches explore a more significant part of the state-space during the deployment that is otherwise not covered in the training data [17]. Still, it may experience the overfitting, concept drift [36], or catastrophic forgetting [37] that renders the model unusable.

Parametric methods such as neural networks have a fixed number of parameters set before training, usually based on insight into the particular method and domain. On the contrary, *non-parametric methods*, e.g., gaussian process regression and locally weighted regression have no parameters, and the model structure adapts to the data complexity. Non-parametric methods can be deployed to robot control [38].

Global methods (also global regression techniques), such as linear and polynomial models, or neural networks, use all available training data to construct a single global prediction model [39]. Even though conventionally, neural networks have fixed structures, there are Neural Networks that can change their structures dynamically, such as reservoir computating [40] or echo state neural networks [41]. *Local methods* (also local regression techniques) such as locally weighted projection regression [42] use multiple models to estimate the underlying behaviour of the system in the local neighbourhood around a query input point [13]. Different structures can be assumed for individual local models based on the problem complexity. Local regression techniques are widely used in robotics [42, 43] for model learning thanks to their simplicity and computational efficiency [13]. Additionally, local regression techniques can cope better with less smooth functions than the global methods because of their local-based structure.

Two examples of non-parametric methods are used for black-box model learning; Gaussian process regression and locally weighted projection regression. The former is global, and the latter is a local method. *Gaussian process regression (GPR)* [44] is a non-parametric probabilistic global method that uses the sample data to construct a kernel-based interpolation basis utilizing user-provided kernel functions to model system behaviour. While being the cutting edge of the global regression method, GPR suffers from poor scalability since the memory and computational complexity scale cubically with the size of the training data [13]. *Locally weighted projection regression* [42] combines local methods with dimensional reduction. Firstly, input data are projected to space with a lower dimension and then local methods are applied [13]. Local and global methods can be combined to exploit the strength of probabilistic methods as done with GPR in [18].

Nguyen Tuong and Peters [13] present three types of models; forward, inverse, and mixed models and additional multi-step prediction models. Similarly to the forward dynamics and kinematics, *forward models* predict the next state of the robot dynamics system given the current state and current action such as model reference adaptive control (MRAC) [45].

Inverse models, on the other hand, predict the action required to transit the model from the current state to the desired future state. A common example is the inverse dynamics model used by computed torque robot control [46] and inverse dynamics control [47]. Inverse model training is straightforward if the relationship between the input and the output of the system is well defined [13]. In that case, standard regression techniques can be applied, e.g., the least square methods [27], neural networks [48], or statistical approximation techniques [44]. Otherwise, indirect modelling where particular error measurements drive model learning is deployed, or distal teaching where the forward model guides inverse model learning can be deployed [13]. As emphasized by Nguyen-Tuong and Peters [13], obtaining data is challenging as a dataset have to be sufficiently rich and persistent excitation causes issues in online setups. Some approaches, such as linear regression [49], cope badly with non-linearities of real inverse dynamics of complex systems because of their fixed parametric structure. Therefore neural networks, static non-parametric model learning, or local methods are used.

Mixed models combine both forward and inverse models to cope with nonuniqueness (ill-posedness) of the inverse model. Even though mixed models are widespread in the neuroscience community, they are not yet deployed in robot control [13].

Multi-step prediction models produce time-sequence of future system outputs without available future measured system states. Deploying is especially challenging since the error made in the past can propagate to future predictions. Multi-step prediction model approaches known as model predictive control (MPC) [50] are widely used in industry. White-box takes on MPC such as ARX or ARNAX are too limited for complex robot system [13]; hence, NN and probabilistic methods are used for such tasks.

In their survey, Jin et al. [17] compile numerous neural network approaches used to control robot manipulators, both fixed-base (redundant, parallel, cable-driven) and floating-base (mobile). *Feed-forward neural networks* [51] are widely used to solving both dynamics and kinematics problems of controlling robot manipulators. Authors distinguish two types of feed-forward NNs: feed-forward NN based on backpropagation (BP) and feed-forward NN with radial basis functions. *Feed-forward NNs using back propagation* are neural networks using sigmoid as activation function without any cycle in their structure. Even though three layers are sufficient to approximate any function, multiple layers can be used to create deep neural networks. Backpropagation adjusts neural network parameters (weight of connections between neurons) based on loss function using the difference between desired NN output and actual output. It is possible to use feed-forward NN to control flexible-joint manipulator [52] or to set-point control 2-DoF planar manipulator [53].

Feed-forward NN using radial basis function (RBF) consists of three layers. The main idea behind this NN is to use radial basis functions to map linearly inseparable samples to higher dimensions through a non-linear transformation to become separable by linear function in higher dimensions. Feed-forward NN using RBF are being used to solve the dynamic and kinematic problem of robot manipulator [54], compensation for non-linear dynamics in contouring control [55], or cancelling out adverse effects of friction of two-joint manipulator [56].

Recurrent neural network (RNN) adds a new separated recurrent hidden layer connected to the regular hidden layer. Bi-directional information flows simultaneously from the hidden layer to the recurrent hidden layer and vice versa. Tian et al. [57] used RNN to control contact forces and position between the end effector and surface for the flexible manipulator.

Long Jin et al. [17] accent need for online learning since implementing this feature cause manipulator to deal with unexpected factors encountered during operation. Therefore, authors consider RNNs since they have a feedback mechanism since they can adapt over time while they do not need off-line learning [55]. However, RNN fails to reveal long-term dependency in the input data; therefore, Jin et al. see potential benefit in using long-short-term memory NN [58] that are capable of learning to classify, process, and predict time series even if considerably long time lags of unknown size are present between events.

Neurons in *Hopfield neural network (HNN)* are not organized in a layer but compose a fully connected graph where every neuron is connected to all others, excluding itself. Each neuron has two-state (either activated or not), and Hebbian learning [59] is used to train them. Although HNN may reach only local optimum [17], Ding and Chan [60] successfully used them to solve kinematics of redundant manipulators for obstacle avoidance.

Spiking neural networks (SNN) are more associated with a real neuro-system than previously mentioned types. SNN show their capabilities for solving time-dependent patterns to control a 4-DoF manipulator [61] or to be deployed as part of target tracking controller for autonomous mobile robots [62].

Central pattern generator (CPG) neural network produces rhythmic patterns without needing sensory feedback used for locomotion control [63]. The main problem of CPGs in locomotion control of multi-legged robots is in a limited range of possible motions given by the rhythmic patterns, although they cope well with disturbances. Their parameters are also hard to tune to produce a viable rhythmic pattern, and therefore CPGs are often hand-tuned or entrained in simulation using the reinforcement learning [63].

Echo state network (ESN) exploit randomly generated reservoir replacing hidden layer in a three-layer neural network. The number of neurons in the reservoir is related to the complexity of the problem, and different reservoir states are being recorded over time as input changes. ESN was used to perform precise position control of robot manipulators by authors of [64].

2.1.3 Grey-Box Control Methods

Grey-box methods combines white-box and black-box into single model to control plants in various areas ranging from robotic manipulators [19–21], coupled water tanks [22, 23], fermentation chambers [25] to satellite attitude control systems [24], neutron beam [23]. Using the white-box model, one can exploit knowledge about the system while black-box handles unmodeled non-linearities of the system. This is especially useful if underlying dynamics is unknown as presented in [25] where the black-box component was designed to model bacteria growth in the Acetone-Butanol-Ethanol fermentation chamber. As pointed out in a review by Ljung [27], grey-box control methods have a rather broad range of touching white-box on one end of its spectrum and black-box approaches on the other.

Genetic algorithms (GA) were built upon Darwin survival-of-the-fittest principle. The user provides a fitness function that associates numbers to each possible solution represented as a set of parameters (chromosome). Optimization occurs iteratively in steps referred to as generations. Each generation, part of the current population, is selected based on selection rule to produce new solutions (offsprings) combining their chromosomes based on crossover rules, optionally local search methods (mutation) can occur. Based on the fitness of the last generation and offspring, solutions are selected for the next step. Using genetic algorithms is not unusual since evolution-based algorithms have been proven useful for multi-criteria optimization as mentioned by Nemes et al. [19] and since candidate solutions provided by GA are Pareto's optimal, engineers can examine different trade-offs as pointed out by Tan et al. [23]. Nemes et al. [19] used GA to optimize both white-box model parameters and complexity of the black-box model implemented by fuzzy logic systems to model the dynamics of the robot manipulator. Tan et al. [23], on the other hand, demonstrates that GA can overcome the local search method (Quasi-Newton's algorithm) for both coupled liquid level system dynamic model and the neutron beam-based thin-film reflectivity detecting system.

Genetic algorithms are not only options how to identify white-box parameters; Wernholt and Gunnarsson [21] used conventional techniques to identify initial values for rigid body dynamics, friction, and joint flexibilities of their manipulator to ensure convergence to global optima and Oaki and Adachi [20] used a decoupling identification procedure to identify parameters on their horizontal two-link robot with elastic harmonic drive gears. Rogers et al. [22] combines models in a novel way by connecting both the output of the white-box and grey-box model input to the Gaussian process regression [44] used as black-box part of theirs model.

A rather unusual grey-box design was presented in [24] for diagnosis fault estimation of reaction wheel in satellite attitude control system. Authors propose a novel approach called Grey-Box Neural Network Models (GBNNM), combining multi-layer perceptron neural network and integrators to approximate both non-linearities and dynamics of the plant.

Prada et al. [25] used mixed-integer optimization to identify unknown dynamics of bacteria growth factors in their Acetone-Butanol-Ethanol fermentation plant. Mixed-integer optimization selected a combination of user-provided functions providing best-fit and ensuring physical coherence.

2.2 State Estimation for Legged Robots

Legged robots are typically equipped with proprioceptive sensors such as joint encoders and IMUs, occasionally with contact sensors, whereas exteroceptive sensors (cameras and LIDARs) are more common for humanoid robots [11]. Based on sensory equipment, three stages of state estimation are distinguished by Hartley et al. [11]: kinematic odometry only, kinematic odometry with additional sensors, and observability-constrained strap-down ErEKF (Error Extended Kalmana Filter). These approaches are further described in the following paragraphs, followed by other rather unusual approaches to state estimation.

Kinematic odometry only approaches estimate robot position and orientation relative to the original pose using joint encoders and contact measurements if available; measurements are then fed into the kinematic model to produce a relative transformation of robot frame. Kinematic odometry only approaches assume that the legs supporting the body stays fixed relative to the ground. However, in practice, this assumption is often violated, e.g., the point-shaped point-shaped foot-tip can rotate on the ground without joints moving [11], or individual legs can slip on a slippery surfaces [65]. Kinematic odometry-based approaches are easy to implement but noisy. According to Roston et Krotkov [65], kinematic odometry drift is caused by kinematic model inaccuracies, joint encoder noise, and foot slippage; hence, it is impractical for mapping and autonomy tasks.

To increase the accuracy of kinematic odometry, additional sensors such as IMUs, gyroscopes, cameras or LIDARs can be considered. Lin et al. [66] use kinematic odometry, contact sensors and multiple IMUs to estimate the state of the RHex hexapedal platform using EKF with a sequence of continuous-time dynamical models. Chitta et al. [67] utilize proprioceptive sensors and a particle filter for tactile sensing through kinematics to localize LittleDog quadruped robot in a priori known map. Cobano et al. [68] fuse measurements from kinematic odometry, global position system (GPS) and IMU to localize SILO4 quadruped robot in outdoor environment. Khalil et al. [69] deployed particle swarm optimization to tune EKF used for pose estimation of Corin hexapedal platform. They demonstrate the robustness of their approach to wall-walking.

Finally, *observability-constrained strap-down ErEKF* [70] combine inertial and kinematic measurements. This method extends strap-down ErEKF mentioned earlier; therefore, no dynamics model is needed since the IMU integration model is being used. This method can compensate for slight random Brownian movement of leg-tips causing drift in the case of simple kinematic odometry-based approaches., but additional information about leg contacts with terrain is required. If leg contacts are provided, filter equations are general enough that the method can be deployed on multiple legged platforms. For example, Bloesch et al. [71] used this method to detect foot-slip on StarlETH quadruped and Lubbe et al. [72] fuse data from onboard IMU and force sensors on the foot-tips to estimate PhantomX hexapedal platform pose. A similar approach was developed by Yang et al. [12] using Square Root Unscented Kalman Filter. Authors combine low-grade IMU, leg joints encoders and torque sensors to estimate the state of hexapedal platform HEBI Daisy.

Fallón et al. [73] developed PRONTO [74] modular framework for state estimation. Fallón et al. [73] used two different inertial measurements units, leg kinematics and LIDAR, to estimate the state of the humanoid bipedal Boston Dynamics Atlas robot. Position, orientation, linear and angular velocities were estimated and using Gaussian Particle Filter, drift-free localization was achieved within a priori known map. Camurri et al. [10] deployed the PRONTO framework on HyQ quadrupedal platform. A logistic regression classifier was used to estimate necessary information about foot contacts from force sensors. Nobili et al. [4] developed a method to estimate position and velocity of torque controlled HyQ quadruped platform while using the approach by Cumarri et al. [10] to estimate foot-contacts. The proposed method used inertial measurements, leg kinematics, stereo vision and LIDAR data with different latencies and frequencies to localize the robot under different light conditions and while traversing varying terrains.

Chilian et al. [75] used an indirect feedback information filter to estimate pose, velocity and sensor biases of the DLR Crawler hexapedal robot. Their method combines previous and current robot states while fusing measurements from leg odometry, IMU and stereo camera. Hwangbo et al. [76] deployed a probabilistic model (Hidden Markov Model) for foot-contact estimation using only leg odometry of StarlETH quadruped robot. Several insight notes were mentioned by the authors. A leg that has high acceleration and velocity is likely not colliding with terrain. A leg that is high off the ground is likely not colliding with terrain. This insight was used to create a transition model: high speed in the air means no contact, and high speed on the ground means slippage. Bloesch et al. [77] propose a two-state implicit filter to estimate pose based on the previous and current state. Their method that does not require an explicit dynamics model was deployed on the planar robot, manned aerial vehicle and blind quadruped. Wisth et al. [78] used the factor graph method combining joint sensing (position, velocity, torque), IMU readings and camera images to estimate ANYmal quadruped robot pose, linear velocity and sensor biases while the robot was walking, trotting, crossing obstacles and ascending staircase.

2.3 Foot Contact Detection for Legged Robots

A straightforward method of foot contact detection is using contact sensors such as micro-switches or direct force measurements using force-sensitive resistors (FSR) and multi-dimensional force sensors. However, micro-switches and FSR are fragile and have limited area and direction of sense, and reliable multi-dimensional force sensors are expensive, heavy and rarely used on walking robots. Therefore the contact force is indirectly estimated using force/torque sensors inside robot joints [10,12,79,80]. Bledt et al. [79] extend momentum-based disturbance observer to discrete-time and fuse force estimates with servomotor encoders and leg dynamics model using Kalman Filtering and Probabilistic models to estimate the probability of the foot contact. Camurri et al. [10] use joint position, velocity and torque to estimate ground reaction forces (GRF) of the HyQ robot legs. The probability of the foot contact is obtained from GRF using logistic regression. Hwangbo et al. [80] use probabilistic methods to estimate foot contact (including maintaining the contact) using the inertial measurement unit (IMU), joint encoders and modelled generalized coordinates and velocities. Yang et al. [12] utilize torque sensors of the Daisy hexapod robot to estimate foot force and express the probability of foot contact relative to the minimal and maximal force estimated during the trial run.

However, force/torque measurement is not always available since there is a significant price and size gap between servomotors capable of torque measurements and those unable to do that. Therefore, Adachi et Nagasaka [81] measure base-frame angular velocities and current in each servomotor of the hexapod robot. A three-layer neural network with ten neurons in the hidden layer is trained to detect obstacles while the robot leg was descending. However, single obstacle height is considered by authors, and the data input size and region were hand-picked regarding the particular obstacle instance, which is very limiting, especially when walking in rough terrains where contact may appear anywhere along with the leg morphology and path. Xu et al. [82] use servomotor current and the Jacobian matrix to estimate foot-tip force to develop adaptive gait for Octopus III parallel-actuated hexapod. The proposed gait uses force estimation and reflexes to detect ground and avoid obstacles. Yang et al. [83] combines spring-loaded inverted pendulum (SLIM) model of a miniature quadruped robot with IMU measurements and actuator data under the Kalman Filter (KF) framework. The contact is detected whenever the difference between the optimal solution produced by KF and measured data exceed the threshold.

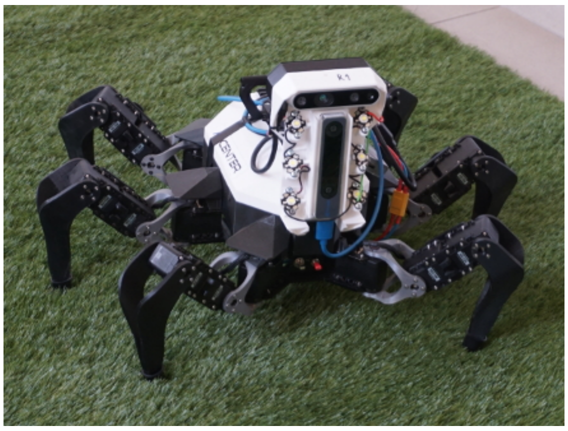
Nevertheless, adding sensors increase the complexity, price and failure rate of the robot unnecessarily. Therefore Faigl et Čížek [7] propose a minimalistic approach for adaptive locomotion using servo position encoders only. Assuming proportional relation between joint torque and position error, authors adaptively set the position error threshold based on the position predicted by the inverse dynamics model for the collision-free motion of the leg. The force threshold-based approach is the most straightforward method of detecting foot contacts suitable for an affordable platform such as

SCARAB with positional feedback and no additional sensors. Additionally, Haddadin et al. [6] support this method as the only sound approach for available data. The baseline approach [7] uses a white-box model, which parameter identification is challenging and does not generalize well for situations mentioned above of extensive changes in the leg or joint parameters. This baseline approach is further described in following section and extended as described in Section 4.

Chapter 3
Background

In this chapter, we present the necessary background. Firstly, platform specifics are explained, including platform main design goal, actuator limitations and manufacturing description. Secondly, the robot controller is outlined with a particular focus on the baseline controller description. The next part describes the dynamic model used by the baseline controller. Lastly, the problem statement and the required properties of the machine learning-based model are described.

3.1 SCARAB Platform Description


(a) SCARAB with the dual-camera rig.

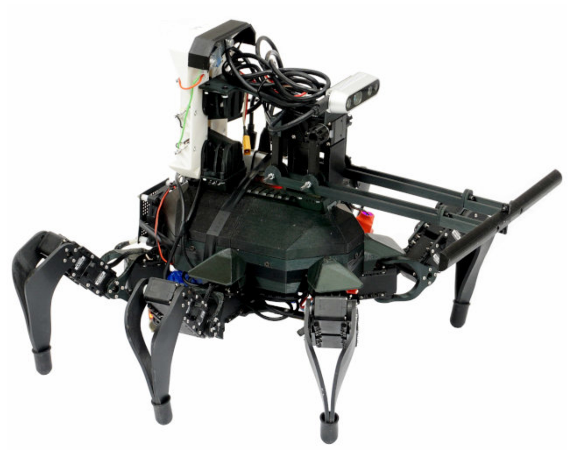

(b) SCARAB with the dual-camera rig and the bumper.

Figure 4: Hexapod walking robot SCARAB (Slow-Crawling Autonomous Reconnaissance All-terrain Bot) with the different payload suited for the particular deployment scenario.

The SCARAB (Slow-Crawling Autonomous Reconnaissance All-terrain Bot depicted in Fig. 4) is a hexapod robotic platform developed at the Computational robotics laboratory. The robot is based on PhantomX AX Mk II [4] by Trossen robotics, but over time all of the original mechanical parts were replaced, and the morphology of the robot has been altered as well; hence, a new name was chosen to differentiate the new robot. Currently, only a single steel reverse-engineered base-plate piece is used from the original PhantomX MK II platform, while all other structural pieces are 3D printed. The SCARAB platform was designed to be modular in payload because sensory equipment varies based on particular deployment goals. Payload mounted on the tactical Picatinny rail range from a camera rig with Intel Realsense D435 depth camera and Intel Realsense T265 tracking camera [8] to 3D printed force sensor (bumper) with double D435 and single T265 [84]. Nevertheless, the SCARAB robot is capable of blind locomotion over irregular terrain using solely using its actuators.

The robot morphology, number of servo motors and leg position relative to the base-frame remains the same for Phantom AX Mk II and SCARAB . However, robot legs have been modified significantly, even though the three-links structure was kept, including the link's nomenclature that reads *coxa*,

[4]`https://www.trossenrobotics.com/hex-mk2`

femur and *tibia* in the direction from the body to the foot-tip. The leg structure was altered to decrease energy consumption when the robot stands still; the coxa link was redesigned, the femur link was significantly shortened, and the tibia link was heavily modified to be 3D printable using fused filament fabrication (FFF). The FFF parameters such as material proportion used to fill a printed partially, layer height, or a number of solid layers were changed during the robot production process, affecting the final pieces' physical properties. Additionally, the tibia link cause issues with the repeatability of the manufacturing process since the printed part was heat-bended as part of post-processing.

The robot is actuated by 18 Dynamixel AX-12 servos, three per leg. Utilized servomotors with internal P-type controller operating at 1 kHz frequency are connected to the daisy chain and provide positional measurements at the maximal frequency of 100 Hz. The availability of only positional measurements limits the possible control approaches. An absence of force-torque sensors and current measurements in used servomotors prohibits using traditional methods such as impedance (compliance) control [85] or force-torque control [12]. Fortunately, a method utilizing only the position feedback of the servomotors for blind locomotion over irregular terrain has been developed by Mrva et Faigl [9] and later extended by Faigl et Čížek [7]. The locomotion control method [7] forms the baseline controller described in the following section.

3.2 Locomotion Control

Controlling actuators of multiple legs is not straightforward regardless of sensory equipment. Therefore, individual leg movement is commonly coordinated in a motion gait [86]. Gait splits control into a periodically repeated *gait cycle*. Within each gait cycle, legs alternate between *stance phase* and *swing phase*; the leg supports the body in the stance phase while the leg is moving to a new foothold in the swing phase.

Various locomotion gaits differ in several aspects, such as the number of legs supporting the body, order of leg movement, coordination of body movement, adaptability, et cetera. At any given moment, at least three legs must support the hexapod to ensure a statically stable gait for the platform. Therefore it is possible to move either one, two, or three legs simultaneously, while five, four, or three legs support the body, respectively. The number of legs transitioning to the new foothold distinguishes pentapod, wave, and tripod gaits. Order of leg movement can be predefined, or the most suitable leg is chosen to be moved in free gait. Leg and body levelling can be separated or occur in parallel. Finally, we distinguish a regular gait with prescribed leg trajectory executed in open-loop fashion and adaptive gait, which reacts to terrain irregularities based on the servomotors feedback.

The gait cycle can be divided into shorter segments, further referred *gait frames*. Within each gait frame, particular legs (called *active legs*) simultaneously move to a new foothold. As the number of legs supporting the body varies between gaits, so does the number of gait frames; two gait frames for the tripod gait, three gait frames for the quadrupod gait, and six gait frames for the pentapod gait. Intuitively, a gait with a lower number of gait frames is faster than a gait with more gait frames. However, a gait with more simultaneously moving legs is more prone to lose stability since there might not be enough legs in contact with terrain to ensure stability if some of the legs fail to maintain contact with the terrain.

Regardless of gait type, the foot-tip must precisely reach a new foothold. For flat terrain, this task is trivial as the relative distance between terrain and the robot's body remains unchanged; therefore, leg trajectory remains the same across gait cycles, and regular gait is sufficient for traversing such terrain. On the other hand, the distance between the terrain and the robot's body potentially varies significantly in irregular terrain. Consequently, open-loop-based regular gait struggles to traverse this type of terrain. Therefore, closing the feedback loop is required to detect the exact moment of foot-tip touching the ground (foot-strike) to ensure reliable locomotion.

Two possible faults of foot-strike detection can occur in adaptive gait; either premature or late foot-strike detection. In the case of *premature foot-strike detection*, the leg does not reach the ground, and it remains mid-air even though the robot assumes the leg is already supporting the body. Contrary, for *late foot-strike detection*, the exact moment of the foot touching the terrain is not detected, and the foot-tip continues to move into a new position trying to penetrate the terrain. This cause servomotors over-loading as most of the robot mass is being pushed up by a single leg. Increased load causes servomotors to overheat, and it adds to gearbox wear resulting in decreasing both the operation time and the service life of the robot. On top of that, late detection can cause loss of robot stability as the centre of gravity is affected, potentially resulting in loss of stability, fall, and robot damage. Detecting late detections is a relatively easy task as late detections start affecting the robot instantly within the same gait frame.

On the other hand, premature foot-strike detection is more challenging since the effect of legs not supporting the robot body is pronounced in later gait frames. The consequences of premature detection are at least as severe as those of late detection. Again the main issues are loss of stability and possible damaging of the actuators. Moreover, from our experience, the shock induced by the robot falling on a prematurely stopped leg strains the actuator's gearbox and significantly decreases its lifetime. This is more common for gaits with fewer legs simultaneously supporting the robot, such as the fast tripod gait. Therefore, it is of the utmost importance for the legged robot to correctly identify the moment of leg contact with the terrain or obstacles to avoid the problems arising from premature and delayed contact detection. The used approach for the detection of the obstacles is described in the following section.

3.3 Locomotion Control With Positional Feedback Only

Numerous control methods such as the force-torque control and compliance control are unavailable to our platform given the limitations of utilized servos, and other control methods require additional sensors unavailable as well. For example, foot contact estimation using foot-tip velocities [10], using leg acceleration and velocity [76], using contact sensors [87] or using force sensors on the foot-tip [72]. However, it was shown by Mrva et Faigl [9] and further extended by Faigl et Čížek [7] that it is possible to detect foot-strikes measuring position using positional feedback only.

The baseline controller proposed by Faigl et Čížek [7] we extend in this theses operates as follows. In the gait cycle, legs alternate between the *swing phase* when they are reaching a new foothold and the *stance phase* when they are supporting the body. During the swing phase, the leg transitions from the initial foothold to the new foothold while detecting the foot contact with the environment. When all legs are on the ground during the stance phase, a *body levelling phase* is introduced that adapts the robot's attitude to cope with the terrain irregularities and moves the body along the desired locomotion direction. Leg movement is stopped if the ground is detected by holding the current joint position. Once all active legs detect the ground, active legs transit to the stance phase and the body attitude is adjusted based on the new leg positions. The overall gait cycle controlling scheme is depicted in Fig. 5.

The swing phase for a single servomotor consists of two steps; trajectory interpolation and trajectory step-by-step execution with foot-strike detection. Firstly, the servomotor increment $\Delta\theta$ is computed:

$$\Delta\theta = \frac{\theta_{\text{fin}} - \theta_{\text{init}}}{t_{\text{des}}/t_{\text{con}}} \tag{1}$$

where θ_{init} is the initial servomotor position, θ_{fin} is the final servomotor position, t_{des} is the desired time it takes servomotor to reach final position, and t_{con} is the control cycle period. Both initial and final servomotor positions are obtained using leg inverse kinematics. In each step of the control loop, desired servomotor position $\theta_{\text{des}}(k)$ is updated and its value is sent to the servomotor:

$$\theta_{\text{des}}(k) = \theta_{\text{init}} + k \cdot \Delta\theta \tag{2}$$

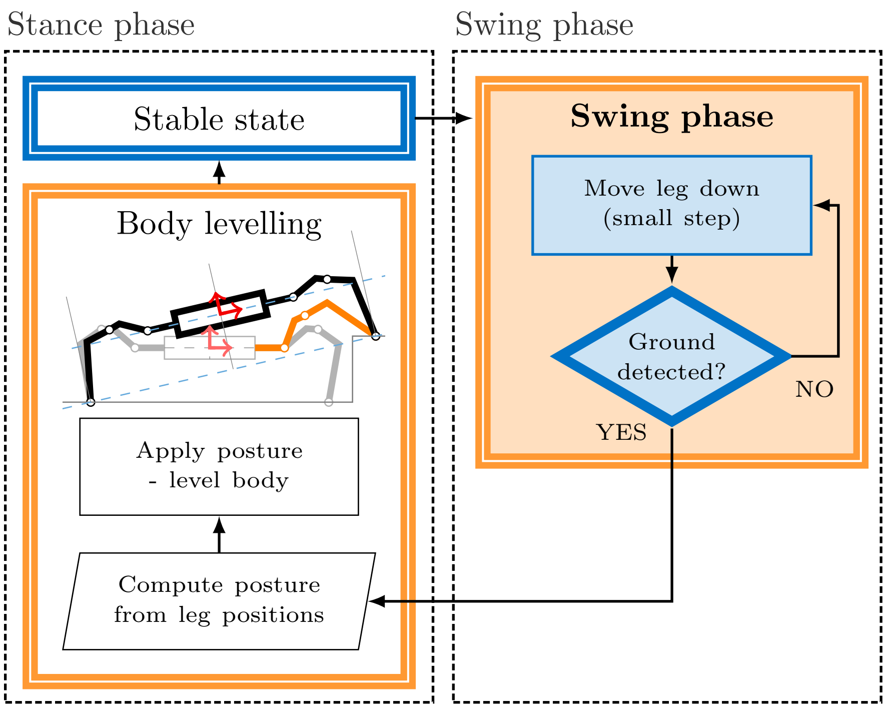

Stance phase
Swing phase
Stable state
Swing phase
Body levelling
Move leg down (small step)
Ground detected?
NO
YES
Apply posture - level body
Compute posture from leg positions

Figure 5: The overall view of a gait cycle for baseline controller [7]. The *body levelling phase* is presented to the stance phase, and the foot-contact is checked when a leg is descending in *Swing phase*. The implementation of ground detection is different for our and baseline particular approach.

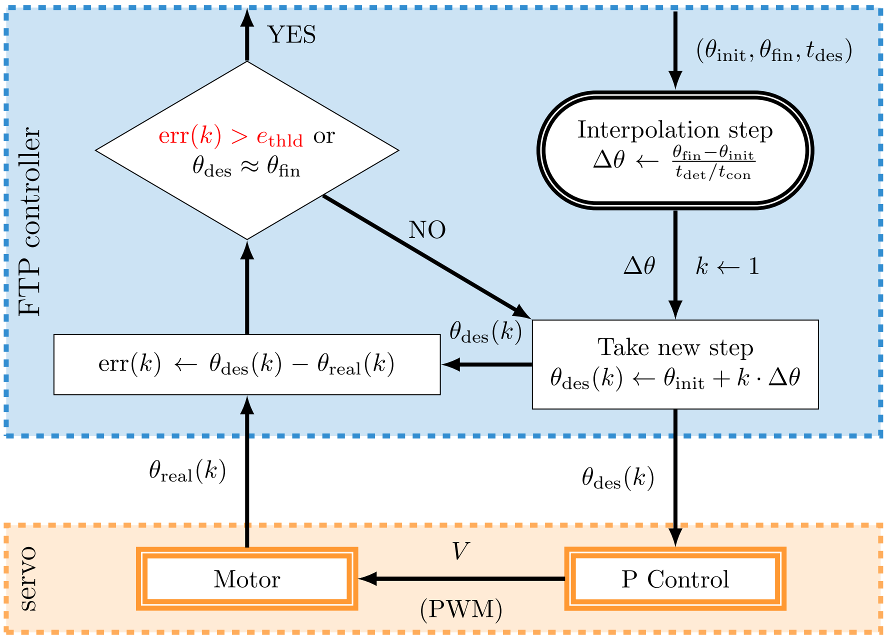

YES
$(\theta_{\text{init}}, \theta_{\text{fin}}, t_{\text{des}})$
FTP controller
$\text{err}(k) > e_{\text{thld}}$ or $\theta_{\text{des}} \approx \theta_{\text{fin}}$
Interpolation step $\Delta\theta \leftarrow \frac{\theta_{\text{fin}} - \theta_{\text{init}}}{t_{\text{det}}/t_{\text{con}}}$
NO
$\Delta\theta$
$k \leftarrow 1$
$\theta_{\text{des}}(k)$
$\text{err}(k) \leftarrow \theta_{\text{des}}(k) - \theta_{\text{real}}(k)$
Take new step $\theta_{\text{des}}(k) \leftarrow \theta_{\text{init}} + k \cdot \Delta\theta$
$\theta_{\text{real}}(k)$
$\theta_{\text{des}}(k)$
servo
Motor
V
(PWM)
P Control

Figure 6: The ground detection overview of the TFP-based controller by Mrva et Faigl [9]. The main differences in comparison to the baseline approach are highlighted in red.

followed by measuring real servomotor position $\theta_{\text{real}}(k)$.

The main difference between the original method proposal by Mrva et Faigl [9] and the latter extension by Faigl et Čížek [7] lies in the way how ground detection is handled using measured servomotor position $\theta_{\text{real}}(k)$. Original approach by Mrva et Faigl [9] computes difference between the commanded (desired) position $\theta_{\text{des}}(k)$ and the measured (real) servomotor position $\theta_{\text{real}}(k)$:

$$err(k) = \theta_{\text{des}}(k) - \theta_{\text{real}}(k) \tag{3}$$

Foot-strike is detected when $err(k)$ exceed fixed threshold $\tau = c_{\text{thld}}$:

$$err(k) > c_{\text{thld}} \tag{4}$$

as shown in Fig. 6.

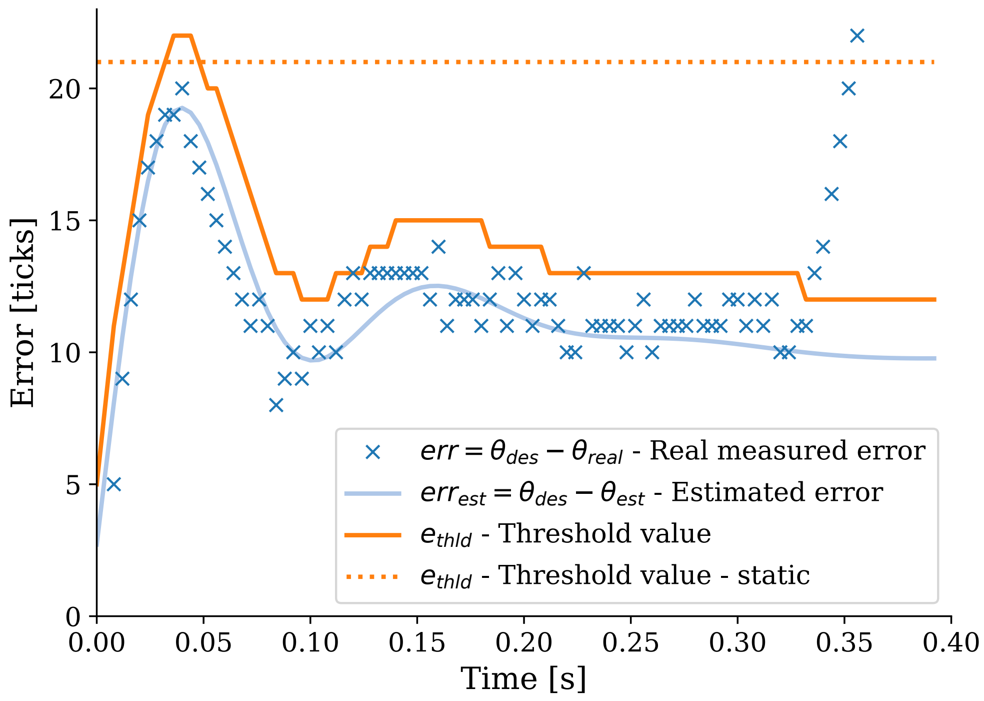


Figure 7: Measured and estimated position error of the femur actuator by the baseline approach [7]. The initial error is caused by significant static friction of the servomotor. Note that static threshold manually tuned value by original controller [9] is most of the time unnecessarily high.

This straightforward approach suffers from three main drawbacks, later removed by Faigl et Čížek [7]. Firstly, a fixed threshold c_{thld} used for deciding whether a leg already reached the stance phase has to be selected appropriately via manual tuning for Mrva's approach. Second, there is noticeable static friction and related dead-zone when the servomotor starts moving, which causes significant position error whenever servomotor starts moving (see Fig. 7). Consequently, a fixed threshold has to be high enough to prevent premature foot-strike detection. Lastly, setting a threshold this high influences robot performance; a higher threshold means higher torque upon leg impact, which affects robot attitude and increases energy dissipation causing servomotor overheating.

To overcome these drawbacks, Faigl et Čížek [7] incorporate the leg inverse dynamics model into adaptive locomotion control. The core idea is to use leg dynamics model to adaptively estimate expected leg error $e_{\text{thld}}(k)$ overtime. The expected leg error $e_{\text{thld}}(k)$ is computed as:

$$e_{\text{thld}}(k) = \theta_{\text{des}}(k) - \theta_{\text{est}}(k) + \epsilon \tag{5}$$

where $\theta_{\text{des}}(k)$ is desired servomotor position, $\theta_{\text{est}}(k)$ is servomotor position estimated using leg dynamics model and ϵ is experimentally found safety margin to compensate the joint discretization and

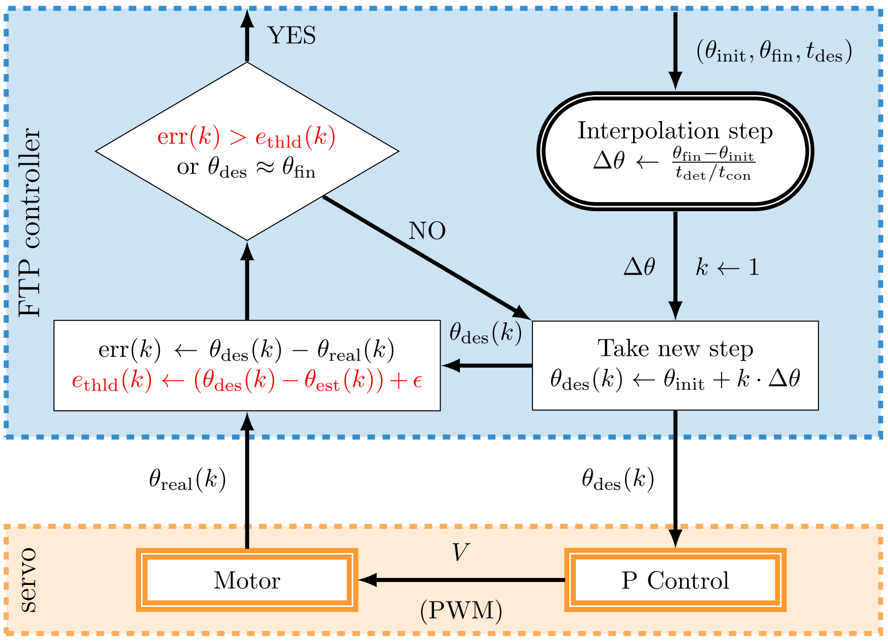

FTP controller
YES
$(\theta_{\text{init}}, \theta_{\text{fin}}, t_{\text{des}})$
$\text{err}(k) > e_{\text{thld}}(k)$ or $\theta_{\text{des}} \approx \theta_{\text{fin}}$
Interpolation step $\Delta\theta \leftarrow \frac{\theta_{\text{fin}}-\theta_{\text{init}}}{t_{\text{det}}/t_{\text{con}}}$
NO
$\Delta\theta$
$k \leftarrow 1$
$\theta_{\text{des}}(k)$
$\text{err}(k) \leftarrow \theta_{\text{des}}(k) - \theta_{\text{real}}(k)$
$e_{\text{thld}}(k) \leftarrow (\theta_{\text{des}}(k) - \theta_{\text{est}}(k)) + \epsilon$
Take new step $\theta_{\text{des}}(k) \leftarrow \theta_{\text{init}} + k \cdot \Delta\theta$
$\theta_{\text{real}}(k)$
$\theta_{\text{des}}(k)$
servo
Motor
V
(PWM)
P Control

Figure 8: The ground detection overview of the baseline TFP-based controller by Faigl et Čížek [7]. The main differences in comparison to the controller predecestor are highlighted in red.

mechanical inaccuracies of the particular leg. Leg is assumed to strike the ground when:

$$err(k) > e_{\text{thld}}(k) \tag{6}$$

as depicted in Fig. 8.

Both approaches stop the leg if a foot-strike is detected or if the final servomotor position θ_{fin} is reached. The value of θ_{fin} is set such that the leg foot-tip attempts to reach a position significantly lower than the current position and consequently attempts to penetrate the terrain. Commanding leg to try to penetrate the terrain is necessary to achieve ground reaction force, inducing the position's error that can be detected.

3.4 Dynamic Model

Leg inverse dynamics model is deployed by Faigl et Čížek [7] to estimate the collision-free motion of the leg while assuming the negligible coupling of the legs. Euler-Lagrange formulation [28] is utilized to analytically model leg inverse dynamics:

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau, \tag{7}$$

where $\mathbf{q} = \{\theta_1, \theta_2, \cdots, \theta_n\}$ is the vector of *generalized* n-dimensional *coordinates* coresponding to the leg joints angles, $\mathbf{D}(\mathbf{q})$ is the symmetric, positive definite *inertia matrix* of the chain of the rigid bodies, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is a tensor representing the *centrifugal* and *Coriolis* effects induced on the joints, $\mathbf{G}(\mathbf{q})$ is the vector of moments generated at the joints by the *gravitational acceleration*, and τ is the vector of actuation *torques* at the respective joints.

Utilized Dynamixel AX-12 servomotors provide neither torque neither electric current measurements that can be used to estimate torque directly. Therefore it is necessary to introduce an additional model for actuator using available positional data only. The dynamic model of the servomotor is given by:

$$J\ddot{q}^M + B\dot{q}^M + F(\dot{q}^M) + R\tau = K\,V, \tag{8}$$

where q^M is the rotor position angle before gear reduction, J is the *rotor inertia*, B is the *rotor damping*, F is the sum of the *static*, *dynamic*, and *viscous frictions* depending on the rotor speed, R is the gearbox *reduction*, τ is the servomotor *torque*, K is the *back electromotive force*, and V is the motor *voltage*. The proper values of J, B, F, R and K have been identified using a real servomotor and the values specified in the manufacturer datasheet.

Dynamixel AX-12 servomotor is internally controlled by P-type position controller which sets voltage value V based on the positional error:

$$V = k_p \cdot err, \tag{9}$$

where k_p is proportional gain of the controller, and err is difference between set position and current position of the actuator. The controller operates with $1\,\mathrm{kHz}$ frequency.

The complete model of leg inverse dynamics is obtained by substituting eq. 9 and eq. 8 into eq. 7. The inverse model precision is affected the most by leg links inertia matrices and friction estimated in the leg joints. Additionally, simplified models of robot legs such as point mass and rigid-rod were used to decrease the complexity of the calculation and measurement. These simplifications and non-stationarities that may occur during the deployment introduce additional errors to the leg inverse dynamics model. Moreover, numerous joint-related and link-related parameters have to be identified before using this analytical inverse dynamics model as described. Process of identification and parametrization of the baseline model is described in Section 5.2.

3.5 Problem Statement

As outlined in Section 3.3, the inverse dynamics model is used to estimate the position of the leg because the direct torque estimation would lead to double differentiation that leads to the inclusion of non-negligible noise in the estimation process. Therefore, the only viable method for a position-controlled system is to monitor the torques using the inverse dynamics as noted in [6]. However, with our actuators, direct information about the joint torque is not available and therefore, only the position estimate is used in the virtual elasticity monitoring according to [7]. Therefore, we formulate the problem as a time-series prediction of the joint position $q(t+m)$:

$$f\left(\mathbf{u}(t-n), \dots, \mathbf{u}(t), \mathbf{q}(t-n), \dots, \mathbf{q}(t)\right) = q(t+m) \tag{10}$$

where f is a function approximating inverse leg dynamics, i.e. function implemented by ML black-box, $\mathbf{u}(T)$ is the vector of reference positions being sent to the servo-motors at time T, $\mathbf{q}(T)$ is the vector of measured servo-motor positions at time T, m is *prediction horizon*, i.e. how many samples ahead the model predicts the position, and the metaparameter $n-1$ is the number of recent measured and reference positions used in predictions, further referred *history size*. In the most general case, the sizes of $\mathbf{q}$ and $\mathbf{u}$ are the same as the total number of servomotors. However, if the coupling between individual legs is negligible, it is possible to construct individual prediction models, one for each leg, similarly to the baseline approach. In that case, the dimensions of $\mathbf{q}$ and $\mathbf{u}$ are reduced to the number of joints per leg, i.e. three. There are multiple requirements on the learned model outlined in the next section, together with the evaluation metrics used for benchmarking the learned model performance.

3.6 Evaluation Metrics

In this section, we specify the wanted quantities and qualities of the model, namely its inputs and outputs and properties of a suitable model, non-stationaries to overcome and the metrics used to evaluate proposed model performance. As described in previous sections, the only available data are servomotor positions, both measured and desired. Hence inputs of the model can only consist of these two quantities. The only varying parameter regarding input is the number of recent joint positions used by the model. The output of the model is the vector of joint angles $\mathbf{q}$.

Regarding properties of the model, we want our model to be at least as accurate as adaptive gait controller [7] since groundwork already provide a prediction about foot-strikes. Achieving this alone will be beneficial since the learnable state estimator does not require laborious parameter identification as the current controller does. Additionally, robustness and good generalization properties are required because of hardly predictable terrain. These hostile conditions even further increase non-stationaries in the system; for example, the dust or fine mud increase servo friction, consequently heating up the servomotor resulting in non-stationary behaviour, mud deposits hand in hand with the possible link damage change weight of the robot legs resulting in non-stationary inertial properties of the system. Furthermore, the proposed model should take an adequate amount of time and data to train and required computational resources should be within the limitations of the SCARAB onboard computer. Online learning is advantageous but not a necessary property. We can achieve this either using an incrementally learnable algorithm or an algorithm with learning time negligible to the mission's duration. Lastly, a model should be capable of providing reliable data to enable rough terrain locomotion.

The Root Mean Squared Error (RMSE) metric has been selected to quantify the model accuracy throughout the experimental evaluation:

$$\mathrm{RMSE}(\mathbf{x},\mathbf{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x(i)-y(i))^2}, \tag{11}$$

where $\mathbf{x}$ and $\mathbf{y}$ are vectors (signals) of the same length n and the $x(i)$ and $y(i)$ are their respective elements at the i-th position. Hence, accuracy of single servomotor prediction is computed as:

$$\mathrm{RMSE}(\theta_{est},\theta_{real}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\theta_{est}(i)-\theta_{real}(i))^2} \tag{12}$$

where $\theta_{est}(i)$ is joint position estimated by the models for the i-th step, and $\theta_{real}(i)$ is the real joint position measured in step i. Errors in RMSE are squared before they are averaged, it consequently makes more significant errors more pronounced. This is important because large errors prematurely stop leg movement in the locomotion controller. For that reason we are using the RMSE metric to compare models in following Sections 5.4.2, 5.5, 5.6 and 5.7. To compare accuracy of multiple servomotors or the regressors, the cumulative RMSE is computed as the sum of the individal RMSEs:

$$\mathrm{cRMSE}(\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{m}\mathrm{RMSE}(\mathbf{x_i},\mathbf{y_i}), \tag{13}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are matrices of the same dimensions and $\mathbf{x_i}$ and $\mathbf{y_i}$ are respective columns of the matrices.

Chapter 4

Proposed Method

This thesis aims to create a learnable model to estimate the robot state defined in the previous chapter. The main motivation for developing a learnable system is to overcome inconveniences of the baseline analytical model [7]. Firstly, the baseline model requires cumbersome leg parameters identification as 18 sets of joint parameters and 18 sets of link parameters have to be identified. Secondly, the analytical model omits leg non-stationarities such as mechanical and electrical characteristics of servos that changes due to actuator heat up, gear-box and joint wear, links shape, mass, and inertial properties differences between legs caused by 3D printing imperfections and the environmental factors. A Robust robotic system should be able to overcome these conditions and parameter variations.

A straightforward solution to adapt the system to non-stationarities would use adaptive control [28,29,88], but these solutions are usually physically inconsistent. We reject white-box methods as they are mostly non-learnable. Approaches that identify model parameters use exhaustive evolution-inspired search methods such as Evolutionary algorithms [19, 22], herd-inspired algorithms [89] or swarm-inspired algorithms [90, 91] that are not trainable online. However, we see a potential benefit in splitting the robot dynamics model into individual dynamics models for each robot leg similarly to the modular controller design [16]. Grey-box models incorporate white-box models that can be identified automatically using evolution-inspired methods; the remaining black-box part can be, on the contrary, trained online. Nonetheless, we do not intend to limit learnability only to the part of the dynamics model; therefore, we aim to explore black-box approaches, where un-informed methods are deployed to handle the whole model by itself. This would also allow deploying the developed approach in the future to other legged robots with more advanced leg morphology.

Based on the literature survey in Chapter 2, we focus on lightweight machine learning methods that do not require extensive computational power so that it is possible to deploy both training and predictions on the onboard computer; therefore, we discard approaches like deep neural networks, spiking neural networks, echo state networks because these neural networks require extensive computational resources and a significant amount of data for training [17]. Ideally, we target online learnable approaches; hence, gaussian process regression is rejected since its limited online learning capabilities. On top of that, Gaussian processes are poorly scalable. On the other hand, locally weighted projection regression is online learnable, but we see no benefit in the dimensional reduction. We expect dynamics to depend on recent history; thus, we will not benefit from long-short-term memory neural networks which can learn long-time-dependent patterns. Consequently, we consider feed-forward neural network [51] and two types of least square methods [27], namely: (i) Ordinary Least Squares regression (OLS) further referred to as the *linear regressor*; (ii) Ordinary Least Squares regression with second-order polynomial features denoted the *polynomial regressor*, and (iii) three-layer feed-forward neural network with Rectified Linear Unit (ReLU) activation function further referred to as *ReLU regressor*.

All regressors have been implemented in the Python programming language. The linear and polynomial regressors utilize Scikit-learn library [92], and the ReLU regressor uses Chainer framework [93]. The size of the hidden layer for the ReLU regressor as well the number of most recent position sextets are meta-parameters that have to be experimentally found as further described in Section 5.4

Similarly to the baseline controller, we expect the leg dynamics to be decoupled, as examined and verified in Section 5.1; therefore, each leg's regressor models are constructed individually. Each regressor predicts the dynamics of the particular leg actuated by three servos rather than the dynamics of eighteen servomotors. The joint positions are predicted one step ahead since the predictions into a more distant future are losing accuracy, i.e. the future horizon $m = 1$. Therefore, the eq. (10) is reduced to:

$$f\left(\mathbf{u}(t-n), \ldots, \mathbf{u}(t), \mathbf{q}(t-n), \ldots, \mathbf{q}(t)\right) = q(t+1) \tag{14}$$

The input data utilized for training the model are composed of n most recent pairs of desired positions triplets $\mathbf{u}(T)$ and measured position triplets $\mathbf{q}(T)$, where the most suitable n is to be found experimentally.

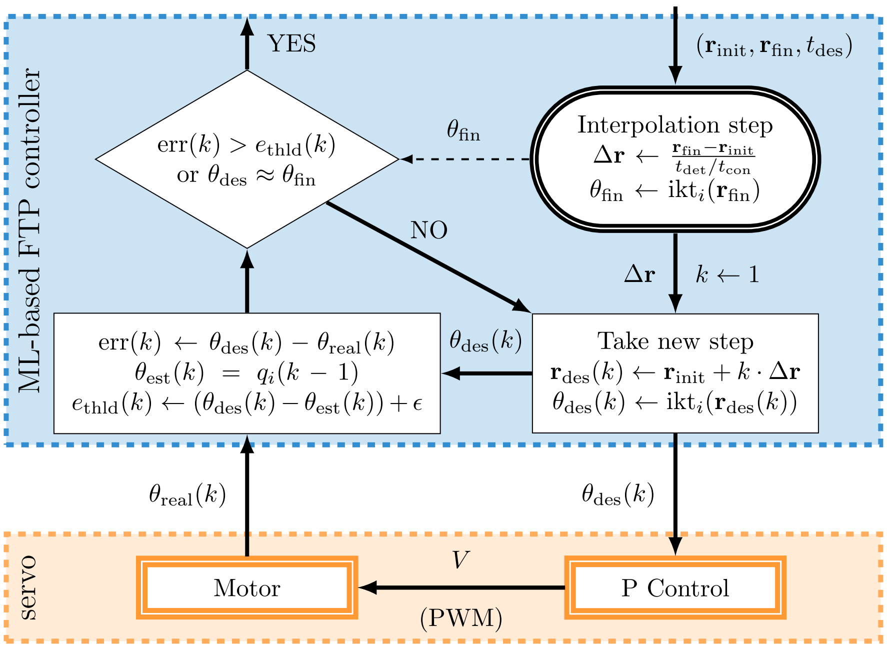

ML-based FTP controller
YES
$(\mathbf{r}_{\text{init}}, \mathbf{r}_{\text{fin}}, t_{\text{des}})$
$\text{err}(k) > e_{\text{thld}}(k)$ or $\theta_{\text{des}} \approx \theta_{\text{fin}}$
θ_{fin}
Interpolation step
$\Delta\mathbf{r} \leftarrow \frac{\mathbf{r}_{\text{fin}} - \mathbf{r}_{\text{init}}}{t_{\text{det}}/t_{\text{con}}}$
$\theta_{\text{fin}} \leftarrow \text{ikt}_i(\mathbf{r}_{\text{fin}})$
NO
$\Delta\mathbf{r}$
$k \leftarrow 1$
$\text{err}(k) \leftarrow \theta_{\text{des}}(k) - \theta_{\text{real}}(k)$
$\theta_{\text{est}}(k) = q_i(k-1)$
$e_{\text{thld}}(k) \leftarrow (\theta_{\text{des}}(k) - \theta_{\text{est}}(k)) + \epsilon$
$\theta_{\text{des}}(k)$
Take new step
$\mathbf{r}_{\text{des}}(k) \leftarrow \mathbf{r}_{\text{init}} + k \cdot \Delta\mathbf{r}$
$\theta_{\text{des}}(k) \leftarrow \text{ikt}_i(\mathbf{r}_{\text{des}}(k))$
$\theta_{\text{real}}(k)$
$\theta_{\text{des}}(k)$
servo
Motor
V
(PWM)
P Control

Figure 9: Overview of proposed machine learning-based method for ground detection.

The gait cycle of the proposed controller is similar to the baseline approach depicted in Fig. 5. The body levelling phase is presented to the stance phase, but the distinction of swing-up phase, swing-forward and swing-down phase is omitted since the proposed controller is no longer restricted to the rectangular trajectory, and the foot-tip can follow any arbitrary trajectory.

However, the contact detection module of the gait cycle depicted in Fig. 9 has been significantly changed. The foot-tip position is interpolated in the cartesian coordinates rather than the joint coordinates as in the baseline approach to obtain complete control over the trajectory shape, Interpolated coordinates are then converted to the joint coordinates using the inverse kinematics module ikt_i of the particular leg i. The same goes for the final position of the trajectory θ_{fin} used in the decision block. Uniform interpolation in the cartesian space does not ensure a uniform step in the joint space. Therefore, differences between two consecutive desired positions θ_{des} slightly vary, as discussed later in the thesis. Additionally, the θ_{est} is estimated using the previous step regressor prediction $q(k-1)$.

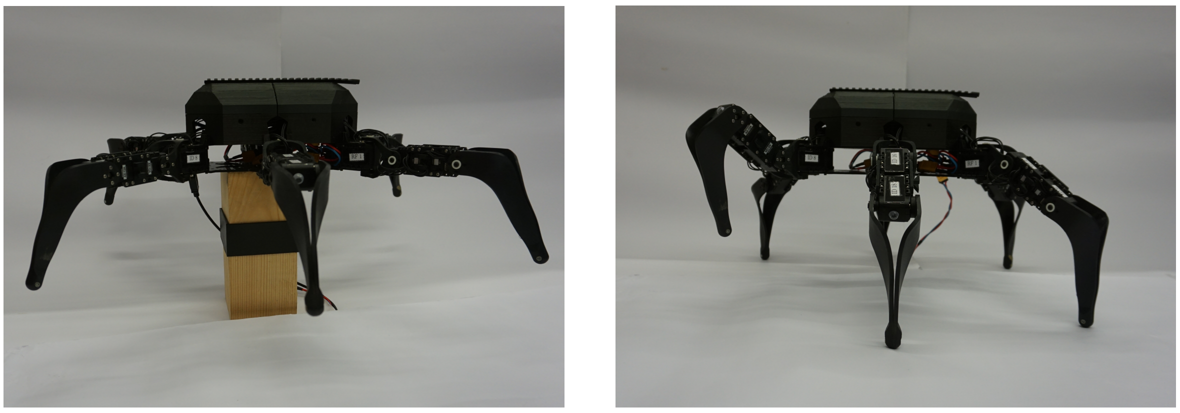
The following chapter experimentally validates inverse dynamics learning for small legged robots and compares the performance of the proposed regressor to the baseline analytical model [7].

Chapter 5
Results

This chapter compares proposed methods to the baseline model [7] in various aspects with the emphasis on the requirements outlined in Section 3.6 and describes the necessary background regarding these experimental comparisons. Leg dynamics decoupling is examined in the first part of this chapter since we need to verify our assumption that leg coupling is negligible to design models for a single leg rather than the whole robot. Section 5.2 lists parameters used by thw baseline model throughout the experimental evaluation. The behaviour and properties of the learned models have been first investigated on pre-recorded datasets and later deployed directly on the robot. Section 5.3 describes the process of datasets collection in different settings and datasets used by other experiments themself. Section 5.4 focuses on meta parameters search; the number of most recent measurements used for prediction is experimentally found, then the number of neurons in the hidden layer of the ReLU regressor is determined. Next four parts (5.5-5.8) compare baseline model to the proposed approaches given the properties listed in Section 3.6. Section 5.5 analyzes precision of the proposed models and compares it to the baseline model precision, section 5.6 examines generalization properties of the proposed models and baseline model when leg dynamics is modified, section 5.7 inspects amount of data needed to train proposed models, and section 5.8 investigates computational requirements of the proposed models. Lastly, proposed methods are deployed on the SCARAB platform to detect leg contacts in part 5.9, and finally, the selected proposed method is deployed to SCARAB platform to verify the approach in rough terrain traversal scenario.

All experiments conducted in this chapter have been performed using the laptop computer with dual-core Intel Core i5-3320M CPU @ 2.60 GHz, 16 GB RAM without GPU acceleration, running Ubuntu 18.04 Bionic Beaver operating system with ROS Melodic. However, some datasets have been collected using Intel NUC i7-10710U with 64 GB RAM mini PC, as noted in the particular cases. The performance of the used computers does not exceed the performance provided by the onboard computer of the SCARAB platform.

5.1 Leg Coupling



(a) SCARAB supported by wooden block ensuring the collision-free movement for any leg.

(b) SCARAB elevated on its legs to ensure collision-free movement for a selected leg.

Figure 10: Hexapod walking robot SCARAB in the pose used in experimental setups.

The coupling between legs occurs when the motion of one leg affects the motion of another leg through coupling over the body of the robot. The coupling depends on multiple factors, but it is mainly affected by the mutual inertia of the legs and body and the stiffness of the servomotors. If the coupling effect between individual legs is negligible, it is possible to construct multiple models for individual legs rather than a single complex model for the whole robot. In this experimental setup depicted in Fig. 10a, the robot is elevated on its legs in a significantly higher pose than the usual height used during locomotion. Standing on the legs reflects the real-life situation of traversing the terrain, and moving the centre of mass up decreases stability, which amplifies the possible coupling. Additionally, a highly elevated pose provides an obstacle-free area for legs movement. Two datasets were collected; the first, the base dataset, consists of a single reference leg periodically repeating prescribed gait pattern mid-air, and the second dataset captures two additional legs moving with the reference leg. In this experiment, we have decided to use the tripod gait pattern for leg control as it is the least stable gait. The desired joint angle θ_{des} and real measured joint angle θ_{real} were collected for all three leg servomotors of the reference leg at the highest possible sampling rate of $100\,\text{Hz}$ using the Intel NUC i7-10710U with $64\,\text{GB}$ RAM mini PC.

Leg position difference of particular joint is computed as:

$$\theta_{\text{diff}} = \theta_{\text{des}} - \theta_{\text{real}}, \tag{15}$$

where θ_{des} and θ_{real} are the desired joint angle and the measured real joint angle, respectively. The periodical pattern of the reference leg is selected because it allows averaging the error throughout multiple gait cycles. Additionally, the reference leg can repeat its trajectory straightforwardly in both datasets. The front left leg has been selected as the reference leg, and the rear left and middle right have been selected as additional legs for the second dataset.

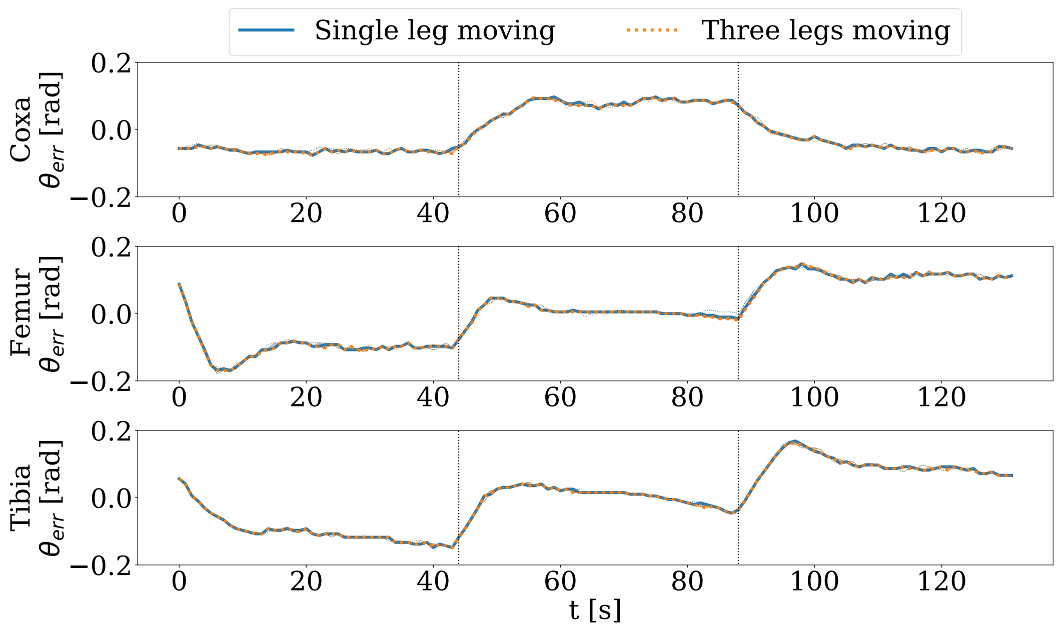


Figure 11: Comparison of the leg possition error between right front leg moving individually and the right front leg moving together with right rear and left middle legs. The thick lines depict the mean error value over 150 repetitions, the light-coloured is bordered by lower and upper quantile, and the thin dashed line represents minimal and maximal errors, respectively.

In total, 150 gait cycles have been collected for each dataset. The foot-tip followed equilateral triangular trajectory with circumradius $r_c = 5\,\text{cm}$. Triangular leg trajectory has been selected since the sudden foot-tip direction changes in the corners are expected to induce the most significant coupling effect among the commonly deployed leg trajectories due to the leg inertia. The collected data has been averaged over multiple gait cycles, and the mean position error θ_{diff} has been calculated according to equation (15). The results are visualized in Fig. 11 for the coxa, femur, and tibia joints, respectively.

Additionally, the cumulative RMSEs have been used to quantify the difference induced by the coupling effect. Firstly, the cumulative RMSE has been computed for individual gait cycles using averaged measured and set values. The averaged measured, and set values have been obtained in the same manner as the averaged gait pattern. i.e. the corresponding samples within different gait cycles have been individually averaged for both measured and set values. Secondly, the cumulative RMSE has been computed for the position error of the averaged gait errors for both datasets. The cumulative RMSE for a single leg moving $\mathrm{cRMSE}(\theta_{\mathrm{real}}^{\mathrm{single}}, \theta_{\mathrm{des}}^{\mathrm{single}}) = 0.0809$ and three legs moving at the same time $\mathrm{cRMSE}(\theta_{\mathrm{real}}^{\mathrm{multiple}}, \theta_{\mathrm{des}}^{\mathrm{multiple}}) = 0.0810$ are in the orders of magnitude greater then the cumulative RMSE of the signal difference $\mathrm{cRMSE}(\theta_{\mathrm{real}}^{\mathrm{single}} - \theta_{\mathrm{real}}^{\mathrm{multiple}}, \theta_{\mathrm{des}}^{\mathrm{single}} - \theta_{\mathrm{des}}^{\mathrm{multiple}}) = 0.0024$.

The provided results suggest no significant coupling between legs even in the conducted worst-case scenario. Therefore, the models can be considered for individual legs rather than a whole robot. Additionally, since legs share similar morphology apart from minor differences in the servomotor orientation and offset angles, the right front leg has been selected for the experimental benchmarking.

5.2 Baseline Model Parameters

The baseline analytical model [7] requires two sets of parameters to be identified; the first characterize the mechanical properties of the legs, the second describes the dynamical properties of the utilized Dynamixel AX-12A servomotors. As the precise identification of leg and actuator parameters is laborious, the original work uses only a single set of hand-tuned parameters for each leg. The values listed in Table 1 are used to calculate inertia matrix $\mathbf{D}$, centrifugal and Coriolis effects tensor $\mathbf{C}$ and vector of moments generated by gravity $\mathbf{G}$ in equation (7).

Table 1: Mechanical properties of SCARAB leg.

Product	**Variable**	**Value**	**Unit**	**Description**
Coxa	a_{c}	52	mm	Coxa link length
CoM 1	a_{cc}	25	mm	Coxa link center of mass position
Mass 1	m_{c}	20	gm	Coxa link mass
Femur	a_{f}	66	mm	Femur link length
CoM 2	a_{cf}	20	mm	Femur link center of mass position
Mass 2	m_{f}	115	gm	Femur link mass
Tibia	a_{t}	132	mm	Tibia link length
CoM 3	a_{ct}	50	mm	Tibia link center of mass position
Mass 3	m_{t}	62	gm	Tibia link mass

The second group of parameters used by the servomotor dynamics model (8) has been identified experimentally using a setup where the actuator is moving without any load. This setup can be achieved only prior to the assembly of the robot, and therefore it significantly complicates later identification of the servomotors parameters. The parameters are identified from the motion of the servomotor between two reference positions by increasing control voltage. The minimum voltage v_{min} when servomotor starts moving was identified as $v_{min} = 0.5\,\mathrm{V}$ and utilized to compute maximal static friction $F \simeq (k/R_a) \cdot v_{min}$, where $k = 3.07 \cdot 10^{-3}\mathrm{NmA}^{-1}$ is the back electromotive force constant, $R_a = 6.5\,\Omega$ is the motor resistance. The values of k, R_a and the gearbox ratio $R = 1/254$ have been found in the actuator datasheet[5]. The values of servomotor dynamics parameters are summarized in Table 2.

Table 2: Dynamic model parameters of the Dynamixel AX-12A.

Parameter	Value	Unit
J	$1.032 \cdot 10^{-7}$	$\mathrm{kg\,m^2}$
B	$3.121 \cdot 10^{-6}$	$\mathrm{N\,m\,s}$
F	$2.369 \cdot 10^{-4}$	$\mathrm{N\,m}$
R	$3.937 \cdot 10^{-3}$	-
K	$3.912 \cdot 10^{-3}$	$\mathrm{N\,m\,A^{-1}}$

5.3 Datasets Collection and Description

The datasets have been collected using SCARAB robot for later offline experimental examination of the proposed regressors. The front left leg of the hexapod has been selected to be used for data collection. The selection has no significant impact since all legs are similar with minor differences.

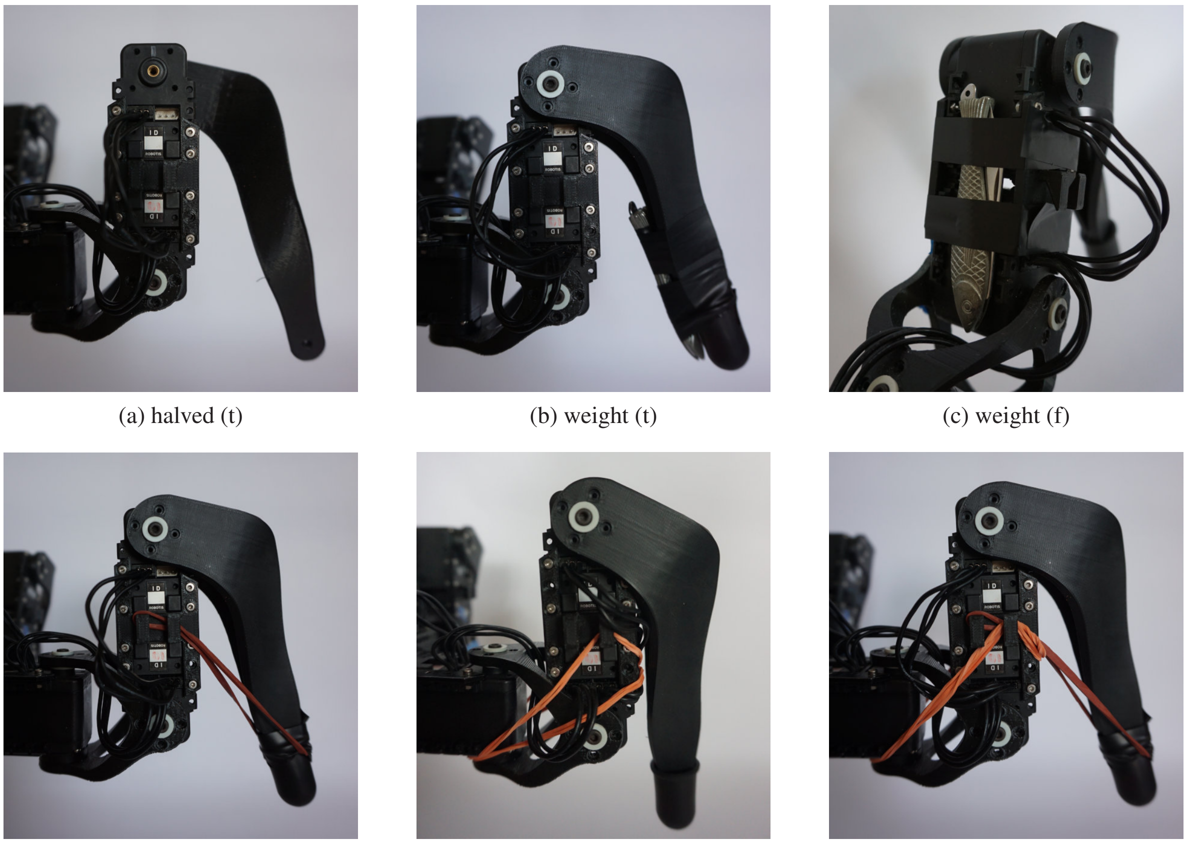
Datasets differ in both leg movement trajectories and non-stationaries. Different leg movement trajectories were selected to capture a wide range of possible leg movements without any particular pattern. Additional non-stationaries were introduced to reflect selected environmental effects on the robot as outlinde in Section 3.6. Namely, the weight of the last leg link was increased and decreased to simulate mud deposits and partial leg damage. Rubber bands were added to the joint servomotors increasing the load of the servomotor to simulate increased joint friction. Selected non-stacionarities are depicted in the Fig. 12.

In total, nine datasets summarized in Table 3 have been collected. During dataset collection, the robot was elevated with the robot body laying on the wooden support such that robot legs can move freely in mid-air without colliding with the environment, as depicted in Fig. 10b. The 2000 and 1000 random target points of the foot-tip were chosen within the operational space of the leg for datasets one and datasets 2-8, respectively. The leg trajectory was produced by interpolating between target points with the maximum allowed step size of $0.4\,\mathrm{mm}$. The produced trajectory of the foot-tip was then transformed to the joint coordinates using inverse kinematics. The final trajectory in joint space was executed using the open-loop controller that commands the leg with the desired joint angles. The desired joint angle θ_{des} and real measured joint angle θ_{real} were collected for all three leg servomotors similarly to the leg coupling experiment. The highest possible sampling rate ($100\,\mathrm{Hz}$) of the servomotor was used for all datasets.

5.4 Meta Parameters Search

Two meta-parameters have to be found before training the proposed regressors; a number of most recent measurements n used for prediction and size of the hidden layer for ReLU regressor l. Firstly, the number of recent measurements is examined with the ReLU hidden layer size of 100 neurons determined by the rule of thumb. After defining input size, the number of neurons in the hidden layer is investigated. Additionally, k-steps-ahead prediction can be considered to enable model-based predictive control, but one-step-ahead is used throughout this thesis.

[5]`https://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx`



(a) halved (t) (b) weight (t) (c) weight (f)

(d) rubber (t) (e) rubber (f) (f) rubber (f,t)

Figure 12: Leg modifications to introduce additional non-stationarities and alter leg parameters.

Table 3: List of collected datasets.

#	**Dataset name**	**Dataset size [n]**	**Induced modifications**
1	vanilla	55 333	No modifications
2	halved(t)	27 859	Weight of tibia link reduced by 12 g (see Fig. 12a)
3	weight(t)	27 033	Weight of tibia link increased by 31 g (see Fig. 12b)
4	weight(f)	27 603	Weight of femur link increased by 31 g (see Fig. 12c)
5	loosen(t)	26 994	Tibia link freely moving regardless of tibia servo position
6	rubber(t)	27 277	Tibia joint load increased with rubber band (see Fig. 12d)
7	rubber(f)	27 273	Femur joint load increased with rubber band (see Fig. 12e)
8	rubber(f,t)	27 493	Merge of rubber(t) and rubber(f) setups (see Fig. 12f)

5.4.1 History Size

Collected datasets consist of timestamped sextets of the desired joint angles θ_{des} and real measured joint angles θ_{real} collected from coxa, femur and tibia servomotors at 100 Hz. We refer to the number of sextets used as input to the regressors as *history size*. More information is fed to the models with increasing history size, but prediction and training time also increase. Data contained in the last three recent measurements (history size of three) should be sufficient to estimate the leg dynamics since baseline model [7] is a second-order system, but lengthy history is expected to increase prediction accuracy.

This experiment uses the vanilla dataset to produce a composed dataset with $n = N$ most recent consecutive measurements. The composed vanilla dateset is split in ratio 0.5:0.5 between the training and testing data. The training data are utilized for training regressors using the cross-validation with ratio of 0.9:0.1. The trained regressors are fed test data, and the predicted servomotor positions are compared to the measured one. This process is repeated for $N \in \{1, 2, \dots, 9\}$ as it is assumed that greater history size does not increase accuracy of the predictions.

Cumulative RMSEs for proposed regressors in relation to the history size are depicted in Fig. 13. Since the ReLU regressor is initialized at random, the training was repeated ten times in the case of the ReLU regressor, and the five-number summary is visualized using the box-plot. The most promising history size $n = 5$ was selected because the error of the linear and polynomial regressors is the lowest, and the ReLU regressor median cumulative RMSE value and the maximum cumulative RMSE are the lowest as well.

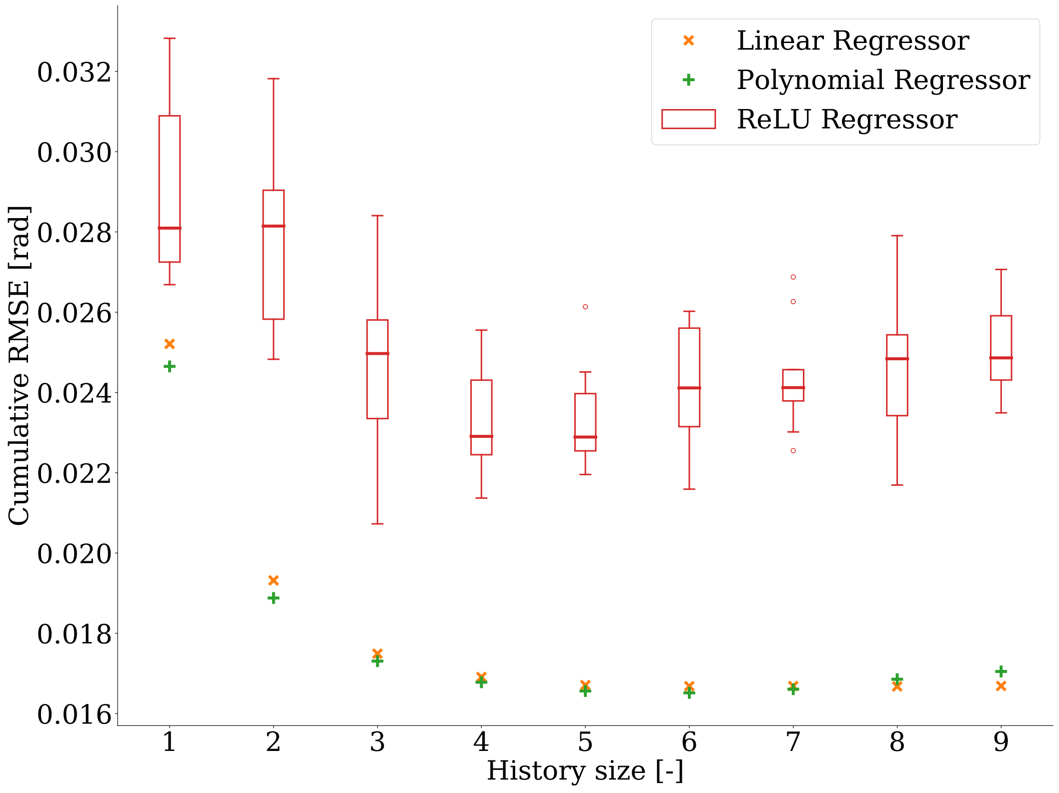


Figure 13: Relation between history size and the accuracy of the proposed regressors measured in the cumulative RMSE. The ReLU regressor is initialized at random; therefore, the training was repeated ten times for each history size, and the five-number summary was compiled. Linear and Polynomial regressors are trained using a deterministic training algorithm without requiring repetition of the training process.

5.4.2 Hidden Layer Size

Similarly to the previous experiment, the hidden layer's size for the ReLU regressor is tuned using a vanilla dataset. The hidden layer size of ReLU regressor l was iteratively set to experiment increasing sequence from 10 to 200 to cover reasonable hidden layer size values. The vanilla dataset is modified such that $n = 5$ most recent sextets of the desired joint angles θ_{des} and real measured joint angles θ_{real} collected from coxa, femur and tibia servomotors are sequenced. Like in the case of the history size experiment, the modified dataset is split in ratio 0.5:0.5 on data used for training and data used for validation and the training data are utilized for training ReLU with a cross-validation ratio of 0.9:0.1. This training process is repeated ten times for each hidden layer size l, and the results are summarized in Fig. 14 using cumulative RMSE metrics and a five-number summary. Based on this, the cumulative RMSE, hidden layer size of the ReLU regressor $l = 100$ was selected since the variance of the results is the lowest as well as the median cumulative RMSE.

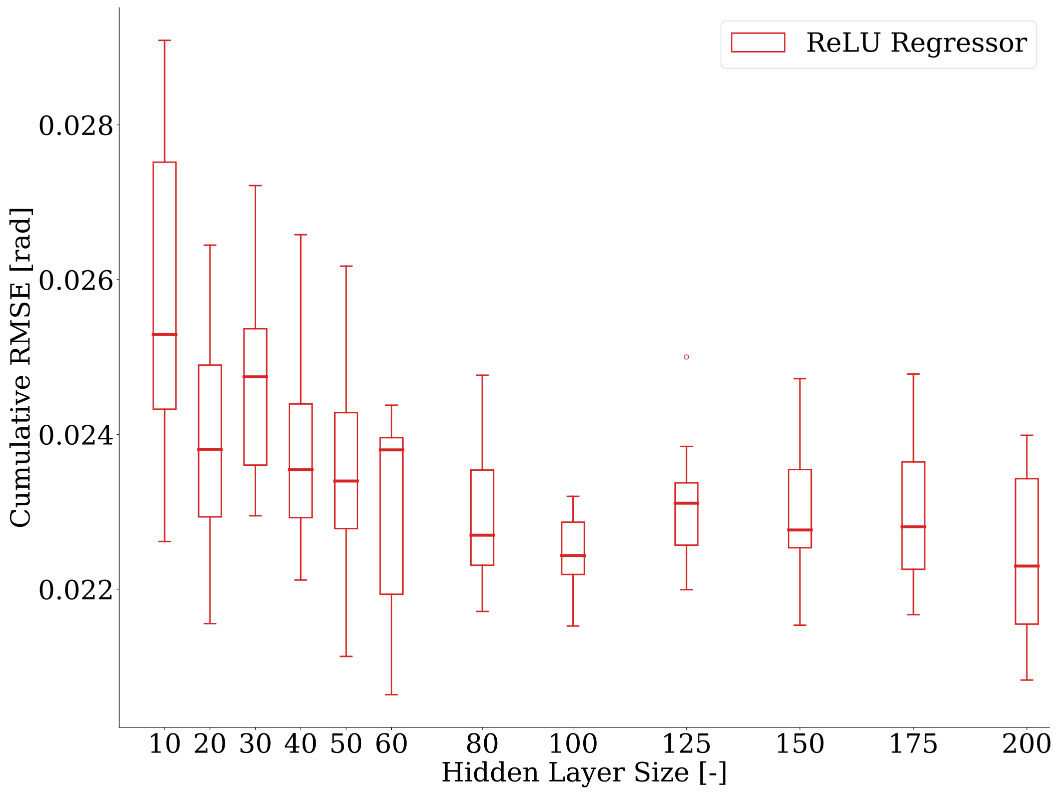


Figure 14: Five number summary of ReLU regressor cumulative RMSE for selected hidden layer sizes.

5.5 Model Precision

In this first benchmarking experiment, the prediction error of the proposed models is compared to the error of the analytical baseline model. Regressors are trained using the vanilla dataset modified to the history size $n = 5$; the dataset is divided in ratio 0.5:0.5 to data used for training and testing data, the cross-validation training ratio is 0.9:0.1. The cumulative RMSE is used to quantify the accuracy of the models as depicted in Table 4.

An example of the predicted position and the prediction RMSE error θ_{err} for each leg servomotor is depicted in Fig. 15. The results indicate that the proposed models estimate the leg position better than the baseline analytical model [7].

Table 4: Comparison of the cumulative root mean squared prediction errors.

Method	**Cumulative Prediction RMSE** [rad]
Baseline dynamic model [7]	0.0633
Linear regressor	0.0167
Polynomial regressor	0.0166
ReLU regressor	0.0172

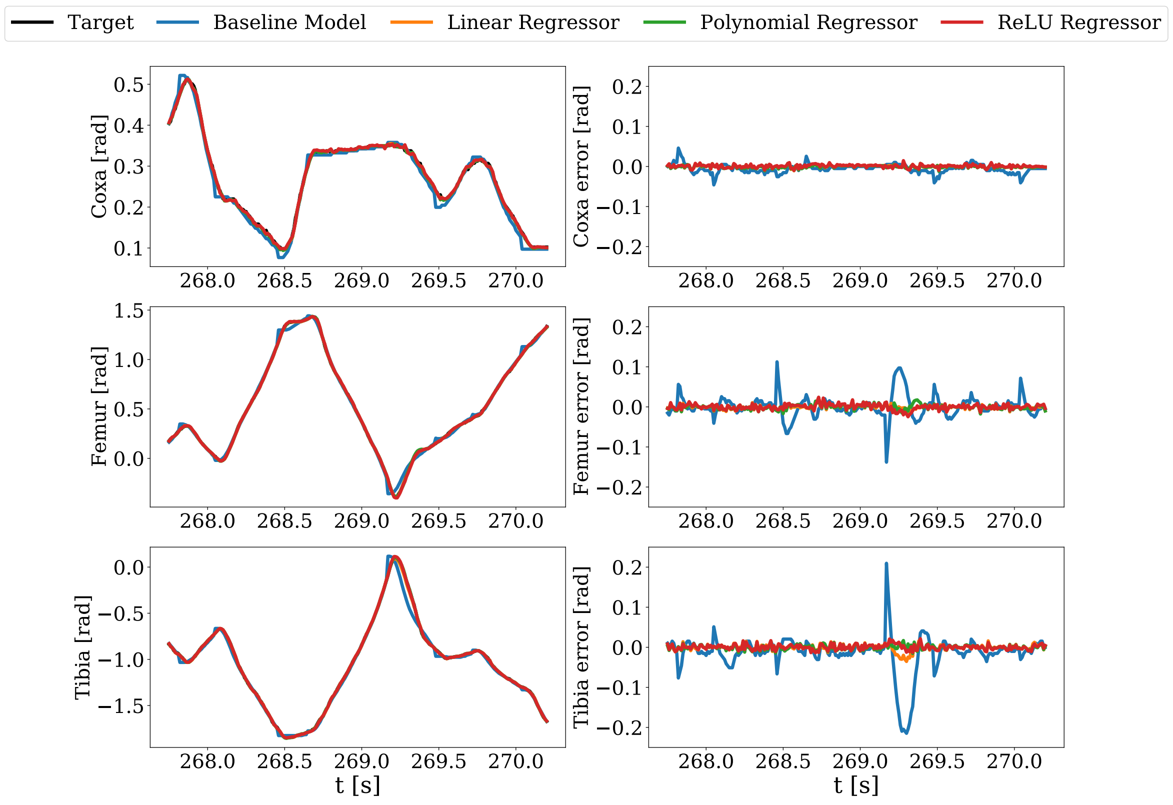


Figure 15: Example of predicted leg trajectory in joint coordinates (left column)and the corresponding prediction error calculated as $\theta_{\text{err}} = \theta_{\text{est}} - \theta_{\text{real}}$ (right column) for considered models. Leg follows a random location from the vanilla dataset.

5.6 Model Generalization

The second benchmarking experiment examined the generalization properties of the regressors that have been trained using the vanilla dataset in the same manner as described in the previous section. Trained models have been then validated on datasets 2-8 collected on modified leg mimicking parameter changes. The cumulative RMSE is utilized for each modification scenario to compare how regressors generalize leg dynamics and handle changes in its parameters. The results summarized in Fig. 16 suggest that the proposed regressors overall generalize leg dynamics better than the baseline analytical model [7] with the only exception of the *halved (t)* scenario, where polynomial regressor achieved significantly lower accuracy of its precision. The colour shade in the figure indicates how much each servomotor RMSE contributes to the cumulative RMSE; coxa servomotor has the darkest shade and tibia the lightest. Therefore, the considerable amount of the polynomial regressor cumulative RMSE in *halved (t)* originates from the tibia RMSE. In other scenarios, proposed methods cope with the parameter changes similarly.

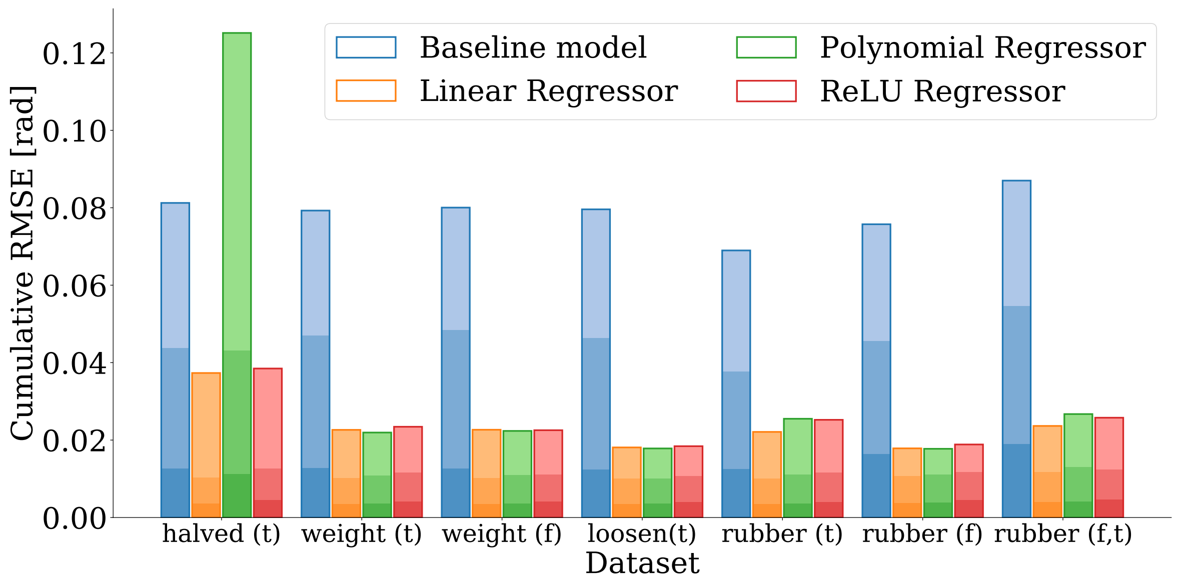


Figure 16: Comparison of model cumulative RMSE for scenarios where leg dynamics is modified. The darkest shade of a particular method is the coxa RMSE, the medium shade belong to the femur RMSE, and the lightest is the tibia RMSE. Cumulative RMSE is the sum of the joint RMSEs.

5.7 Size of Training Set

The third benchmarking experiment examines the relationship between the prediction quality and the amount of data used for training regressors. Aside from collecting new datasets during the robot deployment, it is possible to retrain the regressor in an online fashion. Mid-mission changes in the leg dynamics are potentially so significant that even generalizing models do not cope with them. Therefore, it is desirable to enable relearning from a relatively small batch of data collected ad hoc in the field. The following experiment has been conducted to investigate which regressors are capable of online learning.

Since servomotors are periodically read at the rate $\Delta t = 10\,\mathrm{ms}$, we can compute how long it takes to collect a particular sequence of a given length; therefore, we conveniently denote the size of the datasets in seconds to capture how long is robot immobilize to collect new data. The vanilla dataset has been used in this experiment to create a sequence of exponentially increasing time intervals of training data of the period from $0.1\,\mathrm{s}$ to $30\,\mathrm{s}$. For each such time interval of length m samples, random starting point p has been selected within the interval $[0, L-m]$, where L is the length of the half vanilla dataset dedicated to training. From chosen starting point p, m consecutive samples are selected, and input data of appropriate history size are produced to train regressors. Random selection of the particular length has been repeated ten times, and cumulative RMSE per trial has been computed using prediction error θ_{err}. The five-number summary for each time interval and each regressor is visualized in Fig. 17.

The results indicate that linear regressor overcome baseline model accuracy within a few seconds worth of data where it peaks its performance and polynomial regressor achieves similar performance a few seconds later and peak its performance within few tens of seconds. On the other hand, the ReLU regressor struggles to exceed the baseline model even with half a minute worth of data. These results indicate that linear and polynomial regressors are suitable for online learning, whereas ReLU is not suited for this type of deployment.

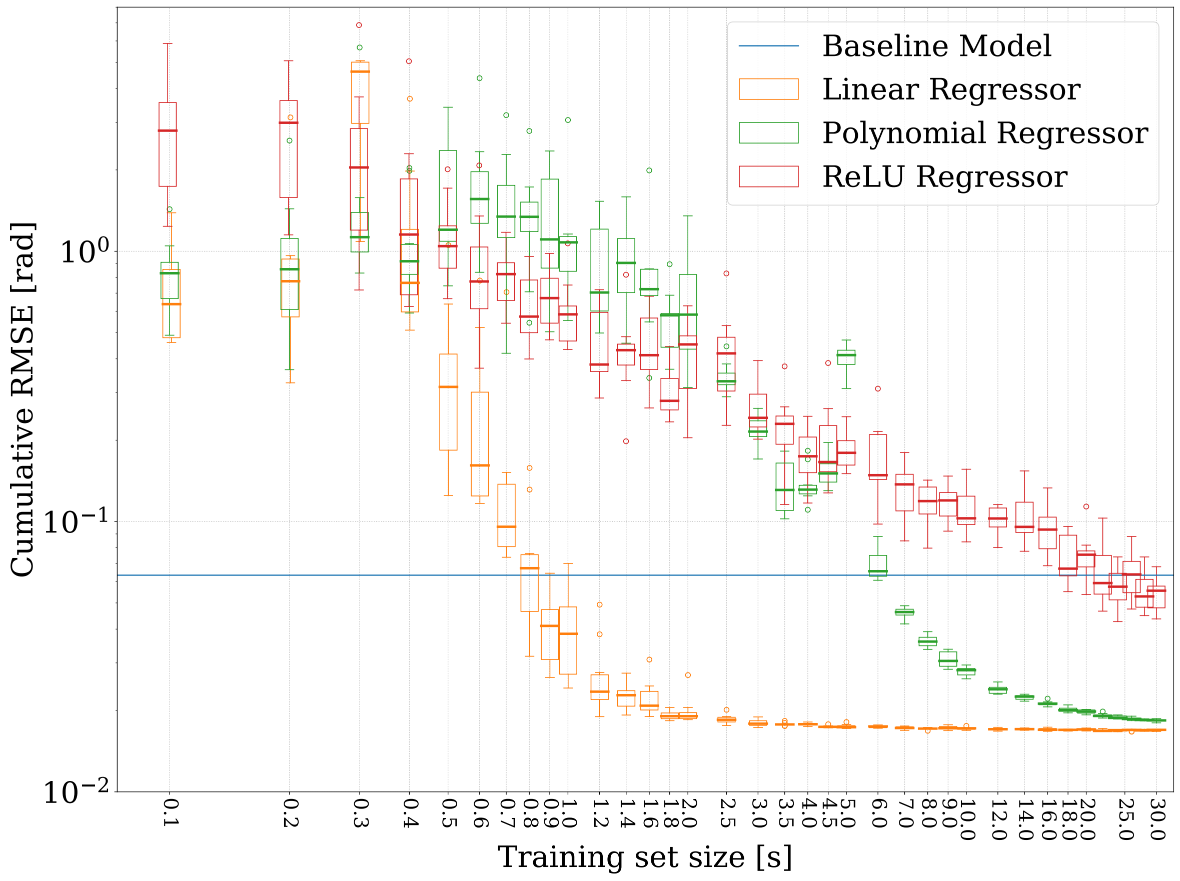


Figure 17: Cumulative root mean squared prediction error for training datasets of different size. The shown five-number summary is computed from ten independent trials. Note both axis are in the logarithmic scale.

5.8 Computational Requirements

The last benchmarking experiment focuses on the computational requirements of the proposed regressors. Computational requirements are essential when deployed on SCARAB onboard-computer; the computationally demanding method increases power consumption and consequently decrease operational time, the time required for relearning the changed-parameter leg dynamics increase the time spent training instead of continuing the mission, effectively slowing down average robot speed. Moreover, time spent producing predictions might increase the gait control period and thus decrease average robot speed. Therefore, the relation between the time spent training and the size of the training dataset and the time spent producing prediction have been examined.

Similarly to the previous experiment, the vanilla dataset has been used, exponentially increasing time intervals of training data starting at $0.1\,\mathrm{s}$ to $30\,\mathrm{s}$ are chosen at random, and each time interval is repeated ten times. The five-number summary of time required to train a particular regressor is computed for each time interval as depicted in Fig. 18.

The results show that the training time of the linear and polynomial regressors are negligible in comparison to the dataset sizes. Contrary, the time spent training is in order of magnitudes larger than the dataset sizes for the ReLU regressor. Based on the results of the last two experiments, the linear and polynomial regressors seems to be suitable for the online learning deployment scenario.

The mean time required to produce a prediction for learned regressors and baseline model [7] is summarized in Table 5. The total time spent predicting the servomotor positions was divided by

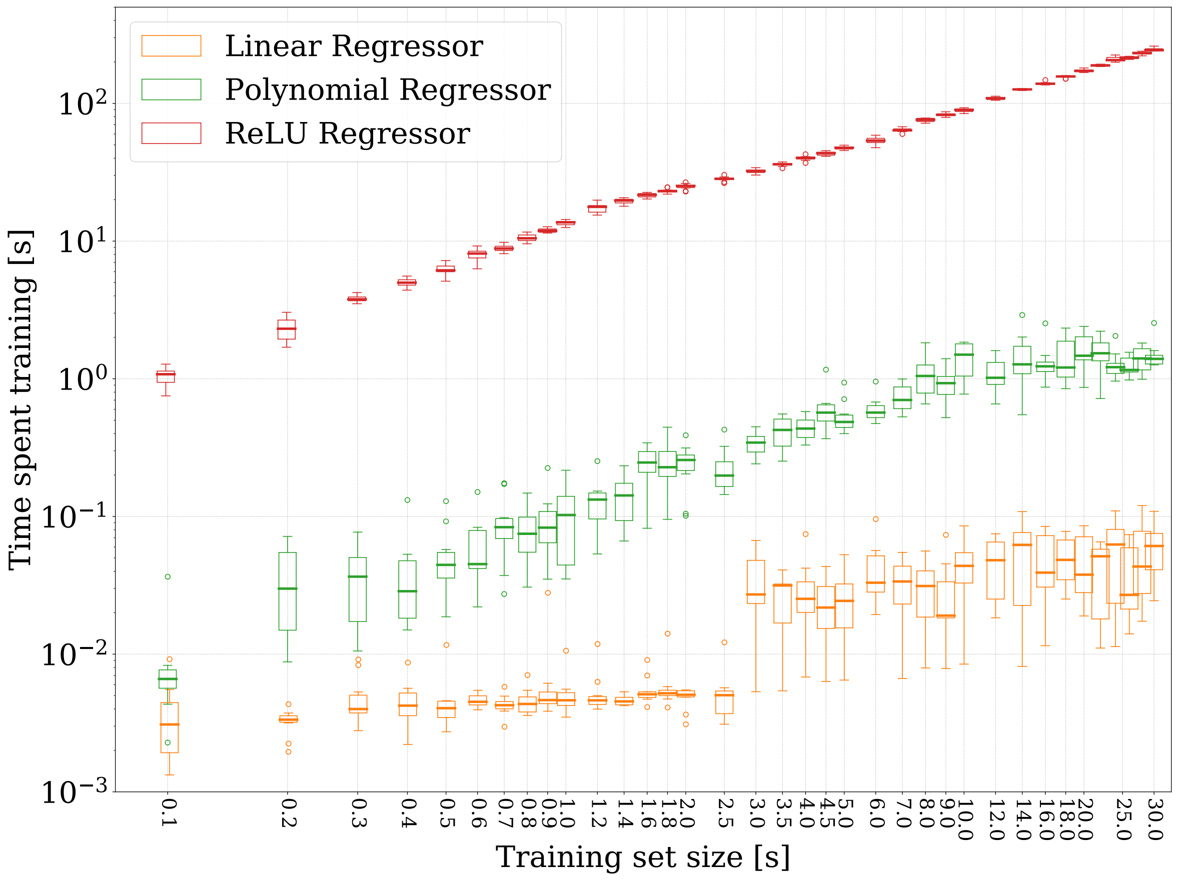


Figure 18: The computational time required to train proposed regressors using dataset of different sizes. The results show ten individual trial per size compiled into the five-number sumarry. Note both axes are in the logarithmic scale.

the total length of the dataset to obtain the mean prediction time. Provided results suggest that all the regressors are capable of producing prediction during the fraction of $10\,\mathrm{ms}$ control loop period; therefore, they can be deployed as part of the controller with ease.

Table 5: Mean computational time required for a single prediction.

Method	**Prediction time** [μs]
Baseline dynamic model [7]	6.7
Linear regressor	**0.3**
Polynomial regressor	18.1
ReLU regressor	8.3

5.9 Contact Detection

The contact detection represents the practical usage of the position prediction. Successful contact detection enables the legged robot to negotiate terrain in a real-life scenario. In this experiment, the robot body was elevated, and the leg moved freely, as in the case of collecting the evaluation datasets. The front left leg follows circular trajectory with the diameter $d = 10\,\mathrm{cm}$, regularly sampled at maximum rate of $\Delta t = 10\,\mathrm{ms}$ within $1\,\mathrm{s}$ resulting in the dataset of 100 points. This trajectory has been performed in six individual trials $\mathcal{T}_i$ summarized in Table 6. In the first trial, $\mathcal{T}_1$ leg follows trajectory freely without any obstruction. During the trials $\mathcal{T}_2$, $\mathcal{T}_3$ and $\mathcal{T}_4$, foot-tip collide with the obstacles of different heights at the different trajectory parts, and for trials $\mathcal{T}_5$ and $\mathcal{T}_6$ collision occurs on the femur link in the second half of the trajectory while the leg was ascending.

Table 6: List of the collision datasets.

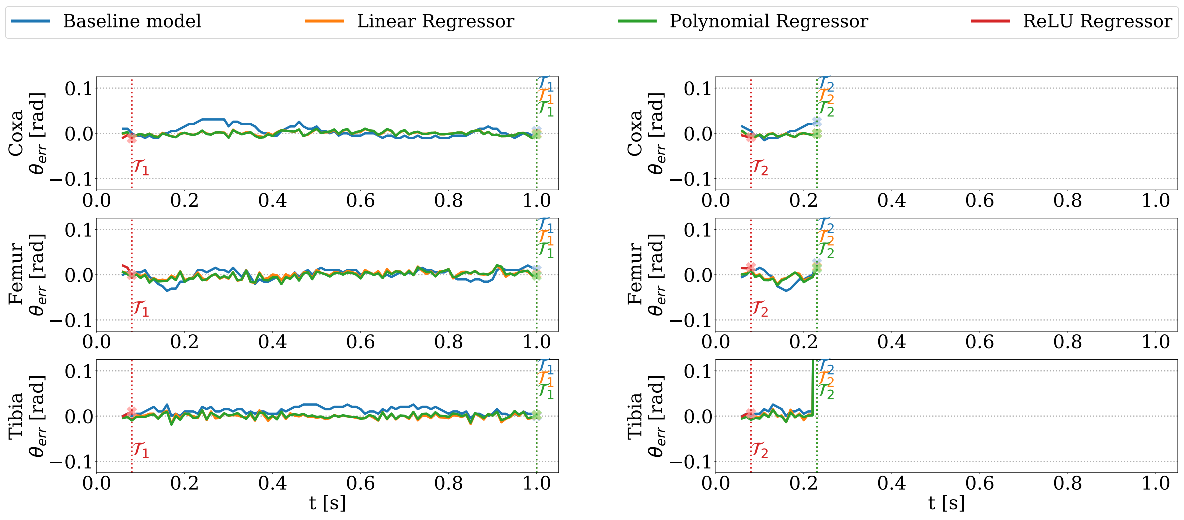
Trial	**Dataset size $[n]$**	**Description**
$\mathcal{T}_1$	100	Collision free path.
$\mathcal{T}_2$	100	Foot-tip collides $7.5\,\mathrm{cm}$ from the lowest trajectory point.
$\mathcal{T}_3$	100	Foot-tip collides $4.5\,\mathrm{cm}$ from the lowest trajectory point.
$\mathcal{T}_4$	100	Foot-tip collides $1.5\,\mathrm{cm}$ from the lowest trajectory point.
$\mathcal{T}_5$	100	Femur link collides in the middle of the ascenting.
$\mathcal{T}_6$	100	Femur link near the end of the leg movent.

The collected data have been used for offline detection of leg contact with the obstacle. The contact has been detected when sum of the errors from the leg joints exceed handtuned threshold value $e_{\mathrm{thld}} = 0.044\,\mathrm{rad}$. Since collision might occur anywhere on the robot leg, no particular servomotor is affected significantly more than others. Therefore, the sum of the error was used to decide whether a collision occurred rather than the error threshold for the individual servomotor. The prediction error is visualized in Fig. 19; the cross indicates detected collisions.

The presented results indicate that linear and polynomial regressors provide similar performance to the baseline dynamic model. In all trials, linear and polynomial regressors detect collision with the object at most a few samples later than the baseline model. ReLU regressor failed within few samples from the trajectory starts.

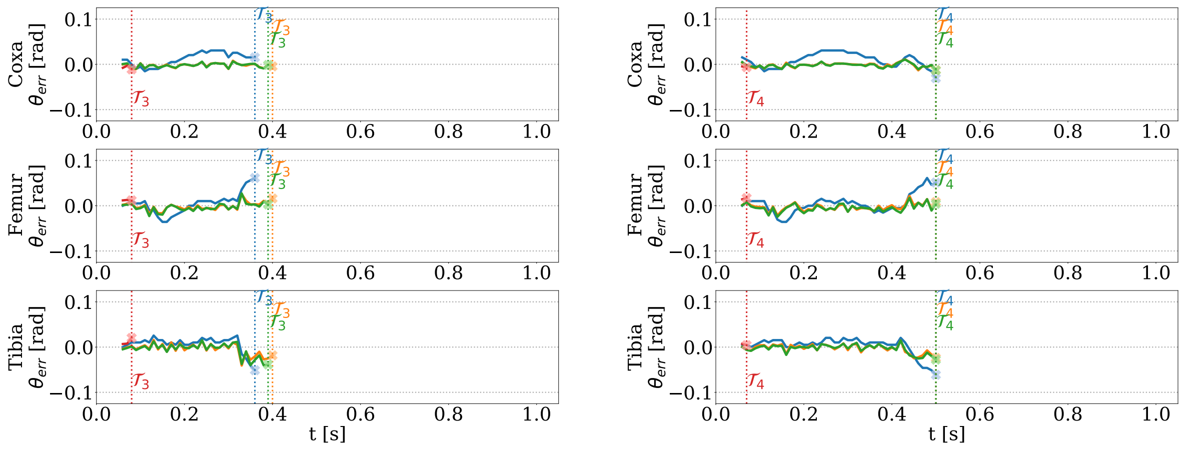
Further examination of the cumulative RMSE for the $\mathcal{T}_1$ reveals that the cumulative RMSE of the ReLU regressor ($\mathrm{cRMSE}_{\mathrm{ReLU}} = 0.026\,\mathrm{rad}$) is more then three times greater than the RMSE of the linear ($\mathrm{cRMSE}_{\mathrm{lin}} = 0.006\,\mathrm{rad}$) and polynomial ($\mathrm{cRMSE}_{\mathrm{poly}} = 0.008\,\mathrm{rad}$) regressors. Therefore, a prediction error histogram for the trial $\mathcal{T}_1$ was constructed to reveal prediction error distribution. As seen in Fig. 20, handtuned threshold $e_{\mathrm{thld}} = 0.044\,\mathrm{rad}$ is not large enough to cover ReLU prediction inaccuracies. A straightforward approach of increasing threshold to cover ReLU inaccuracies causes linear regressor and polynomial regressor to ignore collision. Therefore, separated significantly larger threshold $e'_{\mathrm{thld}} = 0.166\,\mathrm{rad}$ has been specified for the ReLU regressor and collision detection has been repeated as seen in Fig. 21. Using separated thresholds improves the overall performance of the detection system, even though some detections are missed.

This behaviour of the ReLU regressor is surprising since it performed similarly to the other proposed regressors and overcome the baseline model in all accuracy-focused experimental evaluations. We hypothesized that the RMSE used for accuracy-focused experiments is not a suitable metric. Even though the RMSE penalizes the error quadratically, it might not be sufficient since even a single malpredicted position can cause a leg to stop prematurely. The histogram of cumulative prediction errors has been constructed for the second half of the vanilla dataset used for training regressors as depicted in Fig. 22 where for each sample, the prediction error is obtained as the sum of the joint prediction errors. However, we rejected this explanation since the results in Fig. 22 coincide with the RMSE metrics summarized in Table 4 in previous section.



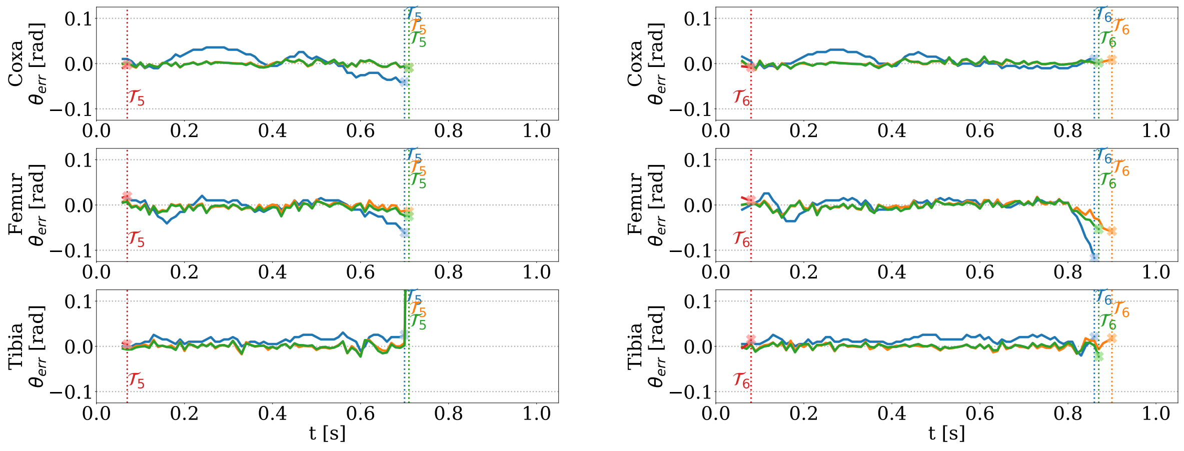
(a) Collision-free run $\mathcal{T}_1$.

(b) Run $\mathcal{T}_2$ with foot-tip collision.



(c) Run $\mathcal{T}_3$ with foot-tip collision.

(d) Run $\mathcal{T}_4$ with foot-tip collision.



(e) Run $\mathcal{T}_5$ with femur link collision.

(f) Run $\mathcal{T}_6$ with femur link collision.

Figure 19: Plots of collision prediction error $\theta_{\text{err}} = \theta_{\text{real}} - \theta_{\text{est}}$ and single threshold $e_{\text{thld}} = 0.044\,\text{rad}$. The first trial $\mathcal{T}_1$ is collision-free. An obstacle has been placed at a different part of trajectory in the remaining five runs $\mathcal{T}_2, \ldots, \mathcal{T}_6$. The annotated vertical lines represent detected collisions of the corresponding regressor and trial with the respective color-coding.

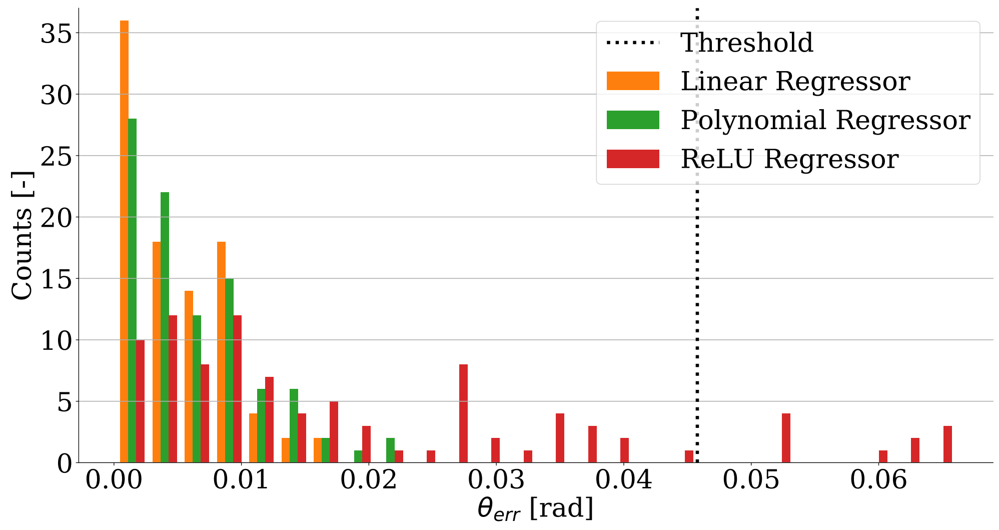
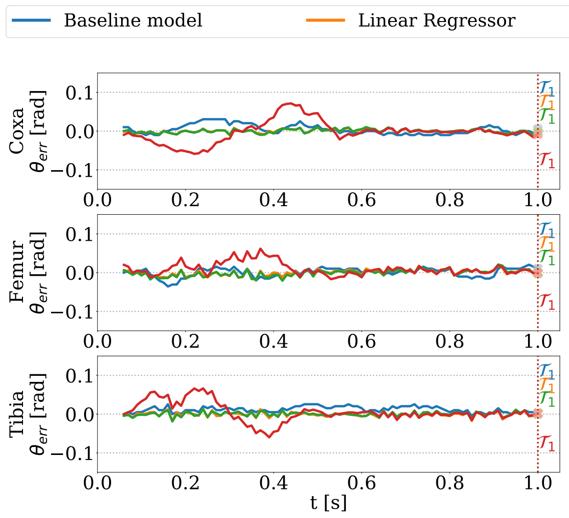
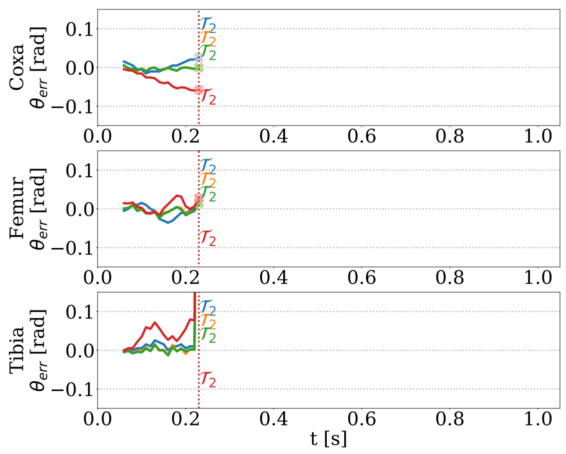


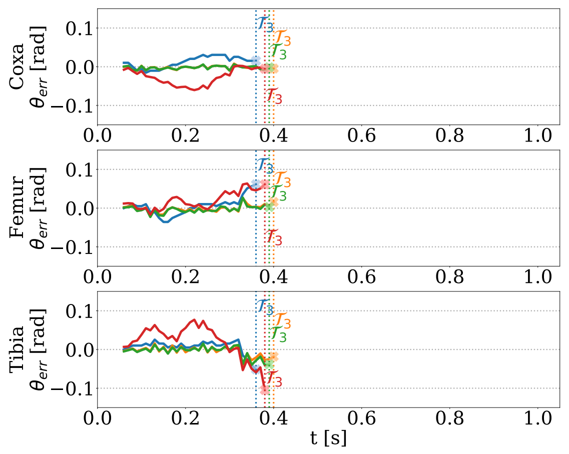
Figure 20: Histogram of the prediction errors across the $\mathcal{T}_1$ run for each consider regressor.



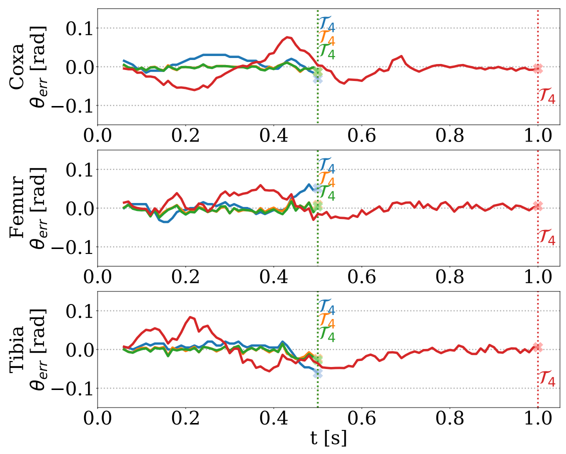
(a) Collision-free run $\mathcal{T}_1$.



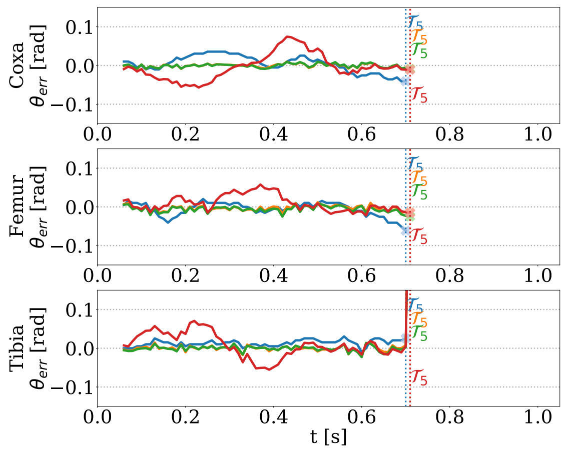
(b) Run $\mathcal{T}_2$ with foot-tip collision.



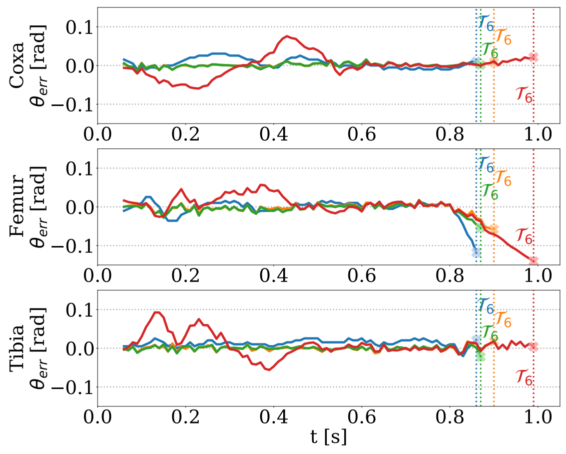
(c) Run $\mathcal{T}_3$ with foot-tip collision.



(d) Run $\mathcal{T}_4$ with foot-tip collision.



(e) Run $\mathcal{T}_5$ with femur link collision.



(f) Run $\mathcal{T}_6$ with femur link collision.

Figure 21: Plots of collision prediction error $\theta_{\text{err}} = \theta_{\text{real}} - \theta_{\text{est}}$, threshold $e_{\text{thld}} = 0.044\,\text{rad}$ has been used for linear and polynomial regressor and threshold $e'_{\text{thld}} = 0.166\,\text{rad}$ has been used for ReLU regressor. The first trial $\mathcal{T}_1$ is collision-free. An obstacle has been placed at a different part of trajectory in the remaining five runs $\mathcal{T}_2, \ldots, \mathcal{T}_6$. The annotated vertical lines represent detected collisions of the corresponding regressor and trial with the respective color-coding.

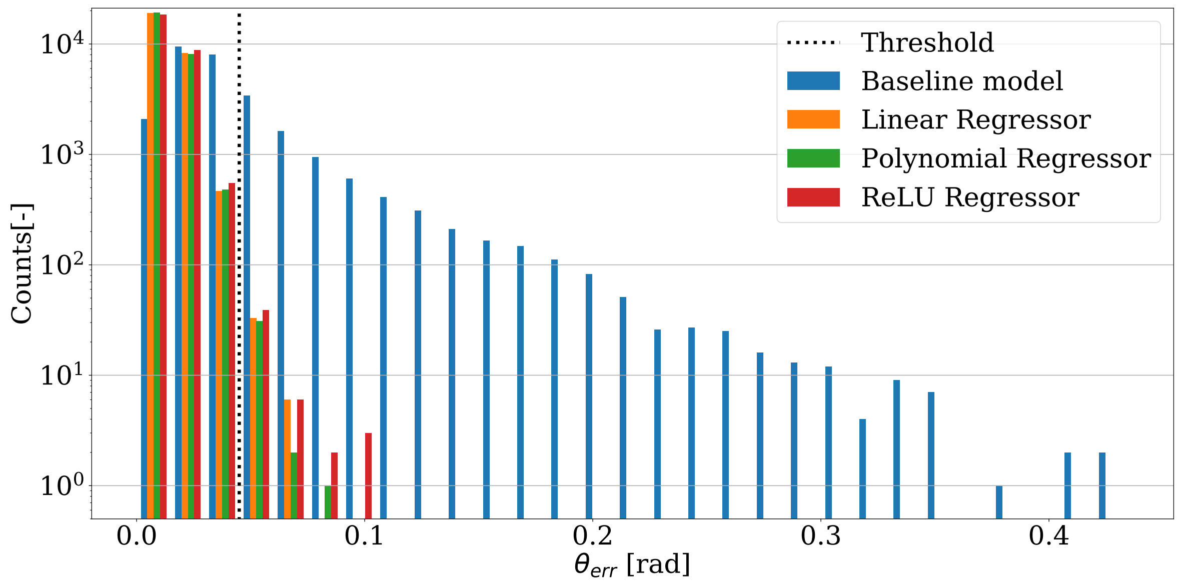


Figure 22: The histogram of the cumulative prediction error of proposed regressors for the vanilla datasets. All three regresors achieve similar performance. Note the logarithmic y-axis.

5.10 Deployment

Based on the previous experimental results, the most suitable regressor is about to be chosen for deployment in this section. The model generalization properties examined in the Section 5.6 are undecisive aside from an unusual behaviour of the polynomial regressor for the halved dataset. The size of the training set examined in Section 5.7 as well as computational requirement studied in Section 5.8 are in favour of linear and polynomial regressors because of the ReLU regressor computational and training data demands. The results from the previous Section 5.9 support these two regressors as well. Therefore, the linear regressor was selected to be deployed for contact detection.

Additional datasets have been collected for every leg. Similarly to the experimental examination of the leg coupling, the robot was elevated on the legs as depicted in Fig. 10b, and the current active leg foot-tip followed a randomized collision-free trajectory. The particular trajectory was created from 1000 randomly selected points within the operational space of the foot-tip, and every two points were interpolated in the cartesian space with the interpolation step $\Delta r = 0.4\,\mathrm{mm}$. In total, six datasets have been collected, one for each leg, in the same way as for the coupling experiment, i.e. the desired joint angle θ_{des} and real measured joint angle θ_{real} were collected for all three leg servomotors of the current reference leg at the highest possible sampling rate of $100\,\mathrm{Hz}$ using the Intel NUC i7-10710U with $64\,\mathrm{GB}$ RAM mini PC. Then linear regressors were trained using the collected data.

The triangular trajectory with the circumradius $r_c = 5\,\mathrm{cm}$ was selected for the locomotion controller. However, contradicting the results gathered up to this point, the robot could not detect collisions reliably. At least one leg stopped prematurely once in a few gait cycles, and more importantly, up to a half of the collisions was not registered by the locomotion controller across all leg. Missed detections were distributed evenly across all legs. Therefore, we hypothesized that the problem is rather systematic than caused by an inappropriately selected threshold.

Since regressors performed comparably well in the previous evaluation scenario, we expected a similar performance to occur in case of any considered regressor. Therefore, we decided to examine the prediction process in detail as described in the following section instead of deploying the “backup regressor.”

5.11 Deployment Analysis

While searching for an explanation of why the deployment failed, we focused on the differences between the circular trajectory used in the contact detection scenario and the triangular trajectory in the deployment scenario. The initial explanation blaming the shape of the trajectory was rejected since the training data consisting of the straight trajectory segments were more similar to the triangular trajectory than the circular shape. Therefore, opposite results are expected to be observed if this hypothesis is true.

Secondly, the statistical quantities of the training vanilla dataset and both circular and triangular trajectories were examined. Namely, we focused on the size of the interpolations steps in the joint space. As described in Section 4, the trajectory is interpolated in the world coordinates with the fixed step size of $\Delta r = 0.4\,\mathrm{mm}$ and then transformed to the joint space using inverse kinematics. Therefore, the difference between two consecutive joint positions is not fixed but varies even for the same interpolation step. The relative histogram of the joint difference (velocity profile) in the vanilla dataset and both considered trajectories is depicted in Fig. 23.

The relative histogram shows that the vanilla dataset and circular trajectory joint differences (velocities) are distributed similarly. However, the triangular trajectory contains specific velocity patterns that are best visible for the coxa and femur joints. These velocities make only a tiny proportion of the training data. We hypothesized that the difference between the distribution of the joint velocity within the training and testing data significantly affects the contact detection accuracy. Additionally, we expect the regressors to predict the leg dynamics accurately for the trajectories with similar ve-

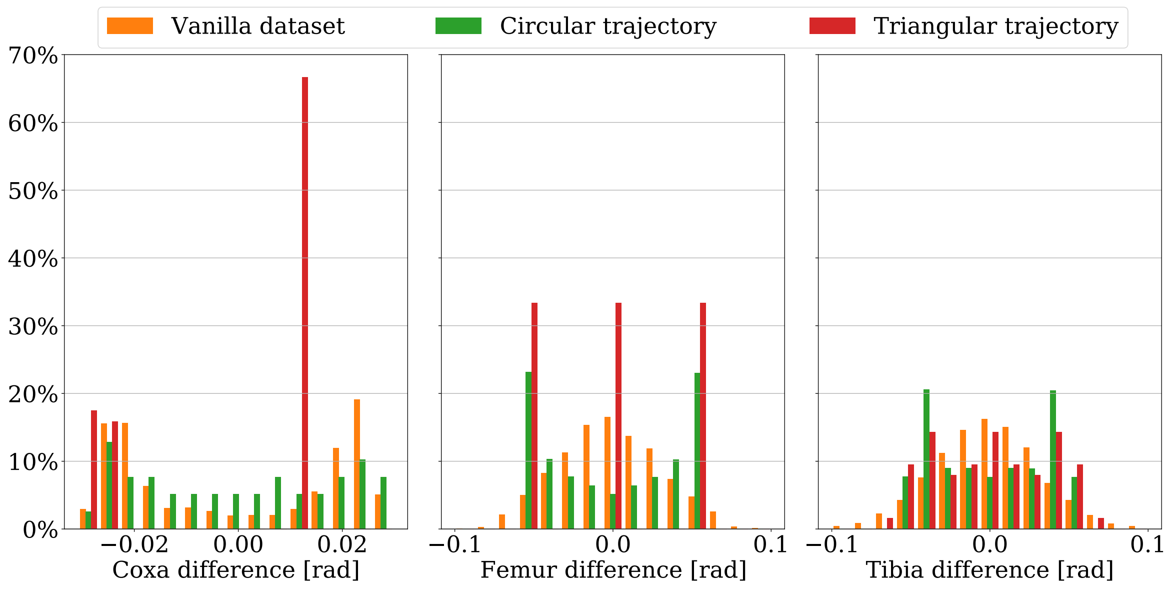


Figure 23: Relative histogram of the velocities (differences between two consecutive joint positions) in the randomized trajectory in the vanilla dataset, circular trajectory used for the contact detection and the triangular trajectory used in the deployment. The triangular trajectory has specific velocity spikes mostly visible for the coxa and femur joint.

locity profiles. On the other hand, trajectories with significantly different velocity profiles and those containing the joint velocities not covered by the training data are expected to be predicted with low accuracy since the comparison in Fig. 23, and deployment failure signifies poor generalization of the trajectories with the different velocity profile.

5.11.1 Effect of Interpolation Step on Prediction Accuracy

Additional datasets have been collected to examine the relationship between the joint velocity profiles within the training and testing data. Each newly collected dataset consists of 1000 randomly selected points within the operational space of the front left leg. The trajectory was constructed by interpolation between the pairs of consecutive points. A different number of samples per cm was selected during the interpolation to affect the velocity distribution for each dataset.

In total eleven datasets have been collected labelled $1, 2, \ldots, 10$ ppcm and $1 - 10$ ppcm. Dataset n ppcm uses n samples per centimetre during the trajectory interpolation, and $1-10$ ppcm uses a random number of steps per centimetre for each pair of interpolated points. During the data acquisition, the robot body was elevated such that the left front leg can move freely, and the desired joint angle θ_{des} and real measured joint angle θ_{real} were collected for all three leg servomotors at the highest possible sampling rate of 100 Hz using the Intel NUC i7-10710U with 64 GB RAM mini PC.

The distribution of the joint differences is depicted in relative histogram Fig. 24. The range of velocities contained in the dataset increases with decreasing number of interpolation points per centimetre (and increasing the maximal joint velocity). Therefore, a wider range of the joint velocities is contained.

In the next step, the new batch of regressors is trained. A regressor instance of each considered type (linear, polynomial and ReLu) is created for each new dataset, and the same dataset is used for training. Datasets other than the training one are used to produce a prediction, and the cumulative RMSE of prediction is computed.

The result summarized in Fig. 25 indicates 1) the linear regressor and the ReLU regressors achieve

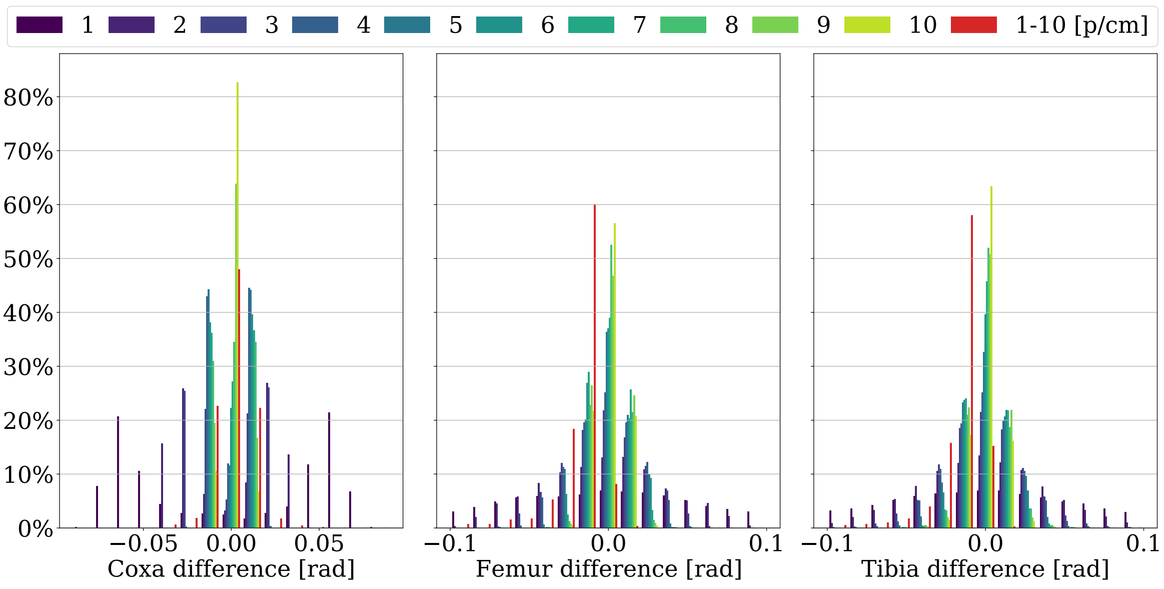


Figure 24: Relative histogram of the velocities (differences between two consecutive joint positions) in the randomized trajectory interpolated with the varying number of interpolation steps per centimetre. With decreasing number of samples per centimetre, the range of velocities contained in the datasets increases.

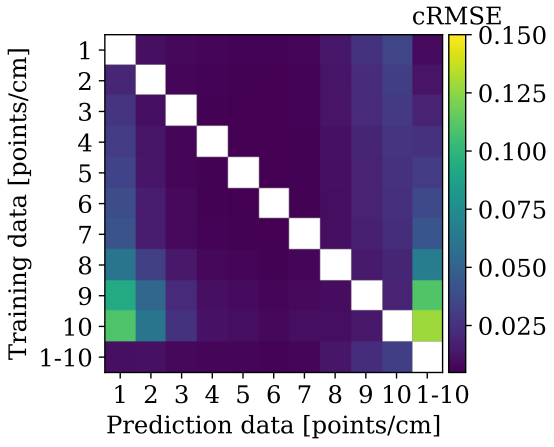
comparable performance, their behaviour and generalization properties across different velocity profiles are comparable, 2) the polynomial regressor extrapolates poorly outside of the provided velocity profile. The wild performance of the polynomial regressor outside the training data is expected since the value of second-degree features grows significantly; therefore, error reaches significantly greater values than the error of other regressors. Due to high values of cumulative RMSE, the base-ten logarithm of cumulative RMSE is also provided.

Regressors trained using dataset $1 - 10$ ppcm performed better than those trained using any other datasets $1, 2, \ldots, 10$ ppcm. In cases of the polynomial and the linear regressor, the regressor instances trained using a dataset with few samples per cm performed similarly to those trained using enhanced $1 - 10$ ppcm dataset, the performance of the ReLU regressor trained using $1 - 10$ ppcm was considerably better than the one trained using 1 ppcm. Hence, the linear regressor trained using the $1 - 10$ ppcm dataset is selected to be deployed in the locomotion controller for the second deployment attempt. However, retrained regressors failed to detect foot contacts at a similar rate to those trained using the vanilla dataset. Therefore, we decided to examine the regressors even further by examining the decision process in detail.

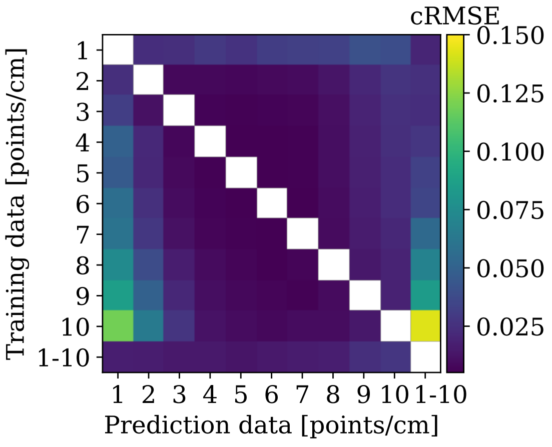
5.11.2 Regressor Structure Analysis

To analyze the decision process of the regressors, we focused on regressor structure to examine how the predictions are produced. Since all proposed regressors combined inputs to produce output, we investigated how input features affect the outputs. This information is encoded inside the regressor structure, in the case of linear and polynomial regressors in the coefficient matrix and the interception vector and the case of ReLU regressors in the weights of the neurons.

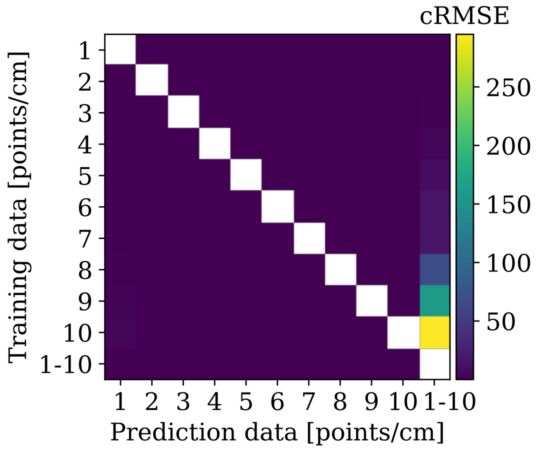
The examination of the linear and polynomial regressors is straightforward since both are represented by a single coefficient matrix and a single vector of interceptions varying in the dimensions only. For linear regressors, the absolute values of coefficient matrix and interception vector are concatenated and visualized in Fig. 26, where the last column represents the static bias, and the other



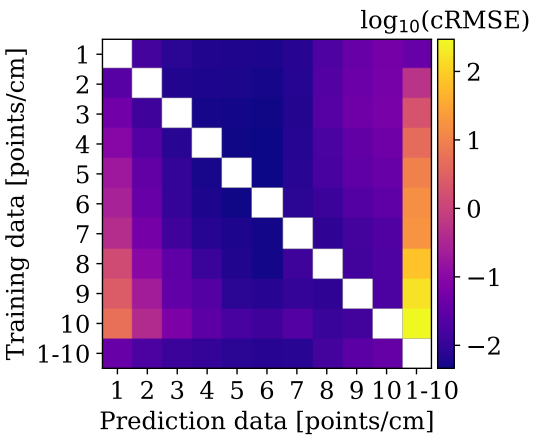
(a) Linear regressor.



(b) ReLU regressor.



(c) Polynomial regressor.



(d) Polynomial regressor in logarithmic scale.

Figure 25: Comparison of the prediction accuracy relation between the interpolation step size in the training and prediction data. The results suggest that regressors tend to overfit the particular interpolation step. Consequently, using the range of the interpolation rather than a single one increases the regressor accuracy. The accuracy of the linear and ReLU regressors is comparable across all cases; nevertheless, the polynomial regressor accuracy decreases significantly for unknown velocity profiles.

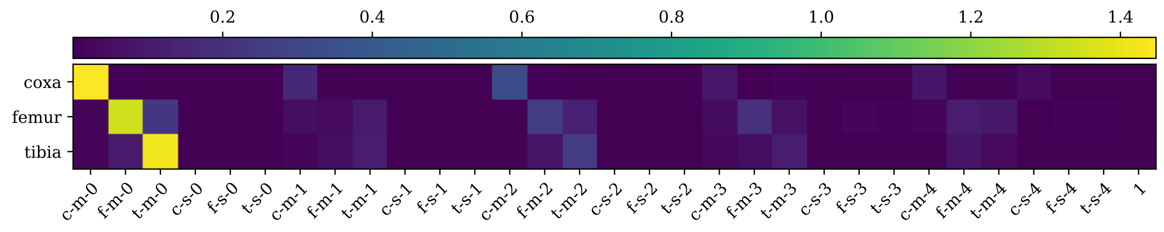


Figure 26: The absolute weights of the linear regressor coefficient matrix and the interception vector are concatenated to the single matrix, where the columns represent the features, the rows are leg joint position predictions, and the colour of a particular cell indicates how much the feature contributes to the prediction of the joint position. The features are labelled such that the first character is the first letter for the servomotor; the second character signifies whether the value is set to a servomotor (s) or measured in the servomotor (m), the last digit is how many time steps have passed since the position was set or measured. The interception vector is labelled 1. Based on this figure, we assume that the currently measured servo position plays a crucial role in predicting the position supported by the past measured positions and the set positions are omitted entirely.

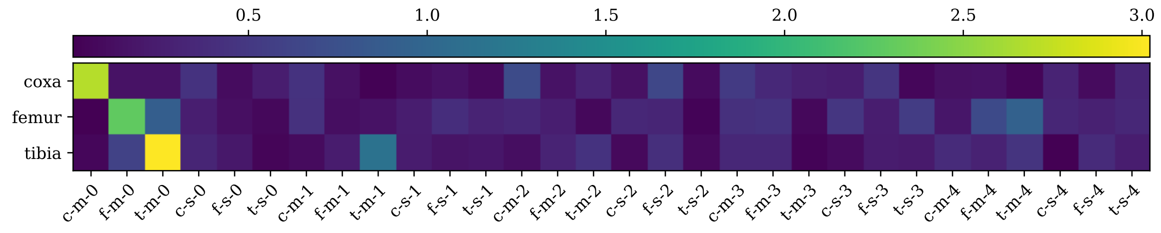


Figure 27: The ReLU regressor absolute product of weight matrices representing each layer, where the columns represent the features, the rows are leg joint position predictions, and the colour of a particular cell indicates how much the feature contributes to the prediction of the joint position. The features are labelled such that the first character is the first letter for the servomotor, e.g. c for coxa, the second character signifies whether the value is set to a servomotor (s) or measured in the servomotor (m), the last digit is how many time steps have passed since the position was set or measured.

columns show the coefficients. The columns are labelled in format $\{\mathrm{c}, \mathrm{f}, \mathrm{t}\}$-$\{\mathrm{m}, \mathrm{s}\}$-$\{0, \ldots, 4\}$, where first distinguishes the joints (c for coxa, f for femur and t for tibia), second signifies whether the value is measured in the joint (m) or set to the servo (s) and the last digit signifies how old the data are (measured in the time steps), e.g. 0 means current measured and set values and 1 means that the values are one time-step old. Similarly, the absolute values of polynomial regressor coefficients are sumarized in Fig. 28 and Fig. 29. Note that the labels used for polynomial regressor are products of the linear regressor labels and that the interception vector is in the first column since it is implicitly part of the polynomial features. For convenience, the matrix is split every 31 columns.

The analysis of the neural network structure is not straightforward, and it is a currently researched field. The ReLU regressors is represented by three matrices, one for each layer: the input matrix $\mathrm{M}_i^{i\times i}$, the hidden layer matrix $\mathrm{M}_h^{i\times h}$ and output matrix $\mathrm{M}_o^{h\times o}$. Visualization of particular matrices would bring hardly any insight. Therefore, the matrix visualized in Fig. 27 is obtained by multiplication of these matrices. For convenience, the shown matrix is transposed. Note that this simple method

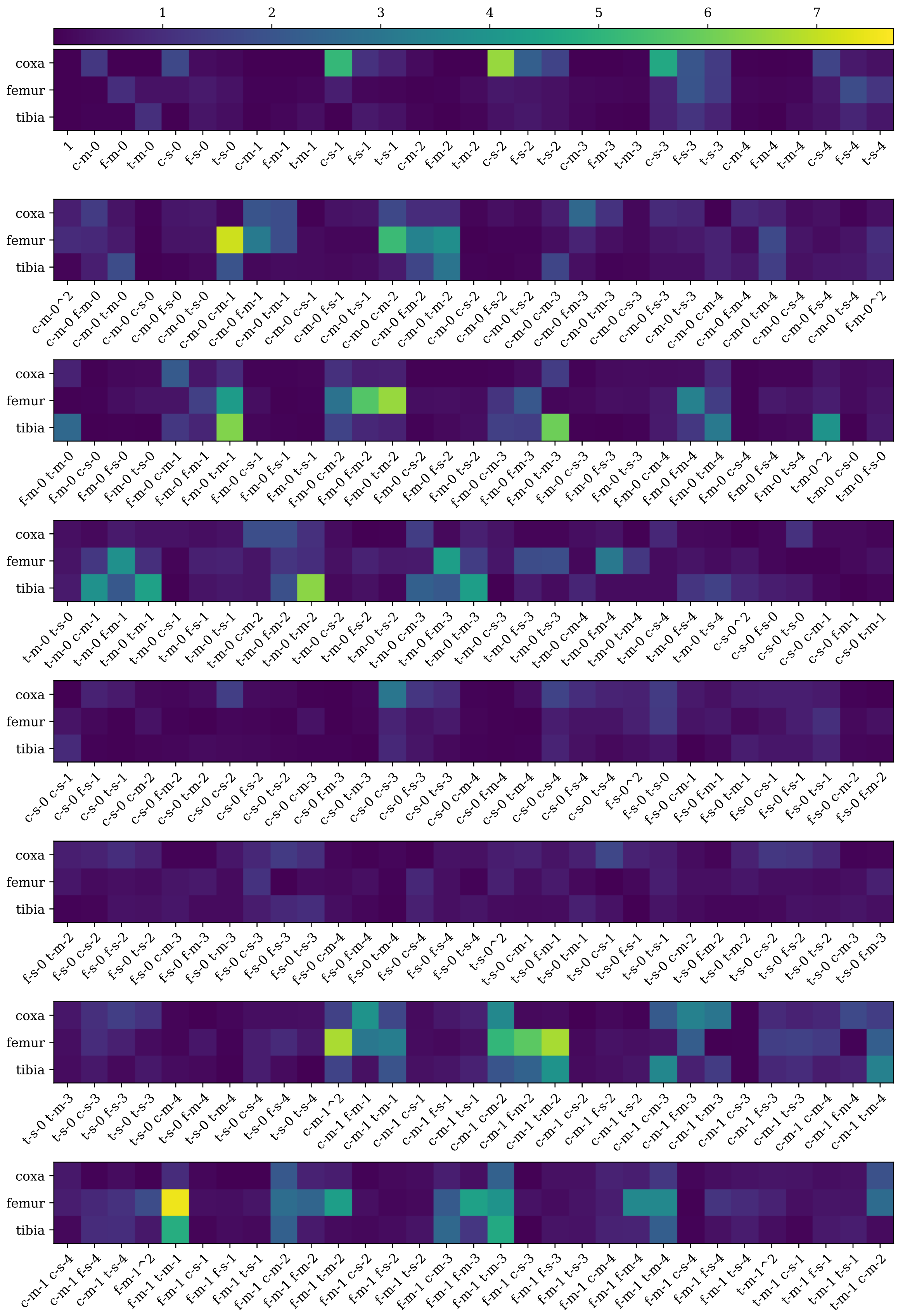


Figure 28: The absolute weights of the polynomial regressor coefficient matrix, where the columns represent the polynomial features, the rows are joint predictions, and the colour of a cell indicates how much the feature contributes to the joint position prediction.

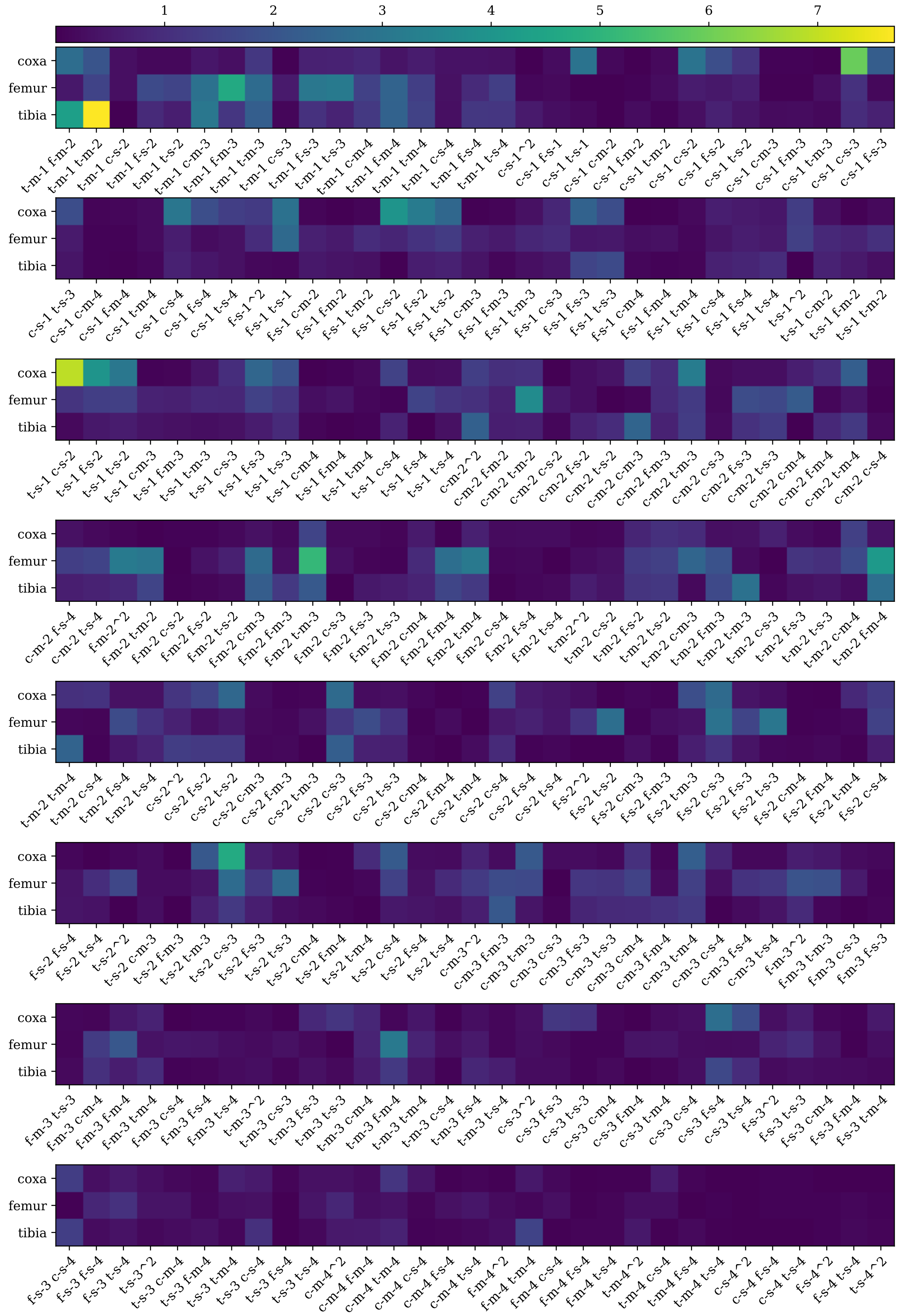


Figure 29: The absolute weights of the polynomial regressor coefficient matrix, where the columns represent the polynomial features, the rows are joint predictions, and the colour of a cell indicates how much the feature contributes to the joint position prediction.

neglects the effect of the ReLU activation function; however, we assume that the effect of the features on the output is preserved.

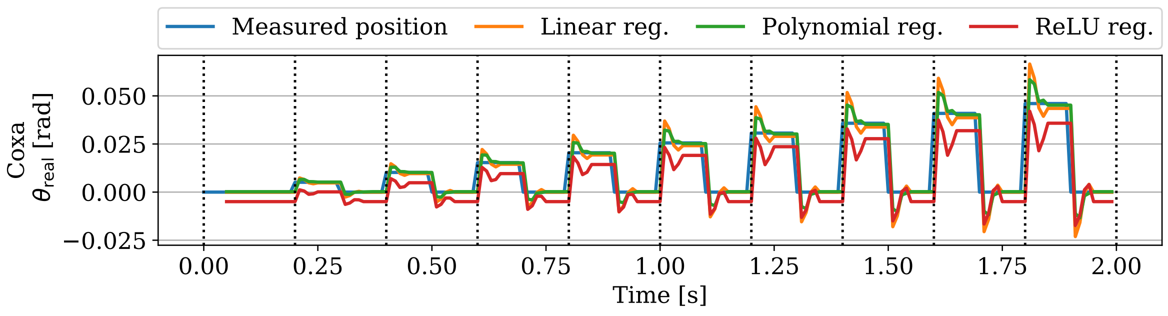
The visualizations show that the predictions of the particular joint are affected most by the last measured value of such joint in cases of the linear and ReLU regressor. Additionally, expected coupling between femur and tibia joint is visible for these regressors as well. Aside from the current measured position, the linear regressor prediction is affected by the past measured position, while the set positions both, current and the past, positions are mostly overlooked. The polynomial regressor matrix is harder to interpret; however, the last measured joint position dependency is also visible. Additionally, the features containing the currently set position have a low impact on the prediction, and the polynomial feature importance decreases with the increasing age of its components. Surprisingly, the set position features are mostly omitted by all regressors for the femur and tibia joint and linear and ReLU regressors for the coxa joint. Note that femur and tibia joints are affected mainly by the foot strike. For polynomial regressor, features containing at least one set position have significantly lower coefficients. Therefore, we deduce that set values have a low impact on the resulting prediction. The experiments conducted to confirm this hypothesis are discussed in the following section.

5.11.3 Influence of Set and Measured Features on Prediction

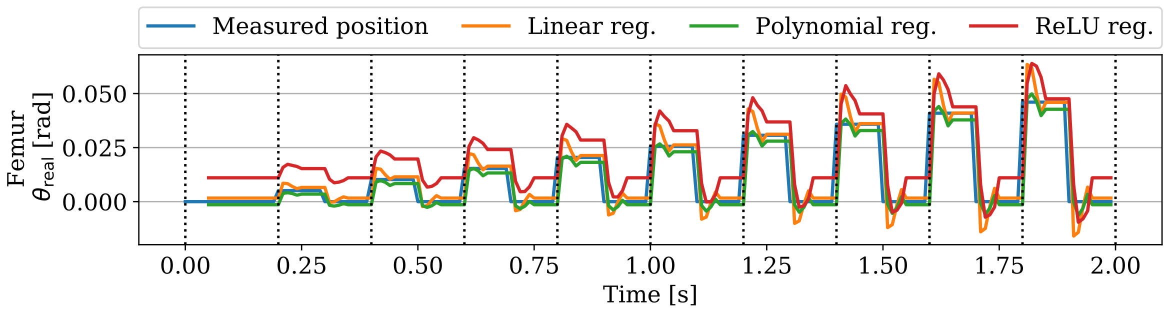
Three arbitrary dataset categories are artificially created to examine the effect of set and measured joint positions on the prediction. The first group of datasets aims to examine how regressors react to changes in the measured data only while the set data remains unchanged. We assume that a correctly trained regressor should ignore these changes up to some extent since the leg is supposed to stay in place given the set position and the measured data contradict the set position. The second group of datasets examines how regressors react to changes in the set data while measured data remains unchanged. In this case, we expect a correctly trained regressor to evaluate that leg is supposed to follow the set position. For these groups, we have selected decoupled approach affecting a single servo at the time rather than an extensive search of all possible joint combinations. The increasing step size of the set position is used to examine how far the regressors could predict without the confirming information provided by measured positions. The third dataset examines how regressors react to continuously increasing servo error in an artificial collision scenario. We believe that the correctly trained regressor predicts that the leg will follow the set position even though the measured position remains unchanged caused by the virtual obstacle.

The first group of datasets, further called *measured-only*, consists of pulses in the θ_{real} for a particular leg joint, while other joint angles are set 0. The pulses have fixed length $l = 20\,\Delta\mathrm{t} = 0.2\,\mathrm{s}$ and duty cycle $D = 50\,\%$, but the amplitude increases by one servo tick $\Delta\theta = 0.005\,\mathrm{rad}$ each new cycle until it reaches 20 ticks. The second group of datasets called *set-only* consists of the pulses in the θ_{des} for a particular leg joint, while other joint angles are set 0. The parameters of the pulses remain the same as in the case of the measured-only dataset. The third dataset called *artificial-collision* consists of the triangular signal in the θ_{des} joint angles, while other joint angles are set 0. The triangular signal has fixed length of $l = 20\,\Delta\mathrm{t} = 0.2\,\mathrm{s}$ and with a settle time of $l' = 30\,\Delta\mathrm{t} = 0.3\,\mathrm{s}$. However, the triangle signal is fed to different leg joints for every cycle to cover all possible combinations. The regressors trained in Section 5.11.1 are used to predict leg position for each of the artificial datasets.

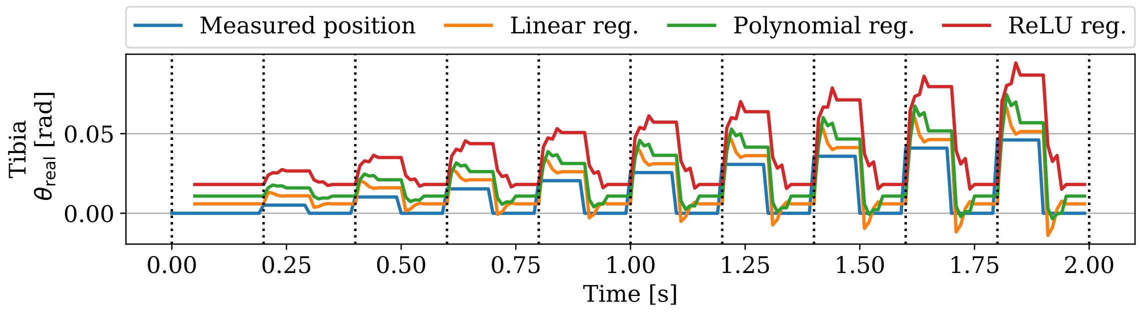
The measured-only datasets depicted in Fig. 30 show that predictions of all regressors are heavily affected by the measured position. For the coxa joint in Fig. 30a and femur joint in Fig. 30b, linear regressor and the polynomial regressor fit precisely to the measured data just a few samples after the measured value has changed. The ReLU regressor behaves in a similar manner; however, the significant static offset is present in all datasets. As shown in Fig. 30c, data in the tibia measured-only dataset affects the tibia position prediction less than in cases of other joints. This experiment confirms that regressors predictions are significantly affected by the measured position, as hypothesized in the previous section.



(a) Coxa joint prediction using artificial measured-only datasetthat contains only measured tibia joints positions.

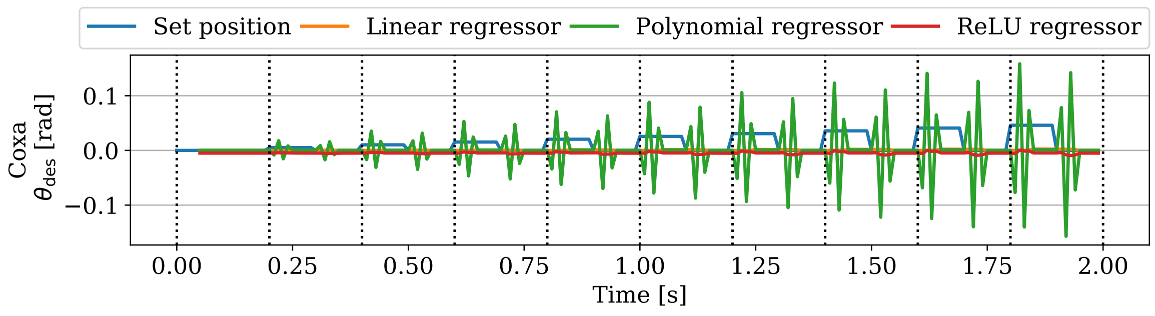


(b) Femur joint prediction using artificial measured-only datasetthat contains only measured tibia joints positions.

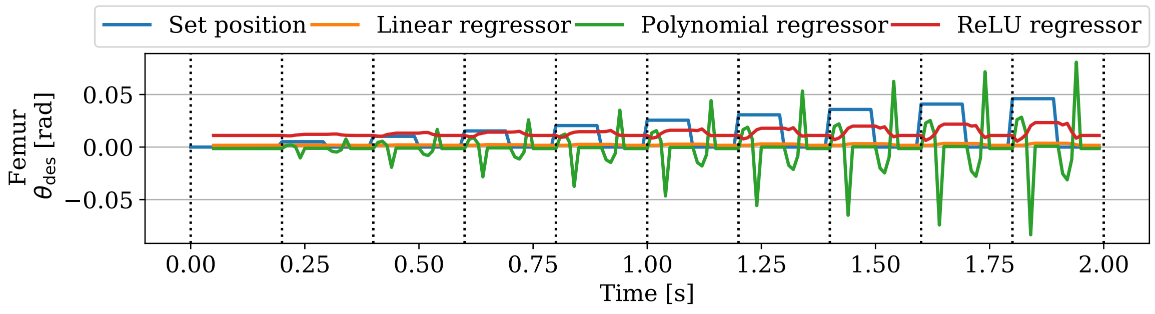


(c) Tibia joint prediction using artificial measured-only dataset that contains only measured tibia joints positions.

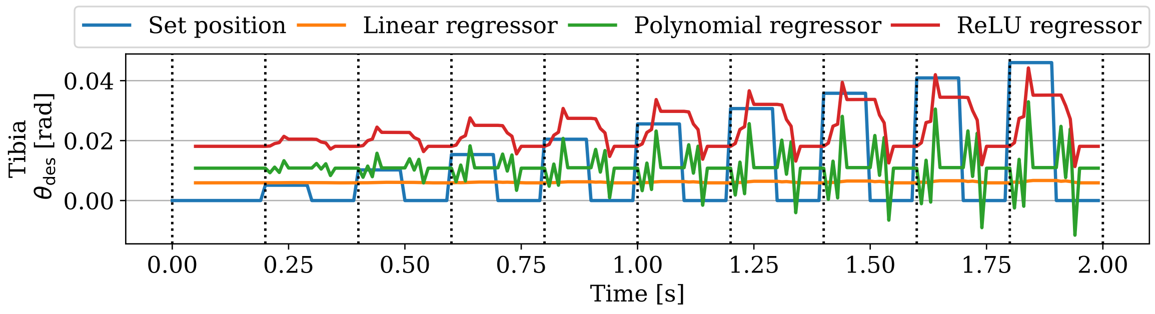
Figure 30: The regressors prediction is based on the artificial datasets consisting of the measured only features for a particular joint to demonstrate how much are the predictions affected by the measured position features. The linear and polynomial regressor predictions heavily depend on the measured features and fit the measured values. The ReLU regressor is significantly affected; however, the static offset is present regardless of the measured prediction features.



(a) Coxa joint prediction using artificial set-only dataset.



(b) Femur joint prediction using artificial set-only dataset.



(c) Tibia joint prediction using artificial set-only dataset.

Figure 31: The regressors prediction is based on the artificial datasets consisting of the set only features for a particular joint to demonstrate how much are the predictions affected by the set position features. The linear and polynomial regressors ignore the set value, though the polynomial regressor reacts wildly to the rising and falling edge in the set position. The ReLU regressor, on the other hand, reacts to the set position in the case of femur joint (b) and tibia joint (c). The coxa joint (a) set features are ignored by every regressor, probably because of the low difference between set and measured data within the training dataset.

The set-only datasets depicted in Fig. 31 show that linear and polynomial regressors mostly ignore the set position when producing the position prediction. The provided figures signify that the set values are neglected by the linear regressor. This statement is supported by the fact that the computed difference between minimal and maximal position predicted by the linear regressor did not exceed the resolution of the servomotors ($0.0026\,\mathrm{rad}$ for coxa joint, $0.0021\,\mathrm{rad}$ for femur joint, $0.0008\,\mathrm{rad}$ for tibia joint, and $\Delta\theta = 0.0051\,\mathrm{rad}$ for a servomotor resolution). The polynomial regressor turbulently predicts positions around the raising and falling edges. This behaviour is likely caused by absent compensation by other polynomial features containing the zero values measurements. Aside from those spikes, the set values are not considered by the polynomial regressor. The ReLU regressor has a significant static offset similar to the previous setup. However, in the case of the femur joint (Fig. 31b) and tibia joint (Fig. 31c), the ReLU regressor reacts to the set positional values the most, and its predictions are closest to the expected behaviour of the well-trained regressor. The θ_{des} values have an inferior impact on coxa joint position prediction for all regressors. We assume that it is caused by the limited coxa movement and speed in the trained data (see Fig. 23 and Fig. 24). Therefore there is significantly lower difference between the set and measured position for coxa joint ($\mathrm{RMSE}(\theta^{\mathrm{coxa}}_{\mathrm{real}}, \theta^{\mathrm{coxa}}_{\mathrm{set}}) = 0.16$) than for the femur ($\mathrm{RMSE}(\theta^{\mathrm{femur}}_{\mathrm{real}}, \theta^{\mathrm{femur}}_{\mathrm{set}}) = 0.43$) and tibia ($\mathrm{RMSE}(\theta^{\mathrm{tibia}}_{\mathrm{real}}, \theta^{\mathrm{tibia}}_{\mathrm{set}}) = 0.37$). Consequently, the coxa set position is omitted. Based on these results, the ReLU seems to perform well in the last artificial-collision scenario since it reacts the most adequate to the position error.

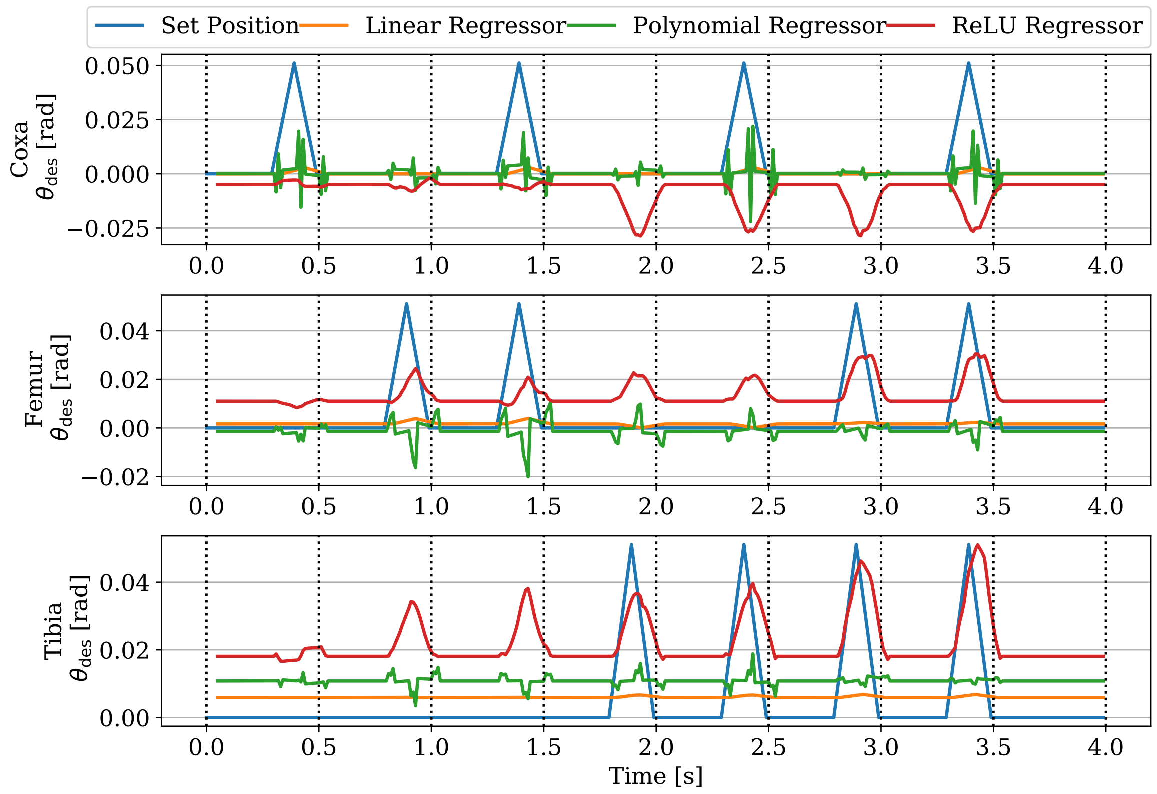


Figure 32: Artificial collision scenario covering arbitrary collisions including every possible joint combinations. The linear regressor reaction to the set position feature is negligible since it predicted position did not exceed the resolution of the servomotor. The polynomial regressor prediction is affected only when the set position changes its difference. The ReLU regresor performs closest to expected performance of well-trained regressor; however, it fails for coxa joint in (a) and unexpected coupling between the coxa joint in (a) and tibia joint in (c) is present.

The artificial-collision dataset is constructed to examine different classes of collision scenarios. Each artificial collision involves different joints, and it is characterized by increasing error between θ_{des} and θ_{real} positions followed by its continuous decreasing. This simulates the situation when the robot leg is attempting to push through the obstacle and then slowly eases the tension. Arbitrary joints combinations are selected to examine different collision types and to involve joint coupling. Note that the servo error is proportional to the torque.

Similarly to the previous experiment, the linear regressor responses to all collision instances are far below the resolution of the servomotors ($0.0028\,\mathrm{rad}$ for coxa joint, $0.0037\,\mathrm{rad}$ for femur joint, $0.0001\,\mathrm{rad}$ for tibia joint, and $\Delta\theta = 0.0051\,\mathrm{rad}$ for a servomotor resolution) as depicted in Fig. 32. The polynomial regressor again reacts wildly to the set value changes and mostly ignore the increasing servo error, except for the femur joint (see Fig. 32), where the leg position slightly follows the set position. However, the response to the set position is negligible compared to the response to the position derivation. Finally, the ReLU regressor responses to the collisions involving the tibia and femur joints look promising, even though in the case of the coxa joint, the ReLU failed miserably. While significant static offset is present for all joints, the ReLU regressor reacts well to the increasing position error of the tibia and femur joints involved the most in the foot-tip collision with the terrain. The noticeable coupling between the tibia and femur joints in collisions between 1.5 to $3.5\,\mathrm{s}$ is expected since the axes of these joints are parallel, in contrast to the coupling between the unrelated tibia and coxa joints.

Based on the behaviour of the regressors in the last experiment, we deduce the following. The linear and polynomial do not react to the set position, which explains why only some collisions are detected in the deployment scenario in Section 5.10. Only if the measured position differs significantly in the first time step of the collision, then these regressors can detect the collision. The increasing error between the measured and set position present during the persisting collision do not affect these regressors enough to increase the error between the measured and expected position. Consequently, the contact is not detected by the locomotion controller and the leg is not stopped.

The circular trajectory has a smoother trajectory in the joint space than the triangular trajectory. Therefore, the sudden changes in the measured position are easily detected by linear and especially polynomial regressors. We conclude that this property of circular trajectory and the hand-tuned results cause the successful deployment of the regressors in the contact detection scenario in Section 5.9. However, we deduce that a reliable contact detection is not possible for the linear and polynomial regressors in any scenario since if the threshold is not exceeded at the exact moment the leg contact begins, it is not detected in the latter time steps.

The static offset of the ReLU regressor present even for no input at all (see artificial-collision scenario between 3.5 to $4.0\,\mathrm{s}$ in Fig. 32) explains why the different thresholds have to be used in the contact detection scenario. The ReLU regressor reacts promisingly in the artificial collision scenarios involving the femur and tibia joints; however, it fails to predict reliably for the coxa joints. To use ReLU regressor for the contact detection, one has to cope with the static offset of the prediction unrelated to the mean values of the training dataset, and only certain collisions involving the femur and tibia joints can be detected. These specific properties make the ReLU regressor unsuitable for reliable contact detection as well.

Chapter 6

Conclusion

In this thesis, the learnable leg state estimator for SCARAB hexapod robot has been developed and benchmarked. In particular, three light-weight machine learning approaches, namely *linear regressor* (Ordinary Least Squares regression (OLS)), *polynomial regressor* (OLS regression with second-order polynomial features) and *ReLU regressor* (three-layer feed-forward neural network with the Rectified Linear Unit (ReLU) activation function) have been used to predict leg position based on the sequence of last six measured and set positions of a particular leg. The leg position predictions are necessary to detect the leg contacts during the rough terrain locomotion of the hexapod robot since deviation from the collision-free dynamics is used to detect collisions as described in Section 4.

The proposed regressors have been investigated experimentally. In the first bundle of the experiments, the leg coupling has been examined (Section 5.1), and the meta-parameters of the proposed regressors have been found (Section 5.4). Thorough examination of the regressors w.r.t. the model precission (Section 5.5), robustness to the parameter changes (Section 5.6), size of training set (Section 5.7) and computational requirements (Section 5.8) have followed. The regressors have been successfully used to detect contacts in Section 5.9. The performance of the regressors deployed on the robot in the rough-terrain traversal scenario exhibit behaviours that were not seen during the evaluation of the regressors. Therefore we have put further effort into explaining the inferior behaviour of the regressors by thoroughly examining the statistical properties of training datasets, the regressors sensitivity to the leg movement speed and the effect of the measured and the set positions on the prediction accuracy (Section 5.11). Finally, the artificial foot contact scenarios have been designed to examine how the regressors react to arbitrary collisions.

The collected results show that despite the promising performance in the initial scenarios, proposed regressors are not suitable for the contact detections since the last measured position is overfitted. In contrast, the set position and the previously measured position is mostly neglected. This results in unreliable foot contact detections. If the contacts are not detected in the first time step when the leg collides with an obstacle, then the linear and polynomial regressors cannot detect contact since they overfit measured positions already affected by the collision. Even though the ReLU regressor reacts to the set position for particular joints, the overall prediction accuracy is poor and suffers from the static offset. However, the methods used to examine the performance of the machine learning-based approaches contributes to our understanding of the prediction process and attributes necessary to develop the contact detection system. Therefore, the methodology can be used to evaluate different types of machine learning approaches.

References

[1] Alana Johnson and Grey Hautaluoma. Touchdown! NASA's Mars Perseverance Rover Safely Lands on Red Planet – NASA's Mars Exploration Program. *NASA*, Feb 2021.

[2] Tina Moore and Amanda Woods. NYPD deploys robot dog after woman shot during Brooklyn parking dispute. *New York Post*, Oct 2020.

[3] Bernard Marr. Demand for these autonomous delivery robots is skyrocketing during this pandemic. *Forbes*, May 2020.

[4] S Nobili, M Camurri, V Barasuol, M Focchi, D Caldwell, C Semini, and MF Fallon. Heterogeneous sensor fusion for accurate state estimation of dynamic legged robots. Robotics: Science and Systems Foundation, 2017.

[5] P. Gonzalez de Santos, J.A. Cobano, E. Garcia, J. Estremera, and M.A. Armada. A six-legged robot-based system for humanitarian demining missions. *Mechatronics*, 17(8):417–430, 2007.

[6] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 33(6):1292–1312, 2017.

[7] Jan Faigl and Petr Čížek. Adaptive locomotion control of hexapod walking robot for traversing rough terrains with position feedback only. *Robotics and Autonomous Systems*, 116:136–147, 2019.

[8] Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanský, Daniel Heřt, Matěj Petrlík, Tomáš Báča, Vojěch Spurný, François Pomerleau, Vladimír Kubelka, Jan Faigl, Karel Zimmermann, Martin Saska, Tomáš Svoboda, and Tomáš Krajník. Darpa subterranean challenge: Multi-robotic exploration of underground environments. In *Modelling and Simulation for Autonomous Systems*, pages 274–290. Springer International Publishing, 2020.

[9] J. Mrva and J. Faigl. Tactile sensing with servo drives feedback only for blind hexapod walking robot. In *2015 10th International Workshop on Robot Motion and Control (RoMoCo)*, pages 240–245, 2015.

[10] M. Camurri, M. Fallon, S. Bazeille, A. Radulescu, V. Barasuol, D. G. Caldwell, and C. Semini. Probabilistic contact estimation and impact detection for state estimation of quadruped robots. *IEEE Robotics and Automation Letters*, 2(2):1023–1030, 2017.

[11] Ross Hartley, Maani Ghaffari, Ryan M Eustice, and Jessy W Grizzle. Contact-aided invariant extended kalman filtering for robot state estimation. *The International Journal of Robotics Research*, 39(4):402–430, 2020.

[12] Shuo Yang, Hans Kumar, Zhaoyuan Gu, Xiangyuan Zhang, Matthew Travers, and Howie Choset. State estimation for legged robots using contact-centric leg odometry. In *arXiv-1911.05176*, 2019.

[13] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive Processing*, 12(4):319–340, 2011.

[14] Klaus Schittkowski. *Numerical data fitting in dynamical systems: a practical introduction with applications and software*, volume 77. Springer Science & Business Media, 2002.

[15] BL Ho and Rudolf E Kálmán. Effective construction of linear state-variable models from input/output functions. *at-Automatisierungstechnik*, 14(1-12):545–548, 1966.

[16] R.J. Full and D.E. Koditschek. Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *Journal of Experimental Biology*, 202(23):3325–3332, 12 1999.

[17] Long Jin, Shuai Li, Jiguo Yu, and Jinbo He. Robot manipulator control using neural networks: A survey. *Neurocomputing*, 285:23–34, 2018.

[18] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

[19] Attila Nemes and Béla Lantos. Genetic algorithm-based optimisation of fuzzy logic systems for dynamic modelling of robots. *Periodica Polytechnica Electrical Engineering*, 43(3):177–187, 1999.

[20] Junji Oaki and Shuichi Adachi. Grey-box modeling of elastic-joint robot with harmonic drive and timing belt. *IFAC Proceedings Volumes*, 45(16):1401–1406, 2012. 16th IFAC Symposium on System Identification.

[21] Erik Wernholt and Svante Gunnarsson. Nonlinear grey-box identification of industrial robots containing flexibilities. *IFAC Proceedings Volumes*, 38(1):356–361, 2005. 16th IFAC World Congress.

[22] T. J. Rogers, G. R. Holmes, E. J. Cross, and K. Worden. On a grey box modelling framework for nonlinear system identification. In Nikolaos Dervilis, editor, *Special Topics in Structural Dynamics*, volume 6, pages 167–178. Springer International Publishing, 2017.

[23] K.C. Tan and Y. Li. Grey-box model identification via evolutionary computing. *Control Engineering Practice*, 10(7):673–684, 2002. Developments in High Precision Servo Systems.

[24] Z. Cen, J. Wei, and R. Jiang. A grey-box neural network based identification model for nonlinear dynamic systems. In *The Fourth International Workshop on Advanced Computational Intelligence*, pages 300–307, 2011.

[25] C. de Prada, D. Hose, G. Gutierrez, and J.L. Pitarch. Developing grey-box dynamic process models. *IFAC-PapersOnLine*, 51(2):523–528, 2018. 9th Vienna International Conference on Mathematical Modelling.

[26] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. In *arXiv-2006.11371*, 2020.

[27] Lennart Ljung. Perspectives on system identification. *IFAC Proceedings Volumes*, 41(2):7172–7184, 2008. 17th IFAC World Congress.

[28] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer Science & Business Media, 2010.

[29] M. Honegger, A. Codourey, and E. Burdet. Adaptive control of the hexaglide, a 6 dof parallel manipulator. In *Proceedings of International Conference on Robotics and Automation*, volume 1, pages 543–548 vol.1, 1997.

[30] K. Vassiljeva, E. Petlenkov, and J. Belikov. State-space control of nonlinear systems identified by anarx and neural network based sanarx models. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010.

[31] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *CoRR*, abs/1901.08652, 2019.

[32] Oliver Herbort, Martin V. Butz, and Gerulf Pedersen. *The SURE_REACH Model for Motor Learning and Control of a Redundant Arm: From Modeling Human Behavior to Applications in Robotics*, pages 85–106. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[33] Stefan Klanke, D. Lebedev, R. Haschke, J. Steil, and H. Ritter. Dynamic path planning for a 7-dof robot arm. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3879–3884, 2006.

[34] A. Angelova, L. Matthies, D. Helmick, and P. Perona. Slip prediction using visual information. In *Robotics: Science and Systems*, 2006.

[35] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal. Learning locomotion over rough terrain using terrain templates. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 167–172, 2009.

[36] Alexander Gepperth and Barbara Hammer. Incremental learning algorithms and applications. In *European symposium on artificial neural networks (ESANN)*, 2016.

[37] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[38] Jay A Farrell and Marios M Polycarpou. *Adaptive approximation based control: unifying neural, fuzzy and traditional adaptive approximation approaches*, volume 48. John Wiley & Sons, 2006.

[39] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[40] R. F. Reinhart and J. J. Steil. Reaching movement generation with a recurrent neural network based on learning inverse kinematics for the humanoid robot icub. In *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pages 323–330, 2009.

[41] Prashant Joshi and Wolfgang Maass. Movement generation with circuits of spiking neurons. *Neural computation*, 17(8):1715–1738, 2005.

[42] Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. Locally weighted learning for control. *Lazy learning*, pages 75–113, 1997.

[43] G. Tevatia and S. Schaal. Efficient inverse kinematics algorithms for highdimensional movement systems. Technical report, 2008. clmc.

[44] Carl Edward Rasmussen. *Gaussian Processes in Machine Learning*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[45] Kumpati S Narendra and Anuradha M Annaswamy. *Stable adaptive systems*. Courier Corporation, 2012.

[46] John J Craig. *Introduction to Robotics*. Pearson Education International, 2005.

[47] Juš Kocijan, Roderick Murray-Smith, Carl Edward Rasmussen, and Agathe Girard. Gaussian process model based predictive control. In *Proceedings of the 2004 American control conference*, volume 3, pages 2214–2219. IEEE, 2004.

[48] J. Steil. Backpropagation-decorrelation: online recurrent learning with o(n) complexity. *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, 2:843–848 vol.2, 2004.

[49] Etienne Burdet, Bernd Sprenger, and Alain Codourey. Experiments in nonlinear adaptive control. In *Proceedings of International Conference on Robotics and Automation*, volume 1, pages 537–542. IEEE, 1997.

[50] Jan Marian Maciejowski. *Predictive control: with constraints*. Pearson education, 2002.

[51] Junfei Qiao, Fanjun Li, Honggui Han, and Wenjing Li. Constructive algorithm for fully connected cascade feedforward neural networks. *Neurocomputing*, 182:154–164, 2016.

[52] F. Abdollahi, H. A. Talebi, and R. V. Patel. A stable neural network-based observer with application to flexible-joint manipulators. *IEEE Transactions on Neural Networks*, 17(1):118–129, 2006. Cited By :193.

[53] Gerardo Loreto and Ruben Garrido. Stable neurovisual servoing for robot manipulators. *IEEE transactions on neural networks*, 17(4):953, 2006.

[54] S Ge Shuzhi, Chang Chieh Hang, and LC Woon. Adaptive neural network control of robot manipulators in task space. *IEEE transactions on industrial electronics*, 44(6):746–752, 1997.

[55] Liangyong Wang, Tianyou Chai, and Chunyu Yang. Neural-network-based contouring control for robotic manipulators in operational space. *IEEE Transactions on Control Systems Technology*, 20(4):1073–1080, 2011.

[56] Deyin Xia, Liangyong Wang, and Tianyou Chai. Neural-network-friction compensation-based energy swing-up control of pendubot. *IEEE Transactions on Industrial Electronics*, 61(3):1411–1423, 2013.

[57] Lianfang Tian, Jun Wang, and Zongyuan Mao. Constrained motion control of flexible robot manipulators based on recurrent neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(3):1541–1552, 2004.

[58] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[59] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[60] H Ding and SP Chan. A real-time planning algorithm for obstacle avoidance of redundant robots. *Journal of Intelligent and Robotic systems*, 16(3):229–243, 1996.

[61] Alexandros Bouganis and Murray Shanahan. Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.

[62] Zhiqiang Cao, Long Cheng, Chao Zhou, Nong Gu, Xu Wang, and Min Tan. Spiking neural network-based target tracking control for autonomous mobile robots. *Neural Computing and Applications*, 26(8):1839–1847, 2015.

[63] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural networks*, 21(4):642–653, 2008.

[64] Seong I Han and Jang M Lee. Precise positioning of nonsmooth dynamic systems using fuzzy wavelet echo state networks and dynamic surface sliding mode control. *IEEE Transactions on Industrial Electronics*, 60(11):5124–5136, 2012.

[65] Gerald P Roston and Eric P Krotkov. Dead reckoning navigation for walking robots. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1991.

[66] Pei-Chun Lin, H. Komsuoglu, and D. E. Koditschek. Sensor data fusion for body state estimation in a hexapod robot with dynamical gaits. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4733–4738, 2005.

[67] S. Chitta, P. Vemaza, R. Geykhman, and D. D. Lee. Proprioceptive localilzatilon for a quadrupedal robot on known terrain. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4582–4587, 2007.

[68] J.A. Cobano, J. Estremera, and P. Gonzalez de Santos. Location of legged robots in outdoor environments. *Robotics and Autonomous Systems*, 56(9):751–761, 2008.

[69] Hassan H. Khalili, Wei Cheah, Tomas B. Garcia-Nathan, Joaquin Carrasco, Simon Watson, and Barry Lennox. Tuning and sensitivity analysis of a hexapod state estimator. *Robotics and Autonomous Systems*, 129:103509, 2020.

[70] Michael Bloesch, Marco Hutter, Mark A Hoepflinger, Stefan Leutenegger, Christian Gehring, C David Remy, and Roland Siegwart. State estimation for legged robots-consistent fusion of leg kinematics and imu. *Robotics*, 17:17–24, 2013.

[71] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart. State estimation for legged robots on unstable and slippery terrain. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6058–6064, 2013.

[72] E. Lubbe, D. Withey, and K. R. Uren. State estimation for a hexapod robot. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6286–6291, 2015.

[73] M. F. Fallón, M. Antone, N. Roy, and S. Teller. Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 112–119, 2014.

[74] Marco Camurri, Milad Ramezani, Simona Nobili, and Maurice Fallon. Pronto: A Multi-Sensor State Estimator for Legged Robots in Real-World Scenarios. *Frontiers in Robotics and AI*, 7(68):1–18, 2020.

[75] A. Chilian, H. Hirschmüller, and M. Görner. Multisensor data fusion for robust pose estimation of a six-legged walking robot. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2497–2504, 2011.

[76] J. Hwangbo, C. D. Bellicoso, P. Fankhauser, and M. Hutter. Probabilistic foot contact estimation by fusing information from dynamics and differential/forward kinematics. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3872–3878, 2016.

[77] Michael Bloesch, Michael Burri, Hannes Sommer, Roland Siegwart, and Marco Hutter. The two-state implicit filter recursive estimation for mobile robots. *IEEE Robotics and Automation Letters*, 3(1):573–580, 2018.

[78] D. Wisth, M. Camurri, and M. Fallon. Robust legged robot state estimation using factor graph optimization. *IEEE Robotics and Automation Letters*, 4(4):4507–4514, 2019.

[79] Gerardo Bledt, Patrick M. Wensing, Sam Ingersoll, and Sangbae Kim. Contact model fusion for event-based locomotion in unstructured terrains. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4399–4406, 2018.

[80] Jemin Hwangbo, Carmine Dario Bellicoso, Péter Fankhauser, and Marco Hutter. Probabilistic foot contact estimation by fusing information from dynamics and differential/forward kinematics. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3872–3878, 2016.

[81] Kei Adachi and Yasunori Nagasaka. Ground contact detection of robot legs by discriminating angular velocities and motor current. In *2017 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, pages 1–7, 2017.

[82] Yilin Xu, Feng Gao, Yang Pan, and Xun Chai. Hexapod adaptive gait inspired by human behavior for six-legged robot without force sensor. *Journal of Intelligent & Robotic Systems*, 88, 10 2017.

[83] Junjie Yang, Hao Sun, Dongping Wu, Xiaodong Chen, and Changhong Wang. Slip model-based foot-to-ground contact sensation via kalman filter for miniaturized quadruped robots. In *Intelligent Robotics and Applications*, pages 3–14, Cham, 2019. Springer International Publishing.

[84] Miloš Prágr, Jan Bayer, and Jan Faigl. Autonomous exploration with online learning of traversable yet visually rigid obstacles. *International Journal of Robotics Research*, 2020. in review.

[85] Yunfei Dong, Tianyu Ren, Dan Wu, and Ken Chen. Compliance control for robot manipulation in contact with a varied environment based on a new joint torque controller. *Journal of Intelligent & Robotic Systems*, 99(1):79–90, Jul 2020.

[86] Donald M. Wilson. Insect walking. *Annual Review of Entomology*, 11(1):103–122, 1966.

[87] Florian Shkurti, Ioannis Rekleitis, Milena Scaccia, and Gregory Dudek. State estimation of an underwater robot using visual and inertial information. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5054–5060, 2011.

[88] Jun Nakanishi, Jay A Farrell, and Stefan Schaal. Composite adaptive control with locally weighted statistical learning. *Neural Networks*, 18(1):71–90, 2005.

[89] Amir Hossein Gandomi and Amir Hossein Alavi. Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, 17(12):4831–4845, 2012.

[90] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, 1995.

[91] Jun Sun, Bin Feng, and Wenbo Xu. Particle swarm optimization with particles having quantum behavior. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 1, pages 325–331 Vol.1, 2004.

[92] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[93] Seiya Tokui, Ryosuke Okuta, Takuya Akiba, Yusuke Niitani, Toru Ogawa, Shunta Saito, Shuji Suzuki, Kota Uenishi, Brian Vogel, and Hiroyuki Yamazaki Vincent. Chainer: A deep learning framework for accelerating the research cycle. In *25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 2002–2011, 2019.