Learning Error-Corrections for Series Elastic Actuators on Humanoid Robots

Thesis · October 2022

DOI: 10.13140/RG.2.2.32160.46086

1 author:

Sebastian Stelter
Cranfield University
3 PUBLICATIONS **8** CITATIONS

SEE PROFILE

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MASTERTHESIS

Learning Error-Corrections for Series Elastic Actuators on Humanoid Robots

vorgelegt von

Sebastian Stelter

MIN-Fakultät

Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Studiengang: Informatik

Matrikelnummer: 6944438

Abgabedatum: 21.10.2022

Erstgutachter: M.Sc. Marc Bestmann

Zweitgutachter: Dr. Chao Zeng

Abstract

Energy conservation and soft robotics are important aspects of humanoid robot development, especially in the context of human-robot interaction. In this thesis, a Series Elastic Actuator (SEA) is introduced to the knee joints of a humanoid robot. As this element adds a positional error to the joint, neural networks are trained to apply a corrective action to each position control command. To provide information for these networks, a new Hall sensor board is added to the robot. Different network architectures and modalities are evaluated. The results show, that using a neural network for solving this task in real-time is possible and returns results comparable to using a PI controller. By adding a compliant element and a hall effect sensor to the knee, this setup also allows for accurate estimation of the motor torque. This estimate is more precise than the estimate of the servo itself.

Zusammenfassung

Die Reduktion des Energieverbrauchs und soft robotics sind wichtige Aspekte humanoider Roboter, vor allem im Kontext der Mensch-Roboter Interaktion. In dieser Arbeit wird ein sogenannter Series Elastic Actuator (SEA) in den Kniegelenken eines humanoiden Roboters verbaut. Um den daraus entstehenden Positionsfehler zu korrigieren, werden neuronale Netze trainiert, die zu jedem Positionsbefehl eine Korrektur hinzufügen. Eine Platine mit einem Hall Sensor wird entwickelt, um den Netzwerken Informationen über die Verformung des SEAs zur Verfügung zu stellen. Es werden verschiedene Netzarchitekturen und die Verwendung unterschiedlicher Modalitäten untersucht. Die Ergebnisse zeigen, dass es möglich ist, ein neuronales Netz für diesen Zweck in Echtzeit zu verwenden. Das Netz liefert dabei Ergebnisse, die vergleichbar mit denen eines PI controllers sind. Durch das hinzugefügte elastische Element und den Hall Sensor kann auch das vom Motor erzeugte Drehmoment genau bestimmt werden. Diese Schätzung ist dabei genauer, als die vom Motor selbst angegebenen Werte.

Contents

List of Figures

List of Tables

List of Acronyms

AMR anisotropic magnetoresistance

DoF degree of freedom

FEA finite element analysis

IMU inertial measurement unit

MAPE mean absolute percentage error

MLP multilayer perceptron

MSE mean squared error

PEA parallel elastic actuator

PHE planar Hall effect

PID proportional, integral, derivative

PLA polylactic acid

PWM pulse width modulation

RISE robust integral of sign of error

ROS Robot Operating System

SEA series elastic actuator

TPU thermoplastic polyurethane

1.Introduction

As the development of robots progresses, interactions between humans and robots become more and more common. Especially in these scenarios, it is crucial to design robots with human safety in mind. One way to ensure the user's safety is soft robotics. By designing actuators that give in under an external load, the risk of injuries can be significantly reduced.

One solution to soft robotics are series elastic actuators (SEAs). By adding a compliant element to an actuator, the actuator gives in to strong external forces while otherwise still being able to be actuated normally. SEAs also provide many additional benefits besides safety. SEAs can store energy and as such greatly reduce both peak and average energy consumption [1] [2]. They also reduce reflected inertia, increase shock tolerance, and increase stability in force control [3].

Despite all these advantages, SEAs are still rarely used in modern robotic applications. One reason is the difficulty of controlling a SEA. Currently, most robot's joints are still designed to be as stiff as possible, to make controlling them more manageable. Direct-drive brushed actuators would be the easiest to control but cannot produce the torques necessary for most robotic tasks. Thus, gear ratio reduction is utilized, trading ease of control for higher torques. This introduces various issues, such as reflected inertia and impedance, which all could be mitigated by a SEA. However, using such a compliant element greatly reduces the actuator's accuracy and complicates controlling it. Thus, they are rarely used.

As an alternative, planetary gear trains can be used, which significantly reduce the effects of reflected inertia. However, those actuators are more challenging to manufacture and control and thus less readily available. They also require higher manufacturing accuracy, as multiple planetary gears simultaneously engage with the sun gear. Misplacing one gear causes an imbalance that can damage the gear train. Recently the trend has gone towards using brushless servo motors, which are more efficient and possess a higher torque-to-power ratio. This means smaller gear ratios can be used, reducing the harmful effects of gear trains. However, as these actuators are more complex and generally more expensive than brushed actuators, this thesis proposes a solution that works with the more readily available and straightforward brushed servos.

While SEAs could mitigate the strong forces applied to the gearbox of the actuator, they also introduce more flexibility into the joint. As the elastic element not only gives in to external forces but also to the intrinsic movement of the actuator, the positional accuracy of the joint is reduced. SEAs are torsion springs (compare Section 3.1). This implies the element behaves equally to torques applied to either side. Therefore, the displacement, which is desired when an external torque is applied, also happens when the actuator is applying a torque. The actuator, therefore, needs to apply a certain amount of torque before the load starts to move. Furthermore, if the load on the other side moves, the actuator must also move to maintain the same torque. This property of the SEA causes complex properties of position, velocity and current [4].

The error introduced here causes inaccuracies in every type of motion. For bipedal

movement, this is especially problematic because the robot frequently is balanced on just one leg. If this leg is not placed correctly, the robot may be unable to keep its balance. Furthermore, if the swing leg has an inaccurate position, it may touch the ground during movement and push the robot off its standing leg.

To reduce this problem, a neuronal network can be trained that learns the error of the SEA and automatically corrects it. This way, existing motion algorithms can continue to be used under the assumption of an ordinary actuator without modification.

Section 1.1 highlights the context of the RoboCup competition as an additional requirement of this thesis. Section 1.2 presents the goals of this thesis, and Section 1.3 gives an overview of the remaining chapters.

1.1. RoboCup

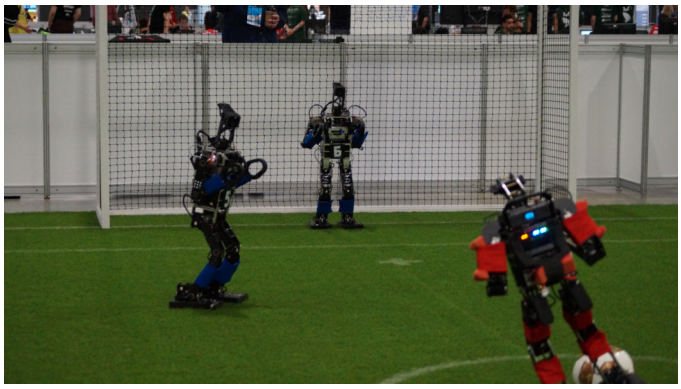


Figure 1.1.: Picture of the RoboCup TeenSize competition 2019. The Hamburg Bit-Bots (**blue**) are defending the goal against an attacker from team ICHIRO-ITS (**red**). Photo courtesy of Florian Vahl.

The *RoboCup* [5] is an annual robotics competition and conference in various disciplines. It was founded in 1997 to further research in robotics by exploring a complex multi-agent real-world environment. The goal of the RoboCup is to create a team of fully autonomous humanoid robots by 2050 that is capable of defeating the human soccer world champion. While the competition started with a sole focus on robotic soccer, various other disciplines, such as rescue robots or home assistance, are now considered as well.

The robots used in this thesis have been designed to perform in the RoboCup Humanoid KidSize League. This means the teams have complete freedom over the robot's hardware and software while still being restricted to a human's shape and sensing capabilities. The league allows bringing humanoid robotics out of controlled lab environments into the real world. It creates a complex multi-agent system and a way to directly compare approaches in a well-defined scenario. An example of the competition, as well as the Wolfgang-OP robot platform used in this thesis, can be found in Figure 1.1.

Two main focuses of the league are stable motions and energy efficiency. Bipedal locomotion is inherently difficult and requires a lot of small error-correcting motions. Some of the strongest muscles in the human body are located in the legs, which illustrates the necessary forces for these motions. However, the size of the actuators used in the league's robots is restricted by size limitations and the requirement of complete autonomy. A fully autonomous robot is required to carry its energy source on its body and thus can only be equipped with a comparably weak battery. Because of this, it is essential to move as efficiently and safely as possible.

1.2. Thesis Goal

This thesis evaluates whether using SEAs is feasible and beneficial for smaller-size low-cost humanoid robots. So far, most approaches utilizing SEAs have used powerful actuators and large torsion springs. However, these cannot reasonably be integrated into a smaller humanoid due to cost and spatial constraints. Most robots in the Humanoid KidSize League use Dynamixel actuators which only produce a relatively small torque and have a considerable amount of backlash. Combined with a weaker torsion spring, this makes solving the problems of the SEA significantly harder to implement. This is because environmental forces, such as gravity, have a relatively more minor influence than the torques from the actuator if both the spring and actuator are stronger. This means that on a weaker spring, external forces will alter the position of the SEA significantly more than on a stronger spring, which makes controlling it harder.

In this thesis, a 3D-printable SEA design is modified and parametrized to generate compliant elements of different spring constants. It is evaluated which of these configurations is most suitable for use in the knee motors of the Wolfgang-OP robot. A small Hall sensor board is developed to measure the displacements caused by the SEA. Using this sensor, as well as other modalities of the robot, software is developed to record datasets during the regular operation of the robot easily. These datasets are used to train lightweight neural networks which can correct the errors introduced by the compliant element in real time.

The performance of the networks, as well as the impact of the different modalities, are evaluated both on recorded data and on the real robot. It is also evaluated which factors could negatively impact the system's accuracy. Finally, the energy efficiency of the SEAs is also evaluated.

1.3. Thesis Outline

Chapter 2 gives an overview of related work in the field. Chapter 3 goes over some concepts required for this thesis, while Chapter 4 describes the existing hardware and software utilized for this approach. The approach is subsequently introduced in Chapter 5. The work is evaluated in Chapter 6, and the findings are discussed in Chapter 7. Finally, the thesis is concluded in Chapter 8.

2.Related Work

SEAs were first introduced by Pratt et al. in 1995 [3]. Their design uses a planetary gearbox and a steel torsion spring to add compliance to the actuator. They recognize a variety of benefits over direct drive or gear train based actuators, such as lower reflected inertia and energy storage (compare Section 3.1). They also propose a force control system based on proportional, integral, derivative (PID) controllers. This system achieved an overall small torque error but significant torque errors for impedance around zero.

The control of SEAs has been studied extensively in the past. Currently, there are three main approaches to mitigating the adverse effects of SEAs. The problem can be solved using PID controllers, machine learning, or mechanical solutions. The vast majority of approaches fall into the first category.

While all of the approaches displayed below show promising results, it should be noted that both actuators and compliant elements used in all of these papers are significantly more powerful. More powerful actuators and higher spring constants significantly reduce the adverse effects of SEAs. However, these actuators could not be mounted in smaller-size humanoid robots due to spatial constraints and often are not affordable, especially to consumers.

2.1. PID Controllers

The majority of works use PID controllers to control SEAs. In the simplest form, a single controller is used, while more complex solutions use cascading controllers to achieve better results. Unless otherwise specified, most approaches shown in this section control the actuator's torque.

Seven years after the first SEA was introduced, Pratt et al. describe the use of a single PD controller to control the torque of the robot and thereby the leg movement [6]. They report high force fidelity and low impedance, making such an approach suitable for legged robots. However, no experiments are shown in the paper to prove its efficacy.

Kong et al. also use a PD controller for torque control but additionally add a disturbance observer to the model [7]. In doing so, they can compensate for the dynamics of the SEA and cancel out undesirable disturbances. They also add a feed-forward filter to reduce closed-loop dynamics. This control system was used to develop an actuated prosthesis for orthosis treatment. In the conducted experiments, it could be concluded that the offset from the target position and the torque error were small enough not to be noticed by the patient.

A similar approach was used by Fotuhi et al. [8]. Instead of adding a disturbance observer, however, they use a non-linear friction estimation model to make the system more robust.

Wyeth et al. proposed a system of two cascading PI controllers to combat the issue [9]. The inner loop controls the velocity of the actuator, while the outer loop controls

the torque. This system manages to control the actuator accurately and improve significantly on the SEA performance. While the original paper only performs experiments in simulation, this approach was picked up by several other scientists to develop robotic protheses [10] [11]. Sergi et al. extend upon this model by adding a PD controller as an outer loop, controlling the impedance of the system [12]. This makes their prosthesis also resistant to external disturbances.

Fernández et al. further modified this approach [13]. Their cascading PID controllers control (from inner to outer loop) current, velocity and position. Furthermore, they added a PID deflection controller parallel to the cascading controllers. A dynamic Gaussian mixture model was used to generate the reference deflection, while the actual deflection was measured with two position sensors. The model achieved a mean torque error of less than 4 per cent.

Zhu et al. use a PID controller as an inner control loop for torque control. An adaptive position controller extends this loop. A Kalman filter is used to increase the convergence rate of the adaptive parameters. This allows the system to react quickly to rapid load changes. The system was tested on a single-legged hopping robot and has proven to be robust.

Finally, the Cheetah 3, developed by Bledt et al., also supports PD control as one of its control algorithms [14]. Even though the Cheetah does not possess any SEAs, their control architecture allows it to walk on uneven terrain, slopes and stairs without issue, indicating that they successfully mitigated most of the problems of non-compliant actuators. They do so by letting their control algorithm distinguish between support and swing legs and choosing different strategies for each case. Furthermore, they use planetary gearboxes with a low gear ratio of 7.67:1. This reduces the adverse effects of gear trains and provides a certain amount of compliance without using a dedicated compliant element.

2.2. Machine Learning

Some approaches have already attempted to solve the issues of SEAs by using machine learning. Hwangbo et al. trained a multilayer perceptron (MLP) to learn the error caused by the compliant element [15]. This model is then used to enhance a simulated environment of the robot. In this environment, reinforcement learning is used to train a network to execute different motions, which can then easily be transferred to the real robot. This approach achieved a torque error of less than 1 Nm on actuators that can exert up to 40 Nm. The system has been evaluated on the ANYmal quadruped robot platform and was able to perform both walking and fall recovery precisely. However, learning the individual motions takes a long time (79 days of simulation time for fall recovery), and it needs to be retrained for each individual motion.

Pavlichenco et al. trained two networks to solve the problem [16]. The first network predicts the direct dynamics from the current state of the robot and the target. The second network then takes the same input data and the prediction of the first network to predict the inverse dynamics. The dataset to perform the training was collected over 45 minutes on a Baxter robot and consists of a collection of pick and place tasks.

Training took around two hours and resulted in an average end effector position error of 1.015 cm, which is an improvement of 66% compared to the baseline.

Agrawal et al. trained a neural network to grasp physical concepts intuitively [17]. They collected samples of the robot poking different objects and recorded how the objects would move when manipulated. These samples were then used to train a Siamese network, which could predict both, which state the world would be in after a certain poke, and what poke would be necessary to cause a particular state of the world. This approach produces promising results and could be extended to predict how to manipulate a SEA.

2.3. Other Approaches

Despite most approaches falling into the first two categories, several researchers have devised unique solutions to the problem.

Lanh et al. use a model reference adaptive controller to mitigate the effects of SEAs [18]. They create a reference model of a SEA, that the controller can follow. The approach tracks the desired position closely but has only been evaluated in simulation.

Similarly, Hutter et al. also use a model to control their ScarlETH robot [19]. They use a virtual model controller that generates virtual components, which interact with the physical components of the robot. Their control scheme is divided into two parts. The virtual model controller is used during ground contact, whereas a Raibert controller is used during the flight phase. This approach tracks desired trajectories closely and allows the robot to traverse uneven terrain.

Wei et al. use feedback linearization as a control scheme [20]. They use the robust integral of sign of error (RISE) method to add further robustness to the system and manage to follow trajectories with relative certainty. However, they only provide results from a very simplified simulation.

Finally, Chang et al. developed a mechanical solution to the problems of SEAs [21]. They add an inerter to the robot to dampen its movements and a wobbling mass to counteract the movements of the SEA. This design reduced energy consumption by 30% and increased the robustness of the walking algorithm. Unfortunately, this approach was only tested in simulation.

3.Fundamentals

The following chapter explains the mechanical and physical principles used in this thesis. The robots in this thesis are equipped with SEAs and parallel elastic actuators (PEAs). The SEA is mainly used to protect the actuators and gearboxes, as well as to conserve energy. Its functions are explained in detail in Section 3.1. The PEA is used to reduce the energy consumption of the actuators and to prevent the actuators from overheating. It is described in Section 3.2. Finally, the use of Hall sensors as rotational encoders is explained in Section 3.3.

3.1. Series Elastic Actuators

SEAs are a simple and efficient way to add compliance to an actuator. SEAs consist of three components: A motor, a gear train and a spring (compare Figure 3.1). While direct drive is preferable from a controllability perspective in many cases, a gear train is usually necessary when higher torques are required. This is because most electrical motors have a low torque density and thus cannot generate large forces at low speeds. Gearboxes, however, introduce many problems to the system:

Reflected Inertia Inertia describes an object's resistance to a change in velocity. The higher the inertia of an object is, the more force is necessary to alter its velocity. The relationship can be described as

$$F = J \cdot \alpha \tag{3.1}$$

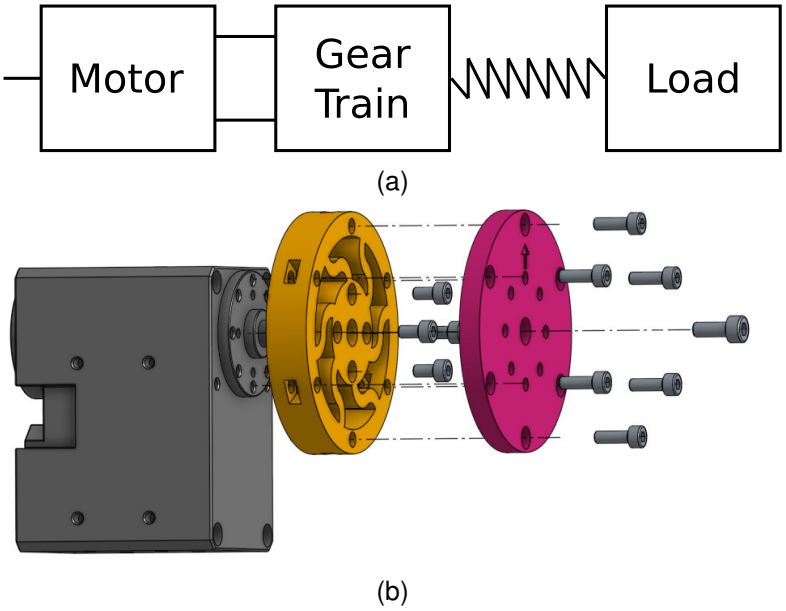
where F is the required force, J is the inertia, and α is the angular acceleration of the actuator [22]. As a motor actuates a load, the load's inertia is reflected back to the motor. For a direct drive system, the total inertia is simply the sum of the motor inertia and the load inertia:

$$J_{total} = J_{motor} + J_{load} \tag{3.2}$$

However, for a system with a gear train, the relationship is quadratically dependent on the gear ratio:

$$J_{total} = J_{motor} + \frac{J_{load}}{N^2} \tag{3.3}$$

where N is the gear ratio. This is not problematic for the regular use case of the motor actuating a load since a larger gear ratio will reduce the load inertia. However, in the case of a shock load, e.g. the robot falling onto the lever attached to the gear train or the robot stomping on the ground, the scenario can be interpreted as the load actuating the actuator. This means the gear ratio is inverted, and the inertia can get very high, causing damage to the gear train or the motor. For example, with the gear ratio of



(a)

(b)

Figure 3.1.: SEA schematic (a) and example (b). The motor and gear train are integrated into the Dynamixel XH540-W270 shown in (b). A compliant element, consisting of an elastic component (**yellow**) and a solid component (**red**) is attached to upgrade it to a SEA.

$272.5 : 1$ in the Dynamixel XH540-W270 servos used in this thesis, this would mean the inertia is increased by a factor of over $74,000$.

Backlash Within a gear train, gears usually do not interlock perfectly. Due to manufacturing inaccuracies, each gear can move a tiny amount without interacting with the other gears. As gear trains of most modern servos consist of multiple gears to reach the high gear ratios required, the relatively small movements of each gear accumulate to allow a larger amount of free movement in the gear train. Over time, through wear and deformation from impacts, this effect can be significantly increased. This means that for each position of the servo, there exists an area in which the load can move freely without the possibility of correcting it. This phenomenon is called backlash.

Backlash causes the system to undershoot targets and create undesirable results, especially in high-precision environments. In the worst case, these problems can cause gears to slip, i.e. skip a tooth, or break.

Backdriveability and Impedance Backdriveability describes an actuator's ability to be used in the opposite direction. This means moving the load attached to the actuator with as little resistance as possible. A high backdriveability is desired, as it increases the impact resistance of the actuator. It also provides safety in a human-robot environment since actuators with high backdriveability will give in to external forces, and thus are less likely to hurt a human.

Mathematically, backdriveability can be described as mechanical impedance. A higher impedance results in lower backdriveability. Mechanical impedance can be compared to the more well-known electrical impedance. While electrical impedance measures the opposition to an alternating current, mechanical impedance measures the opposition to a harmonic force. A harmonic force, similar to an alternating current, oscillates and can best be illustrated with the example of a swinging pendulum. Harmonic forces can be described by

$$F = -k \cdot x \tag{3.4}$$

where x is the displacement from the equilibrium position and k is the force constant. Equation 3.4 implies that the force is always directed opposite to the displacement. The mechanical impedance Z is then defined as

$$Z = \frac{F}{v} \tag{3.5}$$

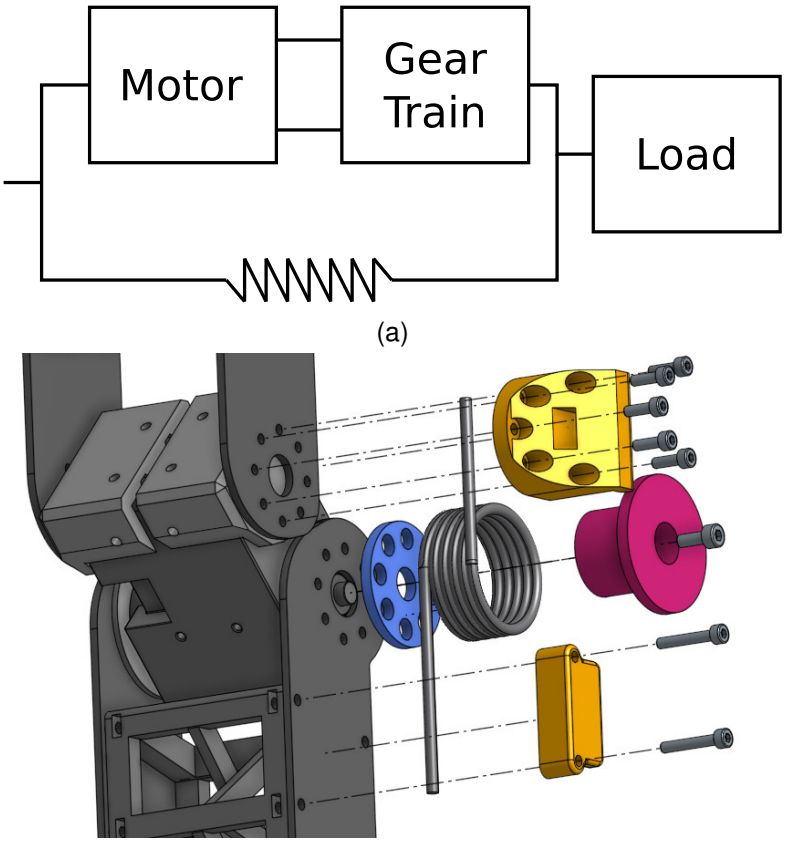
where v is the velocity [23].

Geared actuators have a relatively high impedance, meaning strong forces must be applied to move the actuator via the load. This issue can be addressed via software solution [24] or passively via SEAs.

Torque Ripple and Friction Gearboxes are manufactured with certain tolerances in accuracy. These inaccuracies often lead to misalignment of the gear axes. Furthermore, in the case of Dynamixel motors, gearboxes are often replaced by the end user, trading in easy accessibility for less accuracy. These issues often cause torque ripple. This means that as the motor rotates, the torque will periodically increase or decrease depending on the orientation of the gears. This issue is especially problematic in torque control settings and can be mitigated by applying a counter torque.

As the gears in the gearbox touch, friction is applied. Due to the same reasons as mentioned above, especially when the gearbox is not adequately lubricated, the friction in the system may be uneven. This can cause further inaccuracies in the movement of the actuator.

Benefits of SEAs All the aforementioned issues can be mitigated by adding a compliant element to the gear train. By adding an element that gives in to irregular movements, the effects of reflected inertia, backlash, torque ripple and friction are dampened. Adding an element that gives in to external stimulus significantly increases backdriveability, as impedance is reduced. Besides these benefits, SEAs also dampen impacts, protecting the gearbox. Furthermore, SEAs can help reduce energy consumption, especially by reducing the peak forces required.



(a)

(b)

Figure 3.2.: PEA schematic (a) and example (b) In contrast to the SEA, the elastic element of the PEA is mounted in parallel with the motor and gear train on the outside of the leg. In our case, a torsion spring is used and held in place by the **yellow** and **red** components.

3.2. Parallel Elastic Actuators

While SEAs are mounted in series with the actuator, PEAs instead take a parallel relationship (compare Figure 3.2). With this change in relationship, the properties of the element also change. The primary purpose of the PEA is reducing the torque acting on the actuator by providing a counter-torque.

In the case of the Wolfgang-OP, the PEA is a steel torsion spring and is used to reduce the effects of gravity. The robot spends most of its time in a pose from which it can immediately start walking. This pose has the legs of the robots bent at a 60 degree angle. Therefore, a considerable amount of torque is exerted on the actuator by the weight of the torso. With a spring providing torque in the opposite direction, this effect is minimised. This conserves energy, as well as reduces overheating problems in the actuator.

However, while a PEA reduces the torques required while the robot is standing, the actuator needs to apply additional forces when the leg is lifted into the air, as nothing is counteracting the torque applied by the spring. Therefore, it is important to find the right balance. The torsion spring used in the Wolfgang OP reduces the motor torque by up to 37% with a spring constant of $k = 0.76$ [25].

3.3. Hall Effect Sensors

Hall effect sensors measure magnetic fields. The Hall effect, named after its discoverer, Edwin Hall, describes that a magnetic field can produce a measurable voltage in a conductor perpendicular to an applied current [26]. For any effect to be measurable, the voltmeter and the magnet must be arranged in a certain way (compare Figure 3.3). Suppose a current is applied to a conductor in one direction, and the magnetic field is applied along the second direction, orthogonal to the current. In that case, the change in voltage can be measured along the third direction, perpendicular to both the magnetic field and the current. It is noteworthy that the voltage remains measurable as long as the magnetic field is applied instead of just peaking. This effect appears due to the Lorentz force exerted by the magnet on the current, which slightly curves the electrons in the conductor towards one side, which creates a difference of potential inside the conductor, and thus a measurable voltage.

While traditional Hall effect sensors are capable of measuring the orientation of a magnetic field, for example, by building a Wheatstone bridge, modern rotary position sensors tend to make use of the planar Hall effect (PHE) instead. This bears the advantage that the magnetic field does not need to be applied perpendicular to current and voltage but can instead be applied in the same plane. Even though the name of this effect suggests a similarity to the Hall effect, the PHE is more closely related to the anisotropic magnetoresistance (AMR). Whereas the AMR measures the longitudinal resistance of the sensor, the PHE measures the transverse counterpart [28]. These effects can only be discovered in ferromagnetic metals. Given sufficient strength, the internal magnetisation aligns with the external magnetic field when an external magnetic field is applied to such material. The direction of this internal magnetisation influences

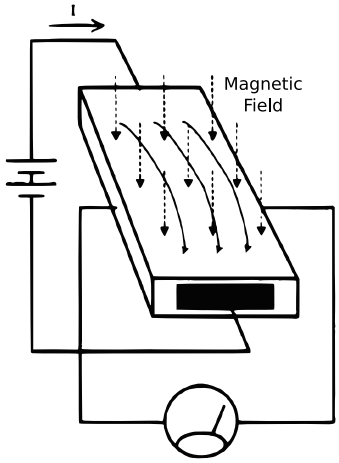


Figure 3.3.: Basic setup of a Hall effect sensor. A current I flows through the sensor, while a voltmeter is added orthogonally. If a magnetic field is applied perpendicular to both, a change in current can be measured. Modified from [27].

the resistance. If the magnetisation is orthogonal to the current, the resistance is minimal; if it is parallel, the resistance is maximal. The PHE resistance can be described as

$$\rho_{phe} = \Delta\rho \cdot \sin\theta \cdot \cos\theta = \frac{\Delta\rho}{2} \cdot \sin 2\theta \tag{3.6}$$

$$\Delta\rho = \rho_{\perp}\rho_{\parallel} \tag{3.7}$$

where ρ describes the resistance in either parallel or perpendicular direction, and θ describes the angle of the magnetic field [29]. This means the PHE only directly depends on two values, the resistances of the sensor and the orientation of the magnetic field. As the resistance is constant, the PHE is a direct projection of the orientation of the magnetic field. Equation 3.6 indicates that this projection takes a sinusoidal shape.

To now actually measure the orientation, a diametric magnet is mounted on the rotating element exactly above the sensor. Compared to an axial magnet, which is magnetised along the axis, a diametric magnet is magnetised perpendicular to its axis (compare Figure 3.4.a). By changing the rotating element's orientation, the magnetic field's orientation also changes, which causes the effect described above.

The values measured by a planar Hall sensor form a sinusoidal wave, with one complete revolution resulting in one phase (Compare Figure 3.4.b). Because of this, sensor readings are ambiguous for angles larger than $\pm 180°$. Multiple Hall sensors with a $90°$ phase shift can be used to avoid this problem. While two sensors would be sufficient to uniquely encode each position, most modern sensors, including the one used in this thesis, use four sensors arranged in a circle instead. This has the advantage that offset errors and mechanical stresses cancel out between opposing sensors, and thus have less impact on the sensor reading [31].

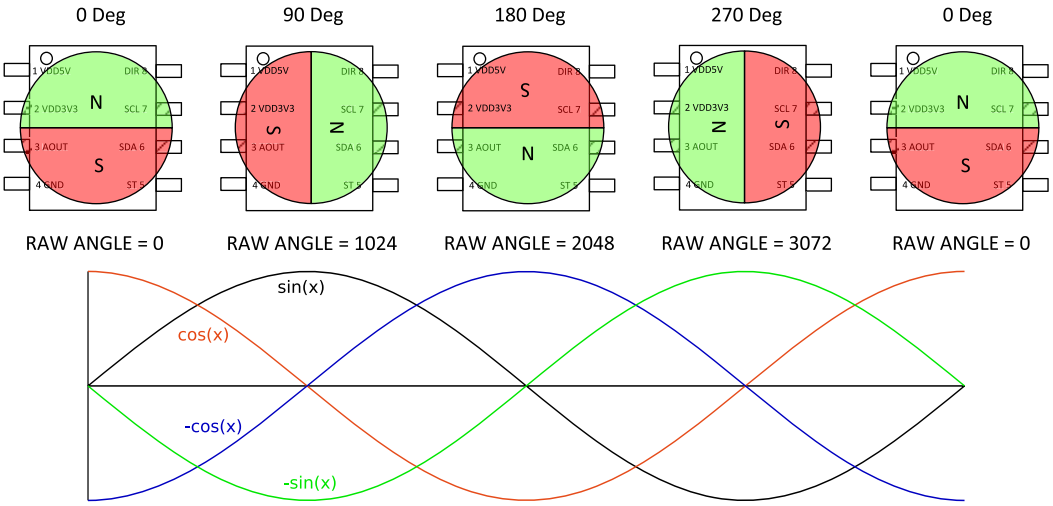


Figure 3.4.: Relations between magnet orientation and sensor readings. Magnet orientations in (a) relate to the readings directly below it in (b). The **black** wave describes the readings of a planar Hall sensor without any offset. The **red**, **green** and **blue** waves display a planar Hall sensor with offset of 90, 180 and 270 degrees, respectively. Modified from [30].

A sine and a cosine wave are generated by subtracting the readings of opposing sensors. Utilising the relationship

$$\tan\theta = \frac{\sin\theta}{\cos\theta} \tag{3.8}$$

the resulting angle can easily be calculated as

$$\theta = \arctan\frac{\sin\theta}{\cos\theta} \tag{3.9}$$

where $\sin\theta$ and $\cos\theta$ are replaced with the sensor readings.

4.Robot Platform

In this Chapter, the robot platform used in this thesis is presented. The *Wolfgang-OP* robot has been used by the Hamburg Bit-Bots since 2018 [32]. The robot platform was initially developed by the WF-Wolves, a RoboCup team from the Ostfalia University of Applied Sciences. It is based upon the *NimbRo-OP* [33], but has been heavily modified since then. The robot has been designed with the RoboCup competition in mind and has performed in the Humanoid League KidSize since then. As such, it has been fitted with human-like sensors and body structure.

Section 4.1 goes into more detail about the hardware properties of this robot platform, while Section 4.2 describes the software stack used by the Hamburg Bit-Bots, especially those components relevant to this thesis.

4.1. Hardware

In this Section, the *Wolfgang-OP* platform is described in more detail. Section 4.1.1 describes the mechanical structure and layout of the robot, as well as the type and position of the robot's sensors. Section 4.1.2 goes into more detail about the actuators used in the robot.

4.1.1. Structure

The *Wolfgang-OP* is a humanoid robot with 20 degree of freedom (DoF). It weighs around 8 kg and is approximately 80 cm tall. 6 DoF are present in each leg, 3 DoF in each arm, and 2 DoF in the head. This layout results in five kinematic chains when viewed from the base link in the torso of the robot. The layout of the robot can be seen in Figure 4.1.

Both SEAs and PEAs have been used in this robot prior to this thesis, albeit in different use cases [25]. The SEAs used in this robot are fully 3D printed from an elastic filament[1]. The geometry of the SEA is inspired by the design proposed by Martins et al. [34]. Three SEAs have been mounted in either shoulder roll motor and the head pitch motor. These mainly serve the purpose of protecting the gearboxes of the actuators in case of a fall. However, the addition of these SEAs does not lead to the problems expected for the knee SEAs, as positional accuracy is not required in these motors. The shoulder roll motors are only used for the stand-up motions of the robot, which are robust against inaccuracies along the lateral axis. The head carries the camera and therefore is prone to inaccuracies in position, as minor errors lead to huge differences when transforming the 2D images into 3D space. This problem, however, is avoided by a secondary inertial measurement unit (IMU) mounted directly below the camera.

The robot also already possesses PEAs mounted in the knees. These serve the purpose of preserving energy and reducing peak torque. As the robot's knees are in a bent

[1] `https://ninjatek.com/ninjaflex/`

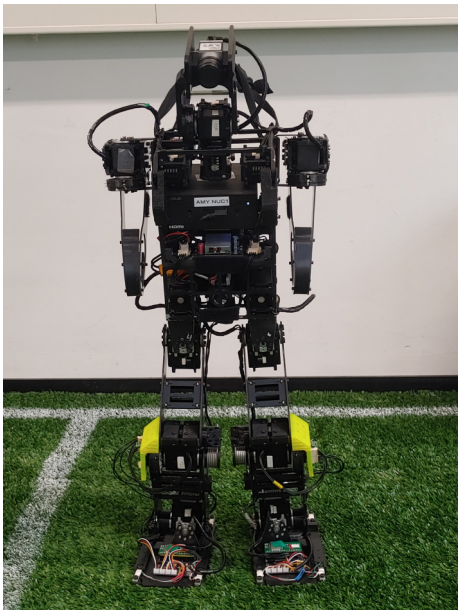


Figure 4.1.: Wolfgang-OP robot with modifications for a SEA in the knee. The Hall sensor is visible in bright yellow.

position most of the time, the force applied to these two motors is significantly higher than the other actuators. This often leads to overload errors and reduced performance, especially with rising temperatures and diminishing battery charge [25].

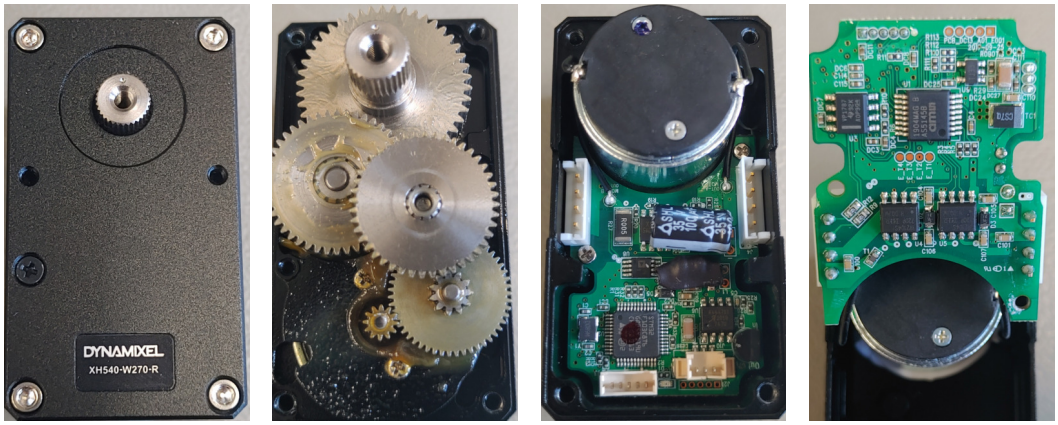
Since the robot used in this thesis has been designed for the RoboCup Humanoid context, it only possesses human-like sensors. The robot is fitted with an industrial-grade camera with a global shutter, two IMUs, one in the torso and one in the head, as well as four load cells in each foot. It also possesses several internal sensors for measuring motor temperature, current and position.

4.1.2. Actuators

The robot platform uses three different types of brushed coreless servo motors, all produced by Robotis[2]. The weaker but lighter MX-64 are used in the head and arms, where large torques are not required, while the stronger but heavier MX-106 are used in the legs. The knees use the XH540-W270, which provides even better performance, as these motors perceive the most stress in the robot.

Besides the DC servo, the XH540-W270 contains a gearbox with a ratio of 272.5:1, a rotary position encoder, and some electronics for communication (compare Figure 4.2). The XH-540 uses the AS5045 as a rotary position encoder, which functions similar to the sensor described in Section 3.3, albeit using a Hall sensor array instead of the circular formation. The sensor output still follows the same shape and scale as the external sensor used in this thesis (compare Section 5.2.1).

[2] `http://www.dynamixel.com/`



(a) (b) (c) (d)

Figure 4.2.: A Dynamixel XH540-W270 actuator. Figure (a) shows the closed motor, (b) displays the gear train, and (c) and (d) show the internal electronics used for communication and analysis.

The XH-540-W270 used as the knee actuator in this thesis has a stall torque of 11.7 Nm and a no-load speed of 46 rev/min at 14.9 V[3]. As our robot is actuated with 15.4 V , these values are expected to be slightly higher. The motor possesses a nominal backlash of $0.25°$.

The servos and the sensors are connected to the robot computer via a QuadDXL board [35]. This board splits the servo communication into four separate buses. This results in an update rate of over 1,000 Hz for the sync read/write cycle. The motors communicate by using the Dynamixel protocol[4] via an RS-485 connection. The protocol can be translated and interpreted by the *ros-control* node and is then further processed as a ROS message (Compare Section 4.2.2).

4.2. Software

The robots in this thesis use a codebase developed by the Hamburg Bit-Bots, based on the Robot Operating System (ROS). Both are explained in the following Sections.

4.2.1. ROS

ROS is a robot development middleware which provides various robotics-related tools and libraries. ROS is a node-based architecture with message-based communication that allows components to be easily switched out or modified without changing other parts of the codebase. It, therefore, is easy to develop novel algorithms or reuse existing

[3] https://emanual.robotis.com/docs/en/dxl/x/
[4] https://emanual.robotis.com/docs/en/dxl/protocol2/

components in a different context. Many standard solutions are already available as ROS packages.

In ROS, components are modelled as nodes, which communicate with each other by publishing or subscribing to topics. This asynchronous communication allows vastly different algorithms to work together effortlessly. Services or actions can be used instead for scenarios where synchronous communication is required. ROS also supports communication between different programming languages and devices. This allows C++ components to interact directly with Python modules without needing bridges. Therefore, programming languages can be chosen freely based on the use case without needing to consider interoperability.

ROS2

During the development of this thesis, the Bit-Bots Codebase transitioned to using ROS2. ROS2 has been carefully designed to improve on the existing components of ROS while addressing its shortcomings. As such, ROS2 provides benefits such as real-time capability, support for multi-robot systems, and improved code stability. The main difference to ROS is the removal of the ROS master node. Instead of having a single master managing communications between all nodes, each node is responsible for its own communications and parameters. Executors are used to manage the scheduling of the individual nodes.

ROS2 also provides some settings for quality of service. This allows a trade-off between ensuring every message is received and getting frequent updates regardless of whether all messages were received. Finally, ROS2 also introduces a rolling release, allowing updates to be released more quickly.

4.2.2. Bit-Bots Codebase

This thesis uses software components from the Bit-Bots codebase[5]. This software has been specially developed for the *Wolfgang-OP* robot platform and provides all the necessary components for the robot to play a soccer game autonomously. The code is open source.

The main modules used in this thesis are the high-level motion generators, `bitbots_quintic_walk` and `bitbots_dynup`, as well as the low-level hardware controller, `bitbots_ros_control` [36][37].

The two motion generators both use a similar approach based on quintic splines for generating the motion patterns in Cartesian space. The motions can be modified by a small and intuitive parameter set, which can also be optimized automatically, for example, by using a Tree-structured Parzen Estimator. During execution, the error in the centre of mass is corrected using two PID controllers. Finally, an inverse kinematics solver transforms the motion into joint space.

While these two components can be used as is, the `bitbots_ros_control` module will be modified during the course of this thesis and thus will be discussed in more

[5] `https://github.com/bit-bots`

detail here.

On the highest level, the module consists of a ROS2 node. This node handles the communication between other nodes and the Wolfgang hardware interface. The Wolfgang hardware interface contains a dedicated hardware interface for each sensor. It also contains one interface per bus for all attached servo motors. One interface per bus is sufficient for the servos, as their communication is handled via sync read/write commands that address all servos on the bus simultaneously. Since sensors are usually addressed individually, the same thing is not possible for their interfaces.

Sensors are modelled as if they were Dynamixel motors and use the same protocol for communication. Therefore they can be directly added to the existing bus system. A custom control table is designed for each sensor type to transmit the sensor's information. To be able to read from or write to this table, a unique hardware interface needs to be developed for each sensor type as well. The hardware interface specifies which registers to read from or write to and how to process the received information. For sensors that return raw data that needs to be further processed to be human-readable, a converter class can also be provided. This is also useful if the sensor needs to be calibrated in order to provide useable data.

To control a motor, a message is sent to the `/DynamixelController/command` topic. The servo interfaces hold this message until the node's control loop triggers the writing of the interfaces. The command values are then directly written to the bus and executed by the servos.

When the control loop triggers the read action, the corresponding hardware interfaces send a read request to the bus and collect the response of the devices. This response is then published to a ROS topic. In the case of the servo motors, this topic would be `/joint_states`.

5.Approach

In this Chapter, the proposed hardware and software architecture is presented. In Section 5.1 the design of the SEA used in this thesis is explained. Section 5.2 describes the data collection process and gives an overview of the hardware and software setup. Section 5.3 showcase the network architectures used in this thesis.

5.1. SEA Development

The SEAs used in this thesis are inspired by the design developed by Martins et.al. [34]. The SEAs developed in their work are small and lightweight, which makes it possible to fit them into the knees of the *Wolfgang-OP*. Their design has also already proven to work with comparable humanoid robots. Based on their work, a 3D-printable version was developed by Bestmann et al. [25]. The compliant element of this SEA is made from NinjaFlex[1] a thermoplastic polyurethane (TPU) based 3D printing filament. That material was chosen as it has similar properties to the polyurethane used in the original work. The solid backplate of the SEA is made from polylactic acid (PLA) filament. The original SEA was designed to be symmetric. Martins et al. claim that this is necessary to react equally to torques in either direction. The 3D printable version used in this thesis does not possess that property. Instead, all beams in the SEA face the same direction. Nonetheless, an analysis of the relationship between torque and displacement (compare Figure 5.5) still shows a linear relationship in both directions. This indicates that this issue does not exist in the 3D-printable version. Furthermore, flipping the beams of the SEA allows the beams to be placed closer together and thus create stronger springs using the same space.

To find the optimal structure of the compliant element, a finite element analysis (FEA) was conducted. For this, a parametrised version of the compliant element was created. This version could be adjusted along two parameters: The number of beams between the inner and outer ring and the thickness of each beam. It should be noted that the number of beams also influences the angle between them, as each beam is equally spaced along the circle. The location of the first beam is always the same. This parametrisation leads to a total of 27 different configurations to be analysed, where every number of beams between 3 and 10, and the thicknesses 4, 6 and 8 mm per beam were analysed. Two of these configurations needed to be excluded from the analysis: The 4 mm, 9 beams configuration could not be generated, as the layout of the beams would create infinitely small vertices. The 8 mm, 10 beams configurations could be generated, but caused the beams to touch, which led to unrealistic properties in the analysis (compare figure 5.1).

The analysis was conducted in NASTRAN [38]. The material properties were taken from either the Inventor material library, the official data sheets or two papers that conducted their own experiments to determine the parameters [39][40]. In the case of

[1] https://ninjatek.com/ninjaflex/

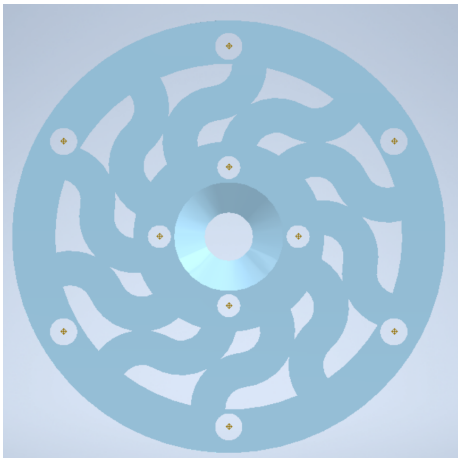


Figure 5.1.: CAD drawing of a SEA with 10 beams of 8 mm width. The beams touch each other, thus losing the desired spring properties.

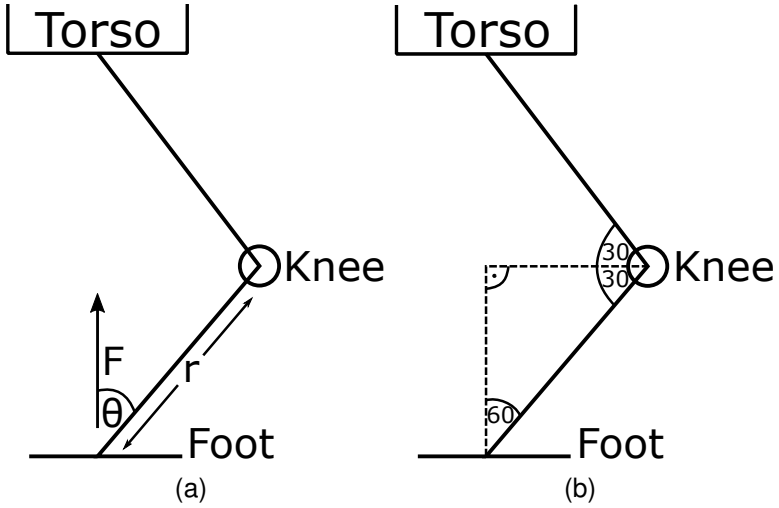


Figure 5.2.: Depiction of how forces act on the robot's knee. (a) shows the transformation from linear force to angular torque, where F is the force, θ is the angle at which the force is applied, and r is the length of the lever. (b) describes how θ is acquired using trigonometry.

NinjaFlex, values for generic TPU were assumed in those locations where the datasheet was missing specific parameters. The bearing was assumed to be made from soft yellow brass. A detailed list of the parameters used can be found in Appendix A.

There are two critical scenarios to consider for analysing the compliant element in this use case. Firstly, the SEA should absorb as much force as possible whenever the feet of the robot hit the ground. This is important, as it protects the gearbox and motor and reduces energy consumption (compare Section 3.1). Secondly, it should deform as little as possible when it is actuated by the motor so that the position of the robot's feet does not differ too much from the intended outcome.

To model the first case, the robot was instructed to walk forward, and the output of the robot's foot pressure cells was recorded. Unfortunately, one pressure cell on each foot was broken during this thesis. This means the full impact force could not be determined directly. Instead, the highest peak force in any cell was measured as $39.23\,N$. As each foot is equipped with four cells and most of the force acts on a single leg during a step, it can be assumed that the actual peak force is at most $4 \cdot 39.23\,N = 156.92\,N$. While this is not an exact estimate, it is a close approximation of the actual forces, which should be sufficient for this case. The general formula for converting a linear force to an angular torque is

$$\tau = r \cdot F \cdot \sin(\theta) \tag{5.1}$$

where r is the length of the lever, F is the linear force, and θ is the angle at which the force is applied (compare Figure 5.2.a). The length of the robot's lower leg is 20.38 cm and the feet of the robot hit the ground while the knee has an angle of roughly $60°$, which also results in the force being applied at an angle of $60°$ (Given by trigonometry, compare Figure 5.2.b). This means the torque applied to the backplate of the SEA equals

$$\tau = 0.2038\,m \cdot 156.92\,N \cdot \sin(60°) = 27.70\,Nm \tag{5.2}$$

For the second case, the stall torque of the XH540-W270 was applied to the bearing. The stall torque according to the data sheet is $11.7\,Nm$ at $14.8\,V$[2].

During the analysis, the peak reaction force at the point where the motor would be attached was measured for the first case, and the highest displacement on the bearing was measured for the second case. Additionally, the peak shear, tensile (1st principal) and compressive (3rd principal) stress were recorded in both cases. It should be noted here that only the rotational component of the force is considered in this analysis. As the foot contacts the ground at an angle, only part of the force results in a torque at the knee motor. The remainder acts as a linear force. As this cannot be simulated easily, it is disregarded here.

The results can be seen in Figure 5.3. As expected, the displacement decreases with the increase in thickness and beam count, following an exponential function for the beam count. It can also be seen that, in general, the reaction force increases with the amount and thickness of beams. However, the relation between beam count and peak

[2] https://emanual.robotis.com/docs/en/dxl/x/xh540-w270/

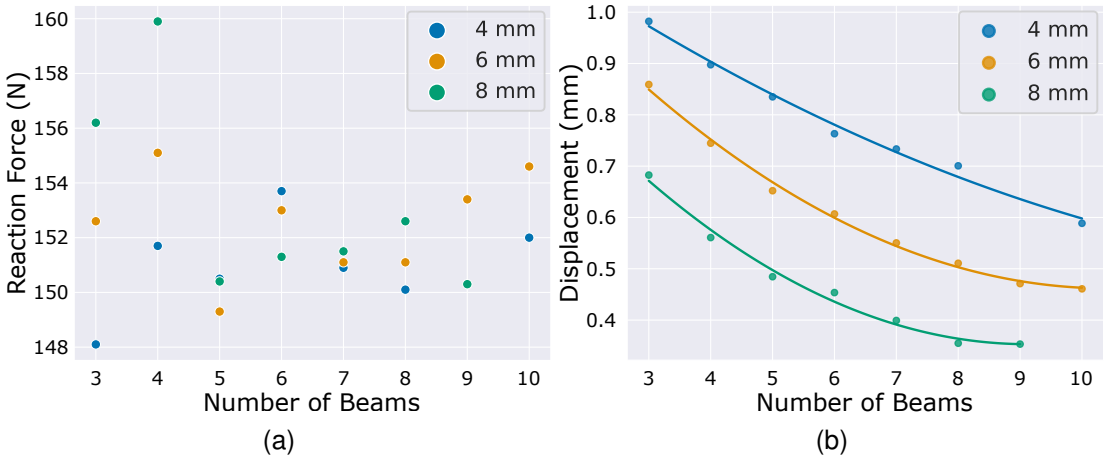


Figure 5.3.: Reaction force (a) and displacement (b) results of the different configurations. Colours represent different beam thicknesses. Regression lines indicate the trend along the beam count for each thickness.

reaction force is not as monotonous as expected, with the highest measured forces in the three and four beam configurations. This can be explained by the beam angles changing with the number of beams. Both configurations mentioned above cause angles in the structure that are close to equilateral triangles, which are considered one of the most rigid shapes. This structure adds additional stability to the compliant element, causing higher forces at the bearing. It should also be noted that even though the difference in maximum reaction force is relatively small in all cases, this number only describes the forces measured in a single point on the bearing. Thus, the actual force acting on the motor may be significantly larger, as the value depicted here is only an indicator of the forces changing instead of a measure of the total force. Finally, the analysis results also describe that the peak stresses on the compliant element itself (which is the part that perceives the highest stress in this setup) decrease with an increase in beam count or thickness. This is expected as larger amounts of material are present, resulting in a more even spread of the forces. Images of the analysis and the designed parts for the subcase of beams with a width of 6 mm can be found in Appendix A. The other two subcases have been omitted here for brevity, as they look similar to those shown.

After considering all the results, the 5 beams 6 mm configuration (see Figure 5.4.b) is most suited for this task, with a peak reaction force of $149.3\,N$ and a maximum displacement on the bearing of $0.6521\,mm$. The 3 beams 4 mm configuration (see Figure 5.4.a) performs slightly better when weighing both parameters equally. However, the enormous stresses measured in the component indicate that the part might break if applied to the robot. Furthermore, the measured displacement is enormous, which would cause additional problems.

However, while evaluating this component, it became clear that it would not be able to carry the weight of the robot reliably. Therefore, the third best configuration was chosen for this thesis with a reaction force of $150.3\,N$ and a maximum displacement

of $0.3532\,mm$. It consists of 9 beams with a width of 8mm. This is also the lowest displacement that could be achieved with this setup. The SEA used in this thesis and its analysis can be seen in figure 5.4.c.

It is also noteworthy that, even though a displacement of less than a millimetre seems unproblematic, the length of the leg needs to be considered, and a 0.3532 mm inaccuracy at the bearing results in

$$x = \frac{d}{r} \cdot l = \frac{0.3532}{14\,mm} \cdot 203.8\,mm = 5.14\,mm \tag{5.3}$$

difference at the foot, where d is the displacement at the bearing, r is the bearing radius, and l is the lever length. The SEA chosen in this Section thus causes a displacement of the foot of $5.14\,mm$. The actual displacement may be even more prominent, as the rest of the kinematic chain and gravitational force are not considered in this example.

Using a six-axis force/torque sensor, the spring constant of the SEA can be evaluated. By applying different forces to the SEA and measuring the corresponding displacement, the spring constant can be calculated as

$$k = \frac{\tau}{\theta} \tag{5.4}$$

where τ is the torque and θ is the displacement. Several different forces were applied to minimise measuring error. This results in a spring constant of $k = 9.106$.

When evaluating the torque-to-displacement relationship (see Figure 5.5), it can be seen that it closely approximates a linear function. A small dead-band area around the zero point can be observed, as a certain force is required to cause an initial displacement. As the displacement increases, some uncertainty is shown as the SEA moves and thus slightly alters the position of the magnet relative to the sensor.

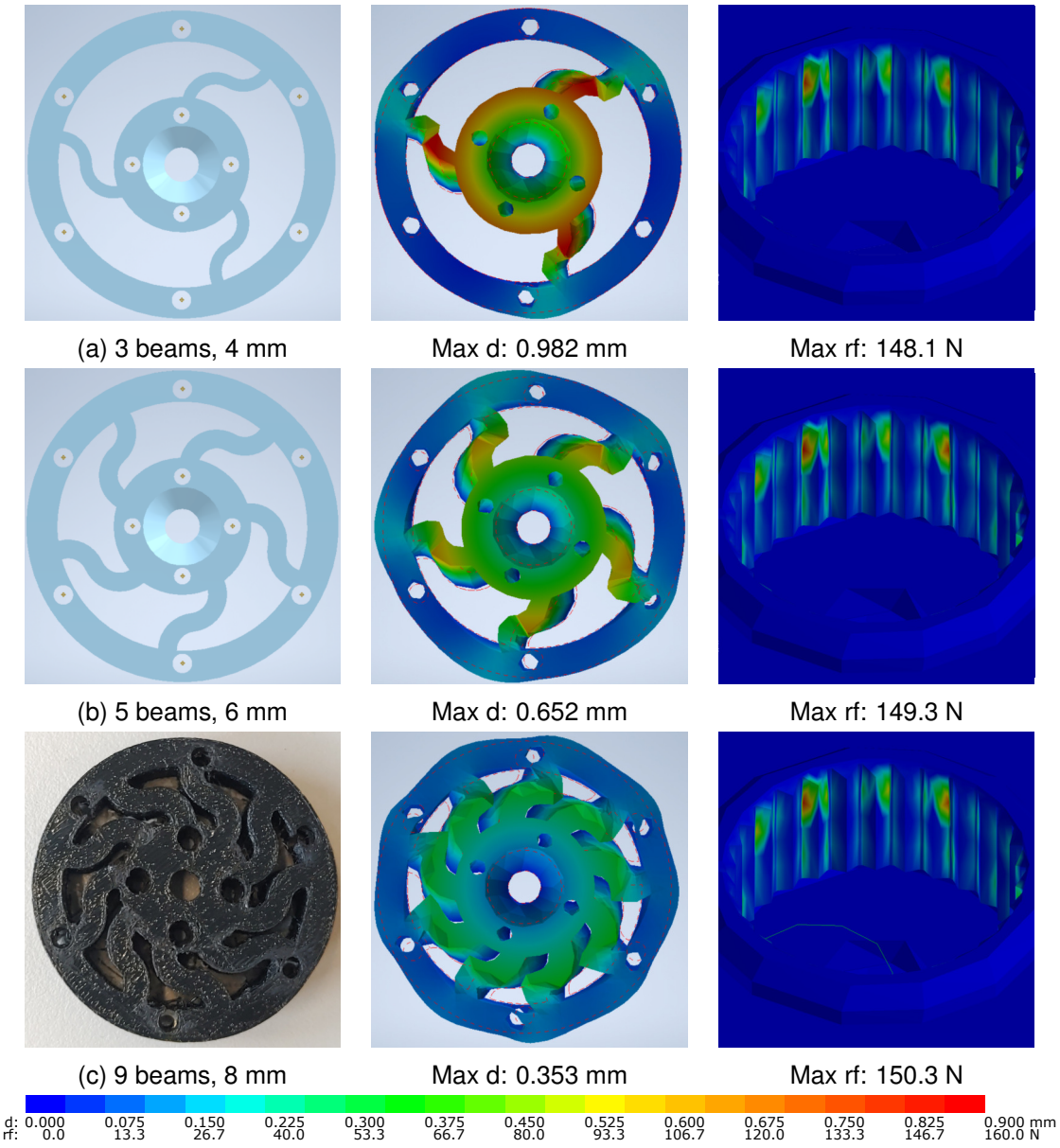


Figure 5.4.: Model, displacement and reaction force for three different SEA variations. (a) and (b) show versions that were insufficient for this use case, while (c) shows the SEA used in this thesis. Red dashed line in the displacement graph shows the original shape. Legend is shown at the bottom.

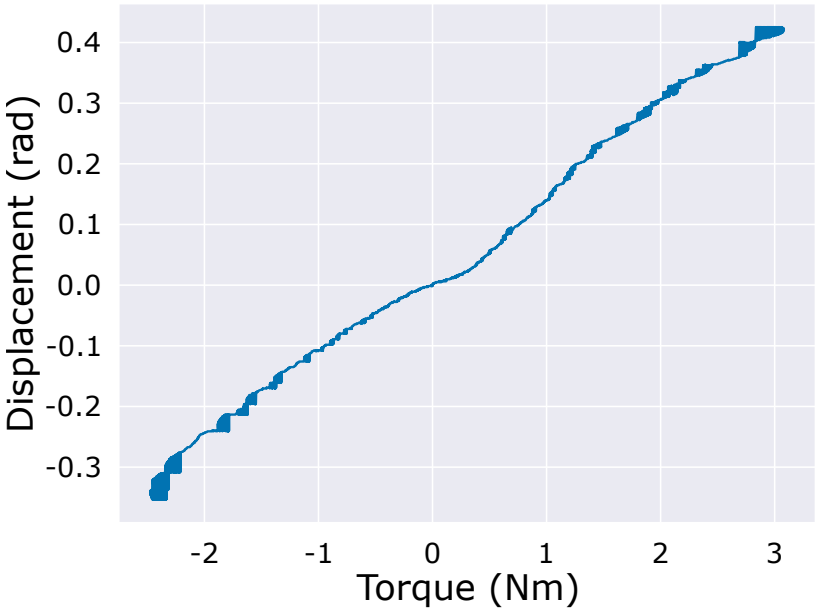


Figure 5.5.: Relationship between torque and displacement of the spring. Wider areas represent uncertainties in the readings.

5.2. Setup

This Section describes the hardware and software developed for this thesis. Subsection 5.2.1 presents the circuit board that was developed to record the position of the SEA using a Hall sensor. Subsection 5.2.2 shows the modified leg assembly. Subsection 5.2.3 showcases the software architecture.

5.2.1. Sensor Board

A custom sensor board was designed to measure the displacement between the target and the actual position. The board needed to read the raw data of the sensor, convert it into a message following the Dynamixel protocol[3] and send it to the robots motor bus. One primary constraint of the board design was the size. Since the board needed to be mounted to the robot's leg while the robot is running, it needed to be as small as possible so as not to obstruct the robot's movements. The final board design fits into 4x4 cm.

The board consists of four main components: The sensor, a processor, voltage regulation and a connection to the Dynamixel bus (compare Appendix B). As a sensor, the AS5600 was chosen. This sensor has the advantage that necessary conversions are already done on board (compare Section 3.3). This means it is not necessary to convert the sine waves of multiple Hall sensors into a position, but it instead instantly returns an encoded angle. Assuming an equal distance between each encoding, this value can

[3] https://emanual.robotis.com/docs/en/dxl/protocol2/

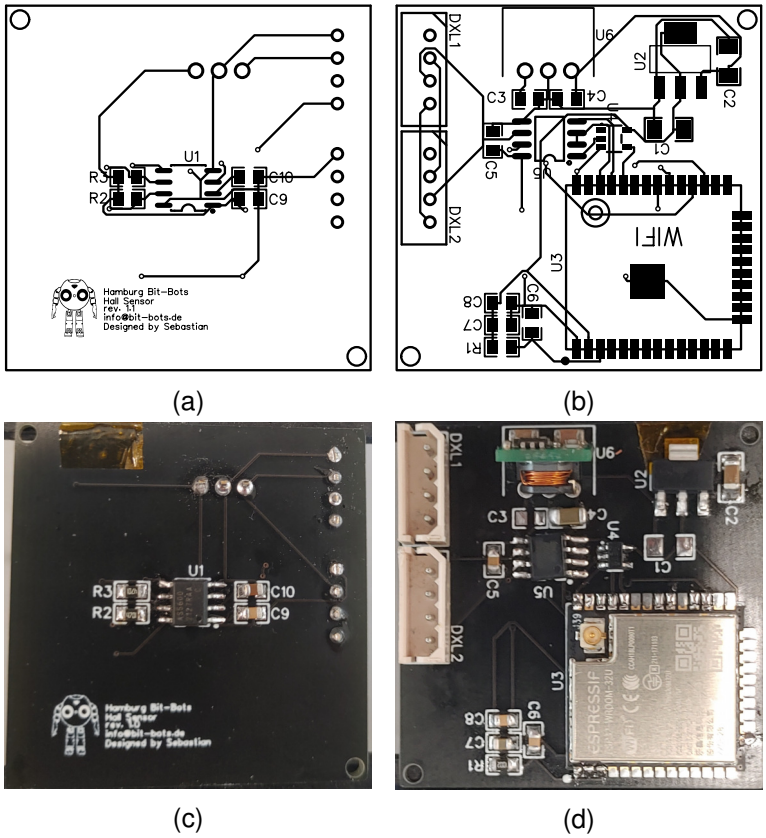


Figure 5.6.: Schematic ((a) and (b)), and real-life example ((c) and (d)) of the sensor board. The sensor is mounted on the front side ((a) and (c)) alone, while all logic is mounted on the back ((b) and (d)), so that the sensor can be as close to the magnet as possible.

easily be converted into radians. Furthermore, the sensor used here is very similar to the sensor used in the Dynamixel motors and produces the same output shape, making both readings easily comparable.

Communication between the sensor and the microcontroller is conducted via pulse width modulation (PWM). In PWM, the phase length of a square wave signal is modified to represent the sensor readings. The wave consists of a fixed length high signal, a varying length high or low signal, depending on the reading, and concludes with a fixed length low reading. The total length of each signal is always the same. The transmitted value can be determined by measuring the length of the high signal.

As a processor, the ESP32 was used. The processor is small, fast, and runs on low voltage and current. Since the processor has two cores, one core can be used to communicate with the sensor, while the other handles communication with the bus. This makes reading the sensor faster, as no scheduling is required. Pre-existing libraries and extensive documentation made the use of this processor simple. There has also already been prior experience with this processor in our work group, supporting the decision to

use this specific model. To allow the processor to communicate with the Dynamixel bus, a Low-power RS-485 transceiver is used to convert the UART signal of the processor into the RS-485 required by the bus and vice versa. Behind it, a level shifter converts the 5V signals on the bus into 3.3V signals required by the ESP32.

For voltage regulation, the power input of the Dynamixel bus (15.4 V) needed to be converted to two different voltage levels. First, the voltage is converted to 5V using a buck converter, and from that level, converted to 3.3V using a low dropout converter.

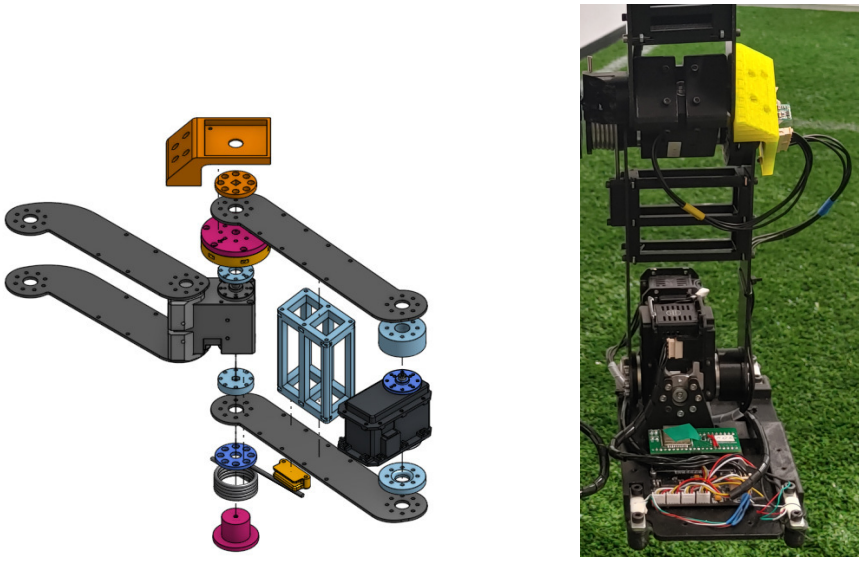
Finally, the board is connected to the bus with two connectors. The finished board can be seen in Figure 5.6.

5.2.2. CAD Model

The layout of the robot's legs needed to be expanded in a way that allowed the mounting of the SEA and the PEA simultaneously without altering the kinematic chain of the leg. For this, the PEA needed to be moved to the inside of the leg, as the SEA needed to be in series with the motor and thus on the side with the motor horn, i.e. the outside. The motor could also be rotated to have the motor horn facing inside, but this was not desirable, as it would no longer be comparable to the previous setup. Due to the constraints of the thesis, developing smaller PEAs was not possible. Therefore only very little room is available between the legs. Fortunately, the legs are still not touching in all common motions. However, if the robot falls over, this could severely damage the legs.

Several spacers needed to be added to the model to fit all parts together. Since the bearing of the XH540-W270 does not have a screw hole in the middle, compared to its predecessor MX106, an additional part needed to be added there as well in order to mount the PEA. While adding the spacers, it was taken care that the relative positions of the knee and ankle pitch motor stayed the same.

Finally, a mount for the Hall sensor and the magnet were added above the SEA. The complete assembly of the leg is shown in Figure 5.7.



(a) (b)

Figure 5.7.: Full assembly of a single leg. Assembly is shown in (a), and real-life implementation in (b). The SEA is mounted at the top, while the PEA is mounted at the bottom (compare fig. 3.1 and fig. 3.2. In order to fit all components together, spacers (**light blue**) were added. A mount for the Hall sensor and the magnet were also added on top of the SEA (**orange**).

5.2.3. Software

For data collection, some additional software components needed to be developed. In total, five new components are introduced. An overview of the software architecture can be seen in Figure 5.8.

On the lowest level, software for the microcontroller of the Hall sensor board was developed. This software is tasked with reading the measurements of the Hall sensor and transmitting them onto the Dynamixel bus. To ensure fast processing, both cores of the ESP32 are utilised, each running a separate thread. While one thread is only occupied with reading data from the sensor via PWM, the other thread solely handles the communication with the bus. Here, a modified version of the Dynamixel2Arduino library [4] is used to make the board act as if it were a Dynamixel servo. Apart from some default items, such as ID or baud rate, a custom control table can be defined. That way, the sensor readings can be transmitted effortlessly.

The counterpart to this microcontroller script is the ROS control Hall sensor interface. When required, this component reads from the sensor board and transforms it into a ROS message. This works by simply sending a read request for specific registers of a particular ID. Since the control table of the sensor differs from the ones of the servo motors, the sensors also need to be excluded from any sync read calls in the Dynamixel

[4] https://github.com/bit-bots/bit_foot/tree/master/firmware/esp32_bit_foot

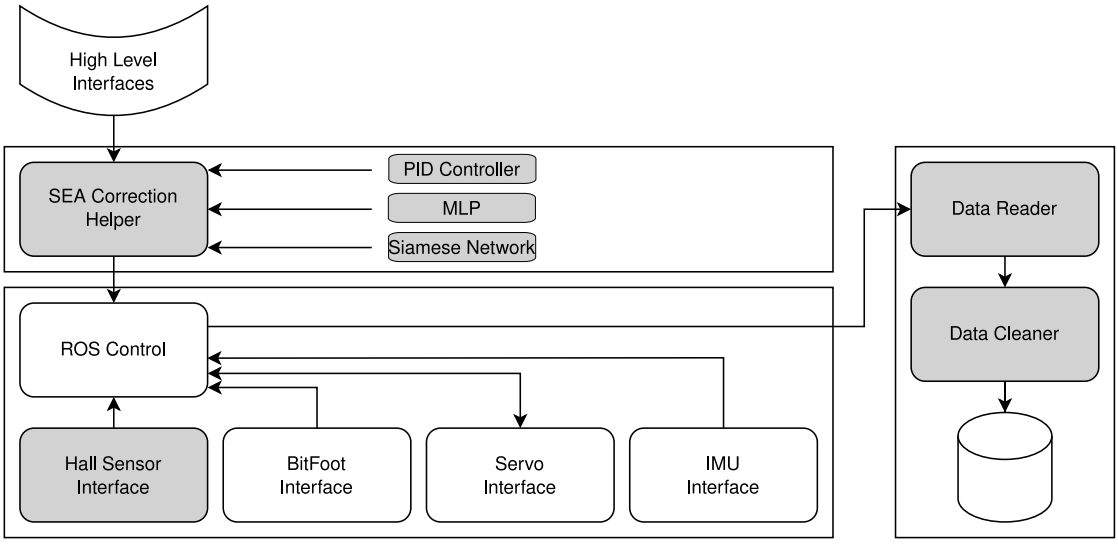


Figure 5.8.: Software architecture of the system proposed in this thesis. Components are grouped according to their respective ROS packages. Arrows represent the flow of information between the components. Grey boxes are components that have been newly introduced in this thesis.

workbench library. As data is returned as consecutive bytes, it needs to be transformed into a float before being further processed. This component also checks whether new data is received from the sensor at all and emits a warning if the sensor does not appear to be working correctly. The Hall sensor interface is integrated into ROS control as a hardware interface, as described in Section 4.2.2.

The third software component is injected between the high-level interfaces and the ROS control node. It allows modifying the motor goals before sending them to the servos. Instead of directly sending commands to ROS control, the commands are intercepted by the SEA correction helper and saved. The servo goals will now no longer be sent when a command is received but instead whenever the servos return their current state. This means new goals are sent continuously, with the same frequency as the states can be read, instead of only sending goals when the high-level motion generators send a new goal. This is necessary, as the effects of the SEA need to be corrected continuously. The way ROS control would behave normally does not change, though, since repeatedly sending the same goal has the same effect as sending a goal once. The way the error is corrected can be chosen freely. By default, the module relays the commands without making any modifications. Other correction methods, such as PID controllers or neural networks, can be easily integrated into the node.

The final two components are directly connected to the recording of the data. The data reader component subscribes to all relevant topics and writes the essential parts of the messages into a CSV file. Namely, the reader logs position, velocity and effort of the motor, position and velocity of the Hall sensor, readings of the IMU and foot pressure sensors, and a timestamp. If a command is being sent, it is also logged. A

time-synchronised subscriber is used to ensure that all data points in a single sample are related to each other. This subscriber only executes the callback if all messages are received within a given time frame. The only topic excluded from this synchronised subscriber is the joint command message, as it is not granted that commands are being sent. The command is therefore stored independently and added to the sample if it is not older than the sample by a threshold. This method can collect samples at a frequency of 100 Hz per motor.

Finally, the collected dataset is passed through a clean-up script. This script removes rows from the dataset where the measurements of all foot pressure sensors are below a threshold. These rows indicate that the robot would have been picked up at that point, and therefore the samples do not accurately represent the working conditions of the robot. The clean-up script also converts the orientation quaternion of the IMU to Euler angles. It also normalises the data and changes some formatting.

5.3. Network Architectures

The proposed architecture uses neural networks to control the positional error introduced by the SEA. Two different network architectures have been applied and evaluated in this thesis. Section 5.3.1 describes the multi-layer perceptron that has been used. Section 5.3.2 displays a Siamese network architecture which can provide additional stability.

5.3.1. Multi-Layer Perceptron

Inspired by Hwangbo et al. [15], who used a small MLP to solve a similar problem, this first approach uses a small and fast-to-train network. The difference, however, is that their network is tasked to add an error to simulation data, while this network will be tasked to remove the error from real-world data. Since each sample only consists of a few values, and the output is one-dimensional, no large network is required to learn the relation. The network chosen here is a MLP with three hidden layers of 64 neurons each. The leaky ReLU function was chosen as the activation function, as it is fast to

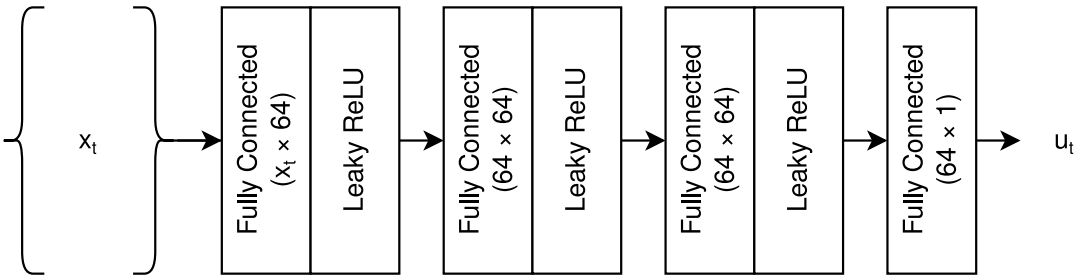


Figure 5.9.: Schematic representation of the MLP network used in this thesis. The size of the input vector x_t varies depending on the modalities used. u_t is the action proposed by the network.

compute and prevents saturation, thus leading to faster convergence. Leaky ReLU was chosen over standard ReLU, as it prevents the neurons from dying and turning into a constant. The architecture can be seen in Figure 5.9. As a loss function, the mean squared error was used.

5.3.2. Siamese Network

Initially proposed by Bromley et al. [41], Siamese neural networks are usually used in image processing, especially for detecting similarities between two images. However, the architecture can also be modified to suit this task.

The network works by simultaneously solving two opposing tasks. Given a state x_t and an action u_t, it predicts the following state x_{t+1}. But given x_t and x_{t+1}, it predicts the action u_t that was required to reach the second state. To connect the two tasks into one network, the states x_t and x_{t+1} are first converted into latent vectors using three fully connected layers of 64 neurons each. Unlike a regular network, however, those fully connected layers share the same weights for both tasks, effectively making them the same layer. This means that during training, the network learns weights that can optimally solve both tasks and can use the knowledge of the other task without overfitting on it. Implicitly the network learns the underlying physical properties of the SEA instead of just learning a one-directional relationship. This makes the network more robust. Two separate fully connected layers convert the latent vectors into the desired outputs. ReLU functions were used for activation.

The loss function for this problem consists of two components:

$$L = MSE(u_t, \hat{u}_t, W) + \lambda MSE(x_{t+1}, \hat{x}_{t+1}, W) \tag{5.5}$$

where MSE is the mean squared error function, and W are the shared weights. λ is used to reduce the impact of the supporting task. For this use case, the supporting task is only used to guide the network in the right direction, which is why a low $\lambda = 0.1$ was chosen.

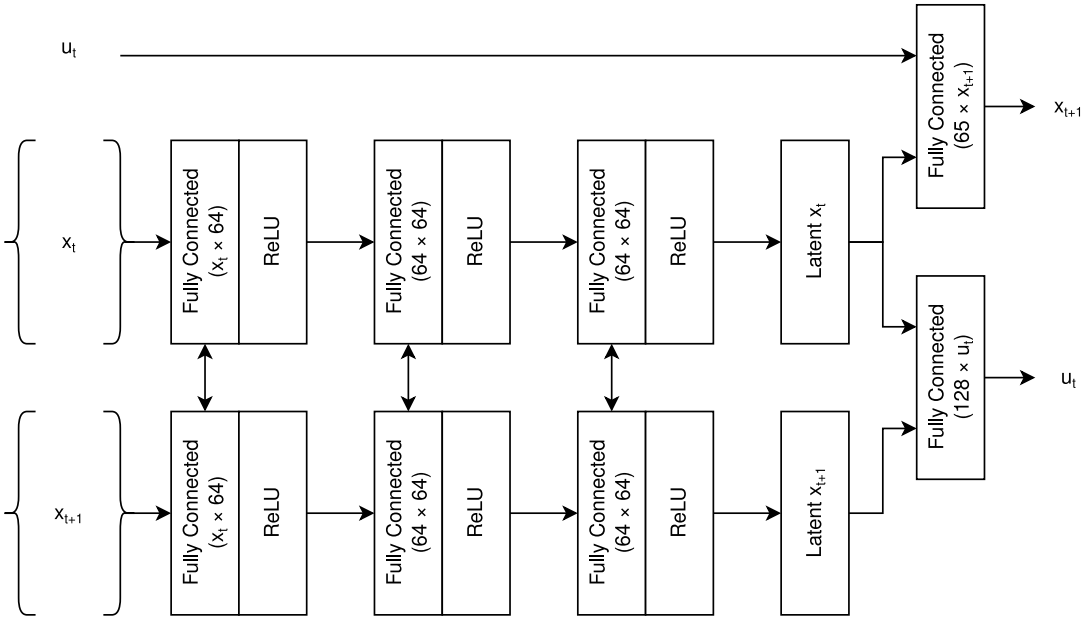


Figure 5.10.: Schematic representation of the Siamese network used in this thesis. x_t is the vector of the current time step, x_{t+1} is the vector of the next step, and u_t is the action required to transition between the states. The weights of the fully connected layers are shared according to the dashed arrows.

6.Evaluation

This Chapter describes the experiments that have been conducted to evaluate the use of SEAs in the knee motors. It is challenging to simulate torsion springs realistically. As the component's properties depend heavily on the deformation and the acting forces, it cannot be easily approximated by a simple function. Modelling a realistic torsion spring in a physics simulation that simulates an entire robot requires a lot of computing power and would be very slow. Approximations of the spring would be faster but lose a lot of accuracy and make sim-to-real transfer difficult. Neither Webots[1], nor PyBullet[2], two of the most well known physics simulators for robotics, natively support torsion springs.

Therefore, it was decided to conduct all experiments of this thesis either directly on the real robot or on a recording of real-world data. First, the error correcting networks are evaluated on the simplified case of the robot being suspended in the air in Section 6.1. After that, the problem is transferred to a robot walking on the ground in Section 6.2. In Section 6.3 possible disturbances of the setup are evaluated. Section 6.4 explores the use of Hall sensors and SEAs as torque sensors. Finally in Section 6.5 the energy conservation properties of SEAs and PEAs are evaluated.

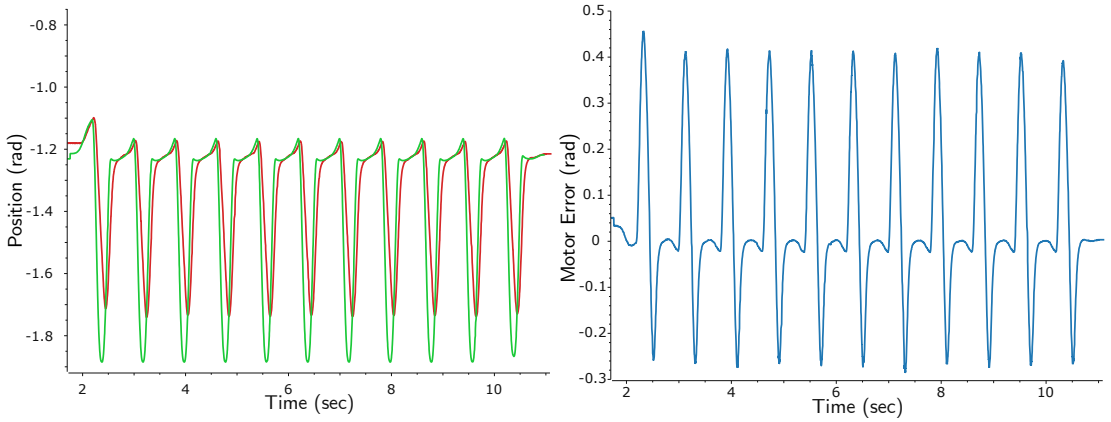
6.1. Free Hanging Robot

As a proof of concept, the approaches suggested in this work have first been evaluated on a robot that is suspended in the air. This eliminates the need for stability and allows experiments to be conducted even if the robot were to fall over under normal circumstances. However, this also reduces the modalities available to this task, as neither foot pressure sensors nor IMU can provide usable data in this state.

Four approaches have been evaluated against a baseline of not taking any corrective actions. All networks in this Section have been trained with a learning rate of $1e-6$ and a batch size of 32. These values have been determined empirically to deliver the best results. The dataset used for training consists of approximately 40,000 samples. It has been recorded within 6 minutes of the robot walking in various directions using the Quintic Walk algorithm [36]. The dataset was sampled with an average of approximately 60 Hz per knee. For each training, the dataset was shuffled and randomly split into an 80/20 train and validation set. Training was conducted for a maximum of 2500 epochs but stopped early if no improvement was reached within the last 100 epochs. Training took around 4 seconds per epoch for Siamese net based approaches and 3 seconds per epoch for MLP based approaches on an Nvidia RTX 2080. This equates to 190 and 300 steps per second, respectively.

Evaluation was conducted by walking in varying directions with varying speeds for 10 seconds each. Available directions were forward, backward, right, turn right, forward and right, and backward and right. Speeds were chosen from the

[1] https://cyberbotics.com/
[2] https://pybullet.org



(a) (b)

Figure 6.1.: Difference between motor position and command (a), and resulting error (b) of the right knee motor while walking forward with 0.1 m/s, without using a SEA. The **green** line displays the target position sent to the motor, while the **red** line shows the position the actuator reached. Figure (b) shows the difference between the two.

set $\mathcal{S} = \{0.05,\ 0.1,\ 0.2\}$, with the unit m/s for translations and rad/s for rotations. Since the robot is symmetric, motions to the left were omitted for brevity, expecting similar results. Speeds for moving sidewards and turning were scaled with the factor 0.5 compared to speeds moving forward, as the robot is not capable of taking quite as large steps in these directions.

The walking engine of the Hamburg Bit-Bots uses unrealistically large motor goals in order to achieve stronger torques and faster movements (compare Figure 6.1). This error is present, regardless of whether a SEA is used or not. This means that a large portion of the observed error cannot be corrected in the first place. On average, this error is 0.103 radians. Since the error varies with speed and direction, an average has been taken for every evaluated combination, and the results have been corrected by this value.

Without taking any corrective action, the baseline approach scored an average absolute error over all directions of 0.091 rad. The highest average was measured on the right knee when walking to the back and the right simultaneously at a speed of 0.2, as 0.136 rad. This is expected, as this is the motion that requires the highest total speed, moving 0.2 m/s backwards and 0.1 m/s to the right. The lowest average was 0.055 rad when moving forward at 0.05 m/s.

As a first approach to reducing the introduced error of the SEA, a simple PI controller was applied. The controller corrected the error between the reading of the Hall sensor and the target position sent in the `JointCommand` message. The gains of $k_p = 0.2, k_i = 0.1$ were determined empirically, as higher gains caused oscillation in the system. The PI controller performed slightly worse, with an average error of 0.098 rad. This happens because these controllers tend to overshoot their goal and thus add

Approach	Average	Max Direction	Max Avg	Min Direction	Min Avg
Baseline	0.0914	back right (0.2)	0.1360	front (0.05)	0.0551
PI Controller	0.0978	back right (0.2)	0.1399	front (0.05)	0.0720
MLP	0.0907	front (0.2)	0.1291	front(0.2)	0.0585
Positive MLP	0.0740	back right (0.2)	**0.1074**	rotate right (0.1)	**0.0465**
Positive Siam	**0.07361**	back right (0.2)	0.1178	right (0.2)	0.0488

Table 6.1.: Average error over all directions, as well as the average of the best and worst direction, for each of the five approaches. Errors have been corrected for the difference between command and motor position. Units of the errors are radians. Speeds of the best and worst trials are given in m/s in parenthesis.

some error back into the system. A better result could likely be achieved here with more fine-tuning.

The second and third approaches use a multi-layer perceptron network, as described in Section 5.3.1. The former was trained on the full normalised dataset as described above and slightly improved over the baseline. It is notable that this approach was able to strongly reduce the errors on one leg, whereas it struggled to correct the error on the other. This can also be seen in Table 6.1, where both the highest and lowest average appear in the same direction and speed. Due to this fact, a second MLP was trained on a variation of the dataset. In this version of the dataset, all samples of the right leg have been inverted. As both SEAs are identical and the robot is symmetric, both legs are expected to behave similarly. Thus, the samples recorded from the right leg can be inverted to look like samples recorded from the left leg. In doing so, the network only needs to learn positive outputs, making solving the problem more straightforward. The network learns a general representation of a SEA instead of learning to distinguish between the left and right leg. The predictions of this network need to be inverted in order to be applied to the right knee. It should be noted that input values could still be negative if they stem from one of the sensors. This is necessary, as the direction of the sensor readings provides valuable information to the network. In all future experiments, this type of dataset will be referred to as the positive dataset. Networks trained on this dataset are referred to as positive networks.

This network was allowed to train for 3,000 epochs, with a convergence reached after 2,683 epochs. This greatly improved the performance of the network, as the range of possible values is smaller.

Finally, a Siamese network, as described in Section 5.3.2, was trained, also using the positive dataset. This approach performed similarly to the positive MLP, with a negligibly small improvement. However, the motions generated by the Siamese network look significantly more realistic. This is also visible in the data, as the Siamese network performs significantly better at reducing the peak errors (compare Table 6.2).

It is also noteworthy that the default MLP converged significantly faster than the other approaches during training. Training of the MLP took only 1214 epochs, whereas both other approaches exceeded the limit of 2500 epochs.

In the next step, the networks were evaluated on an evaluation dataset. The dataset

Approach	Peak Error	Corrected Peak Error
Baseline	0.9983	0.5316
PI Controller	1.0065	0.5566
MLP	0.9513	0.5321
Positive MLP	0.8865	0.4743
Positive Siam	**0.8146**	**0.3520**

Table 6.2.: Peak errors with and without correcting for motor command error. All units in radians.

consists of around 13,000 samples of the robot walking in random directions while still suspended in the air. The dataset was captured after the leg was disassembled, reassembled, and recalibrated to ensure robustness to hardware changes. Both mean squared error (MSE) loss and mean absolute percentage error (MAPE) were captured. The results are shown in Figure 6.2.

It can again be seen that all three networks show similar promising results, with the positive MLP performing best. While the average loss of the Siamese net is worse than that of the positive MLP and only slightly lower than that of the default MLP, the results show that the Siamese net is better at reducing peak values. The losses of the Siamese net are more consistent, with both the first quartile and the entire distribution spanning a smaller area compared to the other approaches. Since removing peaks in error are highly beneficial to the stability of the system, while a small overall error can be tolerated, this gives a massive advantage to the Siamese net. The MAPE was 4.1% for the default MLP, 4.0% for the Siamese net and 3.9% for the positive MLP.

It should be noted, however, that in this experiment, the network's output does not influence the following sample. As the feedback loop has been opened, dynamic effects, such as output oscillations, cannot be evaluated. Thus, networks may perform better in this experiment than in reality.

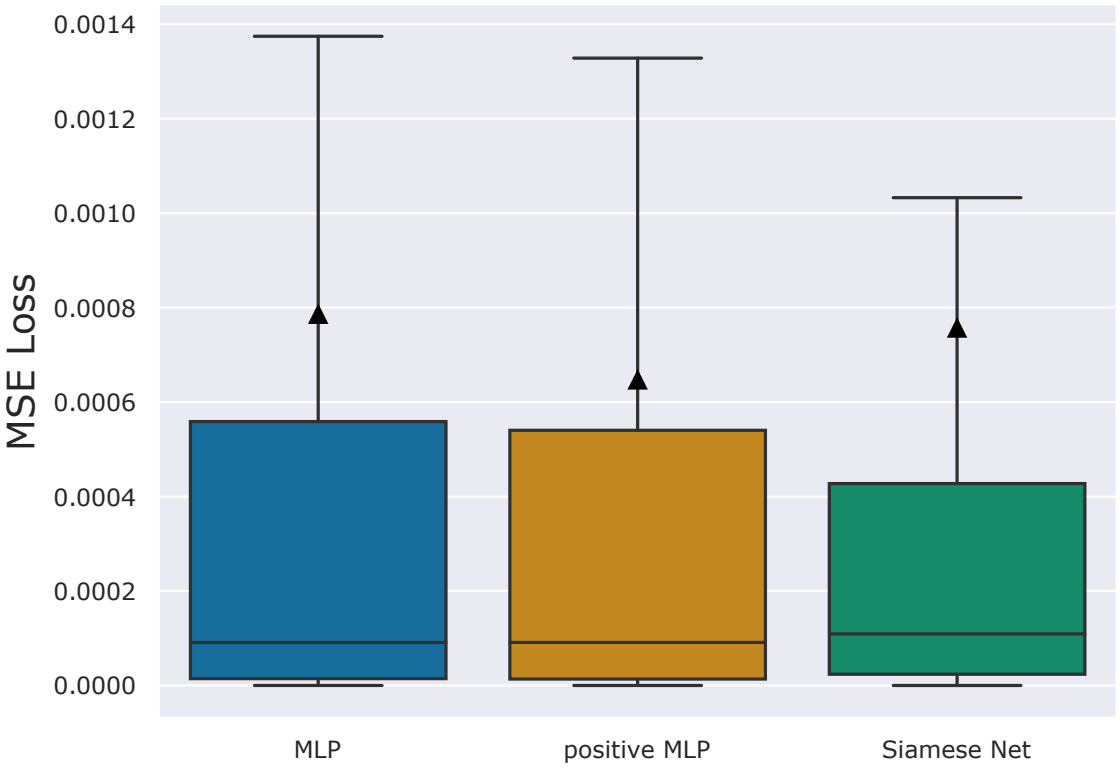


Figure 6.2.: Box plot of losses measured on the evaluation dataset for the different networks. Black line displays the median, black triangle displays the mean. The first quartile is contained within the box, distribution excluding outliers is contained within whiskers. Outliers have been hidden.

6.2. Walking on Ground

In the next step, the problem was transferred to the robot walking on the ground. In Section 6.2.1, it was evaluated which modalities are most performant in solving the problem by comparing the network outputs to the ground truth in a prerecorded dataset. Section 6.2.2 uses the same datasets to evaluate the performance of the different network architectures. Finally, Section 6.2.3 evaluates the performance of the best candidate while walking around with the real robot.

6.2.1. Ablation Study

As the robot is placed on the ground, the IMU and the foot pressure sensors become available to the networks. Therefore, it was evaluated which information should be provided to the network to achieve optimal results. The experiment was conducted by training a MLP on different combinations of the available modalities. Training was conducted on a dataset consisting of around 22,000 samples of the robot walking on the ground in various directions while equipped with SEAs. Evaluation was then conducted on a separately recorded dataset of approximately 10,000 samples. The evaluation dataset already contains the ground truth solutions for each sample. Therefore, it was used to quantify how well each candidate could replicate the target outputs. To prove the versatility of the approach, the evaluation dataset was recorded on a different turf and after recalibrating the sensors. Training was conducted for a maximum of 10,000 epochs with early termination after 100 epochs without improvement. However, none of the candidates used the entire training duration. The slowest candidate took 7,561 epochs to reach its minimum, and this was the case where only the information of the foot pressure sensors, along with the target position, was available to the network. Training used a learning rate of $1e-6$ and a batch size of 32. Training speed was similar to the free-hanging robot experiments; however, due to the smaller dataset, training here only took one second per epoch on an Nvidia RTX 2080. The number of inputs did not measurably influence the time per epoch but influenced the number of epochs needed, and thus the total time.

Overall, eleven different combinations were evaluated, choosing one or multiple of the following modalities:

- Motor Position
- Hall Sensor Position
- Motor Velocity
- Hall Sensor Velocity
- Motor Effort
- IMU
- Foot Pressure Sensors

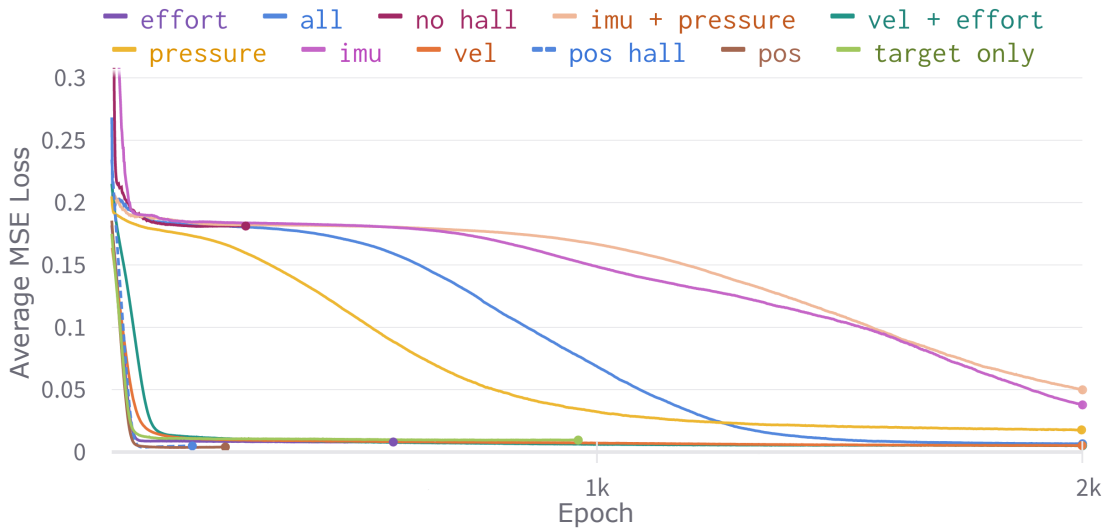


Figure 6.3.: Average mean square error of the validation step during the first 2,000 epochs of training. Part of the legend has been shortened to fit everything. Velocity was shortened to vel, motor position to pos, and Hall position to hall.

The effort, in this case, is not measured directly but instead estimated from the current flowing through the actuator. All candidates always also receive the target hall sensor position as additional input. The different combinations, as well as the loss during the first 2,000 epochs of training, are shown in Figure 6.3

The candidates were evaluated on the evaluation dataset on the basis of MSE and MAPE. The results are shown in Table 6.3, and in Figure 6.4.

These results give a clear indication of the importance of the different modalities. It can be seen that both the IMU and the foot pressure sensors produce very noisy data that makes it challenging to learn connections between the sensor data and the actuator movements. This is indicated by the very long training times and the high MSE and MAPE. This becomes especially apparent in the case of having both sensors available. Here the MSE distribution is spread across a vast range, which shows the noisiness of the data.

It can also be seen that those trials with either the motor or the Hall sensor position available generally perform better than those without. This is expected, as the next target position is usually very close to the current position of the Hall sensor and moderately close to the current position of the motor. This becomes especially clear when looking at the last three cases. The IMU + pressure sensor case, consisting of 15 of 18 available sensor-related inputs, performs poorly with a lot of noise that cannot be adequately recognised. The results significantly improve by adding the current motor position. as well as two of the three remaining sensor inputs. Finally, by adding the

	MSE	MAPE	Epochs
Target Only	0.1864	0.0087	860
Motor Position	0.1217	0.0041	133
Hall + Motor Position	0.1375	0.0050	**65**
Velocity	0.1460	0.0059	2060
Effort	0.1875	0.0086	479
IMU	0.1427	0.0065	6847
Pressure	0.1433	0.0061	7561
Velocity + Effort	0.0960	0.0034	5517
IMU + Pressure	0.2273	0.01136	3821
No Hall Position	0.1478	0.0058	1971
All	**0.0541**	**0.0011**	5146

Table 6.3.: Average MSE, MAPE and Training duration of each tested combination of modalities. Rounded to 4 digits.

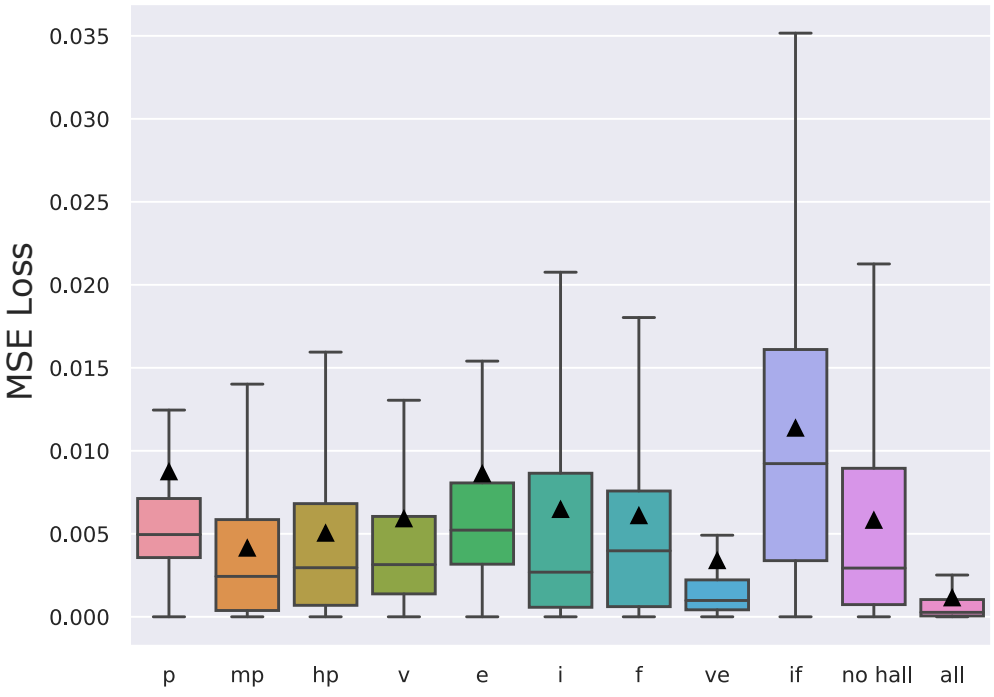


Figure 6.4.: Box plot of losses measured on the evaluation dataset for the different modalities. Black line displays the median, black triangle displays the mean. The first quartile is contained within the box, distribution excluding outliers is contained within whiskers. Outliers have been hidden.

position of the Hall sensor and the resulting velocity, the trial produces the best results out of all trials. Ultimately, however, it would be beneficial to end up with a solution that does not require the Hall sensors to be attached to the robot. These sensors are fragile and mounted in positions the robot is reasonably likely to fall onto, thus destroying them. Therefore, a combination of velocity and effort was chosen for the following experiments, as these inputs seemed to perform very well.

Both effort and velocity are abstractions of the movement of the actuator. They also reflect the deformations of the SEAs well, as higher efforts would deform the SEA more, and changes in velocity would also lead to the SEA flexing in either direction to compensate for the change. The velocity information also helps to grasp the whole state of the actuator, as it indicates how fast and in which direction it is moving. As the network only ever sees one sample at a time, it can not determine in which direction the actuator is moving without knowing the velocity. Thus, these inputs give much information on the problem and can support the network in predicting the required actions with high precision.

Examples of each of the candidates' predictions for the left leg can be found in Appendix C. When visually inspecting the results, it can be seen that candidates with only very little information available tend to learn an average value. The average is a local minimum that never causes a very high loss in a sample, but is not usable, as it also never produces a very low loss in a sample. It can also be seen that especially the velocity component is necessary to learn the short but high peaks of the walking motion. The velocity also adds a significant error at the peak of the motion, as the velocity inverts at this point to move the actuator backwards. This happens because these networks need to learn the difference between the two legs. As a negative velocity is more likely in the right leg, which is mounted inverted, the predictions for the left leg briefly jump to a negative position. This effect is no longer observable in networks, which only use positive inputs.

The visual inspection also suggests that the combination of velocity and effort might perform better than using all available inputs, as it tracks the peaks more closely. As the peaks constitute only a tiny part of the motion, missing them has a relatively small impact on the MSE and MAPE. Nevertheless, they are the most critical part of the motion. The peaks represent the robot lifting its leg, whereas the valleys equate to the knee movement while the leg is on the ground. Errors that appear while the leg is on the ground are problematic, as they can lead to the robot falling over. However, not reaching the motion peaks is a bigger problem, as it means that the robot does not raise its legs properly and thus will not make any progress while likely also knocking itself over. Furthermore, a large portion of the error in the velocity and effort combination can be removed by training the network only on positive data and then inverting the predictions for the other leg.

6.2.2. Evaluating Network Architectures

After the best set of modalities had been established, the different network architectures were evaluated. Both the MLP and the Siamese network were evaluated in their stan-

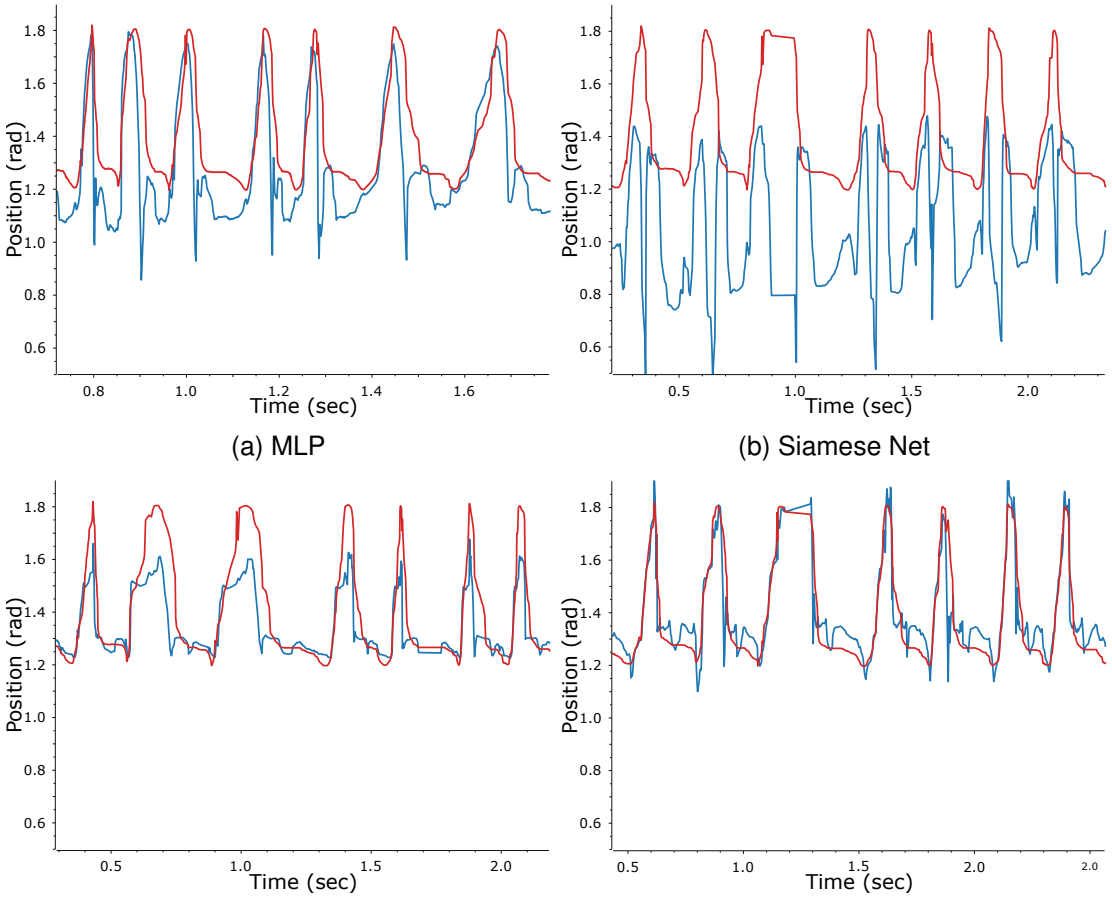
	MSE	MAPE	Epochs
MLP	0.096	0.0034	5517
Positive MLP	**0.0564**	0.0016	**2110**
Siamese Net	0.1262	0.0068	9985
Positive Siamese Net	0.0575	**0.0011**	8974

Table 6.4.: Average MSE, MAPE, and training duration of each tested network. Rounded to 4 digits.

dard and positive form. All networks were trained and evaluated according to the same conditions and on the same datasets as described in the previous section. The MLP took around one second per epoch, while the Siamese net took around two seconds per epoch on a Nvidia RTX 2080.

The evaluation shows that while both Siamese net approaches take significantly longer to train, they can produce better results. It also shows that the approaches trained only on a positive dataset perform better in all aspects than those trained on the unmodified dataset. The best-performing network is the positive Siamese net, while the worst-performing network is the standard Siamese net. As shown in Figure 6.5, the standard Siamese net also learns the shape of the motion correctly but has an offset towards the zero position to accommodate both legs. It should be noted that the standard Siamese net is the only network that would have exceeded the limit of 10,000 epochs without reaching a local minimum. More extended training might have resulted in a better result here, but this was not feasible in the scope of this thesis, nor would it be worthwhile in practical application, as 10,000 epochs already take a substantial amount of time.

The positive Siamese net, however, is capable of tracking the target motion precisely. It matches the valleys closely while also keeping the peaks intact. In doing so, it achieves the lowest average MSE, and an only negligibly worse MAPE, compared to the positive MLP, which performed second best (compare Table 6.4).



(a) MLP

(b) Siamese Net

(c) Positive MLP

(d) Positive Siamese Net

Figure 6.5.: Comparison between network architectures when predicting goals for the left knee. The same seven steps of the evaluation dataset are visualised for all candidates. **Red** represents target, **blue** represents network prediction. X-axis represents experiment time and is not related to real-life speed of the robot.

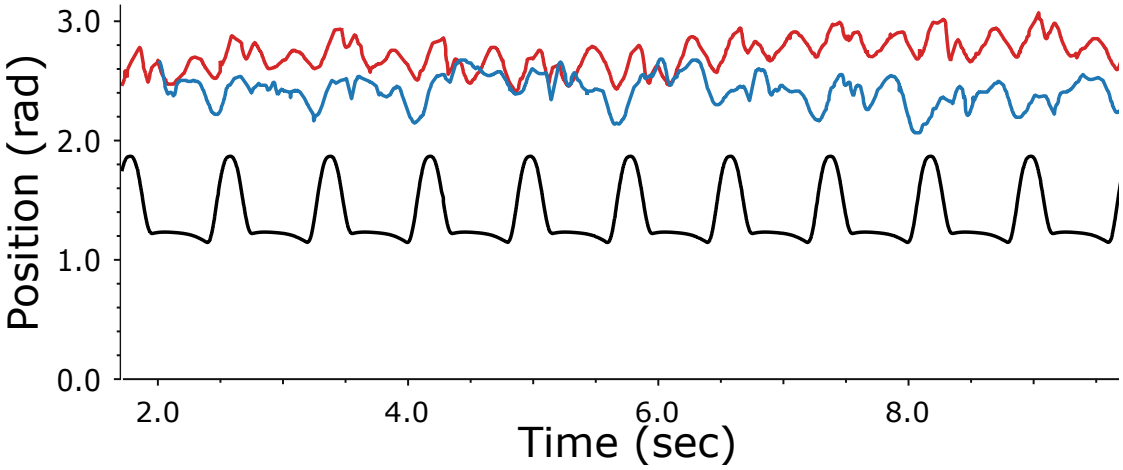
6.2.3. Transfer to Real Robot

In the final step, it was attempted to transfer the results from the previous Section to the real robot. However, the transfer faced some complications. The positive Siamese network produced the correction most closely, yet still did not manage to generate a sufficient walking motion (compare Figure 6.6).

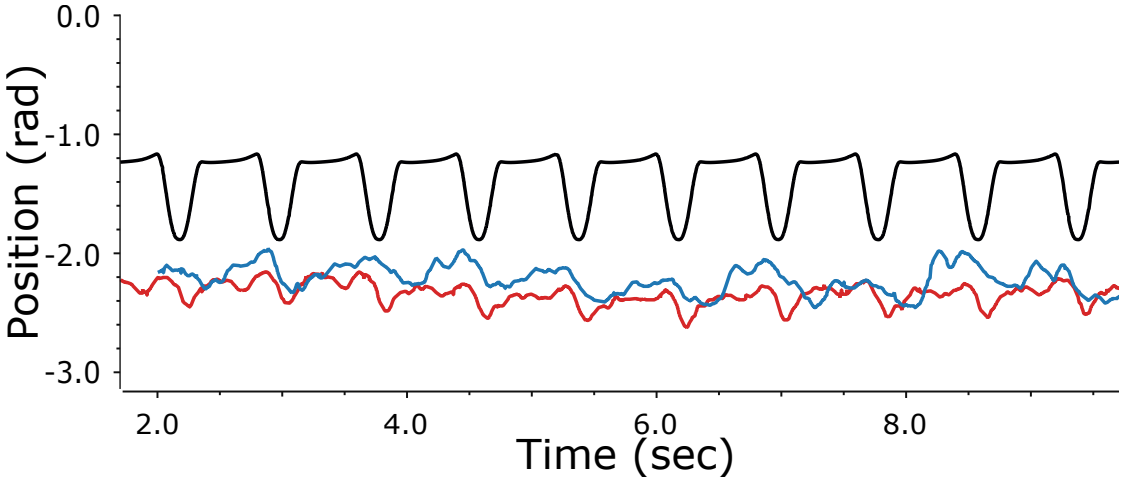
To be able to use the Siamese net, two consecutive samples need to be provided, as the net predicts the transition between the states. This is not a problem when evaluating on a dataset where all information is pre-recorded and thus available. However, when transitioning to the real robot, it needs to be established how the samples are generated. For each prediction, the network requires velocity, effort, and Hall sensor position. Velocity and effort can be directly measured for the current time step x_t. While the Hall sensor position could also be measured, this would require the Hall sensor to stay attached to the robot, which is unwanted, as described above. Thus, as an alternative, the previous command is used as an input here, assuming that the network has reached its target previously. Both variants were evaluated, with the one using the previous command performing better. This is likely because the number of feedback loops is reduced, giving the network a regular input that is independent of its actions. For the target time step x_{t+1}, the Hall sensor position is the received command. As we do not care about either velocity or effort at the target, it is simply assumed that both should be the same as they are in the current step. In theory, however, this would allow for even more sophisticated motions than are possible without the network.

The network manages to generate a motion that roughly resembles the walking motion. However, the peaks and valleys are not articulated enough. This is likely because, compared to the previous experiments, the feedback loop is now closed, and the network's predictions actively influence the following sample of the networks. This can create oscillating effects. Furthermore, so far, the network has only ever seen the relations between one sample and the immediately following. The changes between these samples are tiny, which means the network could only learn how to correct minor errors. When faced with more significant errors, the corrections do not suffice to generate the expected motions. Finally, due to the weight of the robot, there naturally is a significant offset between the target position and the reading of the Hall sensor. This happens as the SEA is constantly displaced by a significant amount when placed on the ground. The network was not able to negate this offset. The offset could potentially be mitigated by using stronger SEAs, but those would, in turn, reduce their benefits. The motions of the motor and the Hall sensor readings seem to lag behind the sent commands. This is expected behaviour and also happens in those cases where no corrective actions are applied.

When comparing the results of the network with the case of not taking any corrective action, we can see a slight improvement. This indicates that this approach, in principle, works but does not have a large enough impact to reach the correct motion successfully. Especially the amplitude of the generated motion is more pronounced compared to not correcting it.



(a) Right Knee



(b) Left Knee

Figure 6.6.: Positions of the Hall sensors when using or not using the neural network. (a) shows the right knee, (b) shows the left knee. **Black** lines indicate the target motion. **Red** line shows sensor readings when taking no corrective action, **blue** line shows sensor readings when applying the Siamese net.

6.3. Disturbing Factors of Hall Sensors

Several experiments were conducted to ensure that the readings of the Hall sensors were accurate in the noisy environment of a moving robot. First, the effect of the actuator's magnetic field on the hall sensor was evaluated in Section 6.3.1. Secondly, the effects of displacement of the magnet were evaluated in Section 6.3.2.

6.3.1. Magnetic Fields

Electric servomotors work by generating a strong magnetic field to rotate the output shaft. These magnetic fields could influence the readings of the Hall sensor, which is mounted close to the actuator. To ensure that this is not the case, an experiment was conducted.

The magnetic field of a linear solenoid can be calculated as

$$B = \frac{\mu N I}{L} \tag{6.1}$$

where μ is the permeability, N the number of windings, I the current and L the length of the solenoid.

The servo used in this thesis is the Dynamixel XH540-W270. According to the datasheet, it uses a Maxon motor. From the products listed on the Maxon store page, this is most likely the DLX22Lø22mm [42]. This seems likely, as it matches the no-load speed of $11700\,rpm$, stall torque of $348\,mNm$ and diameter of $22\,mm$ mentioned in the Dynamixel data sheet. This actuator uses three solenoids in a triangular pattern with a magnetic core. Due to the shape of the motor, each solenoid can be at most 8 mm long, for a total length of $L = 24\,mm$. The motor described above uses a NdFeB magnet. This material has an average permeability of $\mu = 1.32 \cdot 10^{-6}$. While the number of windings for this model is not specified, similar Maxon motors possess around $N = 930$ windings. The current at stall torque in the XH540-W270 motor is 5.9 A. This means the magnetic field of the solenoids equates

$$B = \frac{1.32 \cdot 10^{-6} \cdot 930 \cdot 5.9\,A}{0.024\,m} = 0.301\,T \tag{6.2}$$

A magnetic field of this strength would be strong enough to oversaturate the Hall sensor, which is rated for 30 to 90 mT. However, the magnetic fields of solenoids are highly directional, and the solenoids would be mounted in a parallel plane to the sensor, reducing its influence. Furthermore, the sensor would be mounted about 3 cm from the solenoids, which would greatly reduce any remaining magnetic fields, according to the inverse square law.

To evaluate whether these assumptions hold, an experiment was set up in which the motor horn was fixed. That way, a current could be induced into the motor without moving the magnet, which the Hall sensor should read. During this experiment, no influence of the magnetic field of the actuator could be detected.

6.3.2. Displacement

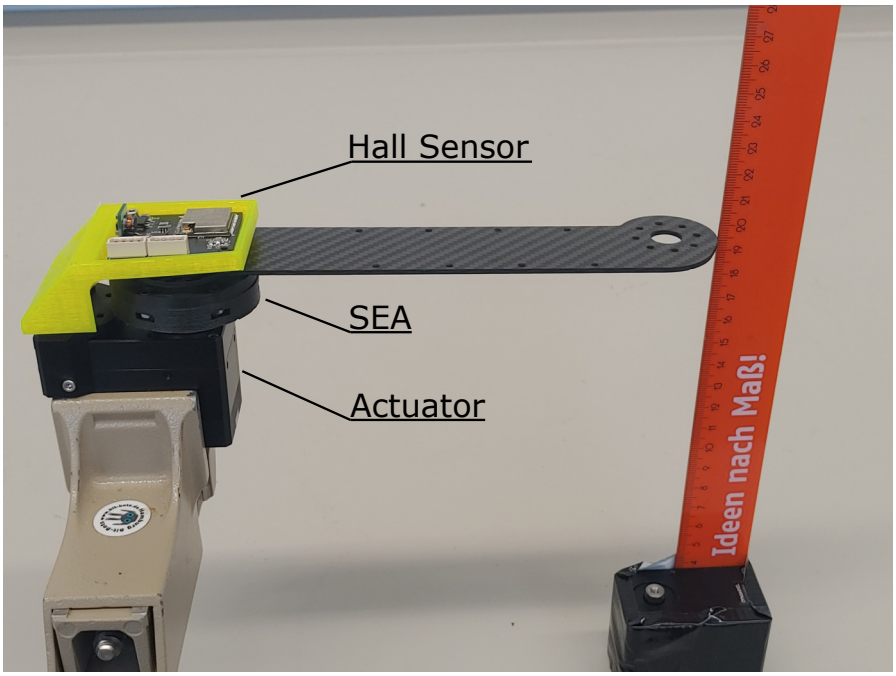


Figure 6.7.: Setup of the displacement experiment. Cables have been removed for better visibility.

The next issue that needs to be evaluated is the inaccuracies caused by the displacement of the magnet along axes other than the rotational axis of the motor. Since a SEA is made from an elastic material, it can be bent in any direction. Especially since the TPU used in this thesis is a very soft material, and the loads on a humanoid robot are often uneven, these displacements will necessarily happen. To study the effects on the sensor readings, an experiment was conducted.

The SEA was fitted with a lever of known length, which could be moved to cause a rotation in the SEA. Since the SEA is rotationally symmetric, only one rotational axis was evaluated, assuming that the other axis would behave similarly. The experiment setup can be seen in Figure 6.7. The translation was not evaluated since the layout of the robot does not allow any relevant amount of translational displacement. Different magnet orientations were briefly evaluated, confirming that the behaviour is the same across all directions. The lever was displaced by $1\,cm$ in either direction, causing a rotation of $3.053°$ at a lever length of $0.188\,m$. Further displacement could not be evaluated for the risk of damaging the SEA. However, since the results seem to follow a linear function (compare Figure 5.5), it can be assumed that it would keep following that trajectory for larger displacements.

The results show an average error of $1.65°$ per one degree of displacement, with a peak error of $6.53°$ at maximum positive and $-4.07°$ at maximum negative displacement. While this error is still reasonably small, it is yet another error source that could disturb

the results. It should also be noted that the error heavily depends on the calibration of the sensor, and an inaccurate calibration could cause a significantly larger error.

6.4. Using SEAs as torque sensors

Since the spring constant of the SEA is known, we can use it to estimate the torque emitted by the actuator. To do so, the same equation is used as for determining the spring constant:

$$\tau = k \cdot \theta \tag{6.3}$$

In the following, the accuracy of this estimation is evaluated and compared to the torque estimation of the actuator itself. As ground truth, a six-axis force/torque sensor is used. The SEA is mounted directly to the actuator, with the force/torque sensor behind it. Finally, the other side of the sensor is fixed. As the output of the actuator cannot move in this setup, the reading of the Hall sensor cannot change. Thus it has been omitted in this setup and replaced by a fixed value. The torque of the servo was gradually increased until reaching 4.8 Nm. The PWM limit register was used for this while the actuator was in position control mode, as defined in the Dynamixel documentation[3]. Torques in both directions were evaluated. The setup can be seen in figure 6.8.

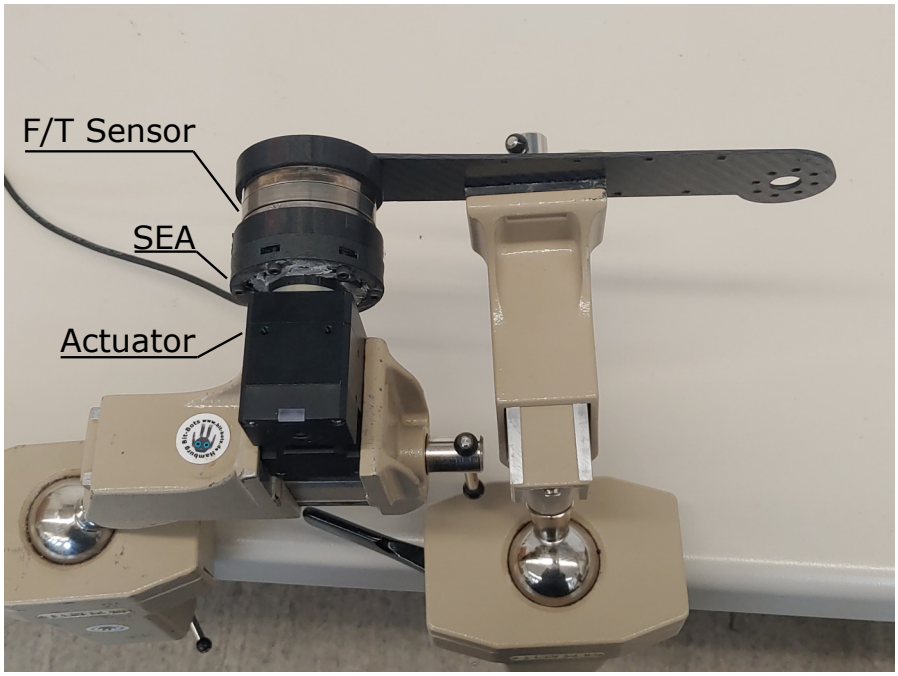


Figure 6.8.: Setup of the torque measurement experiment. Cables have been removed for better visibility.

[3] https://emanual.robotis.com/docs/en/dxl/x/xh540-w270/

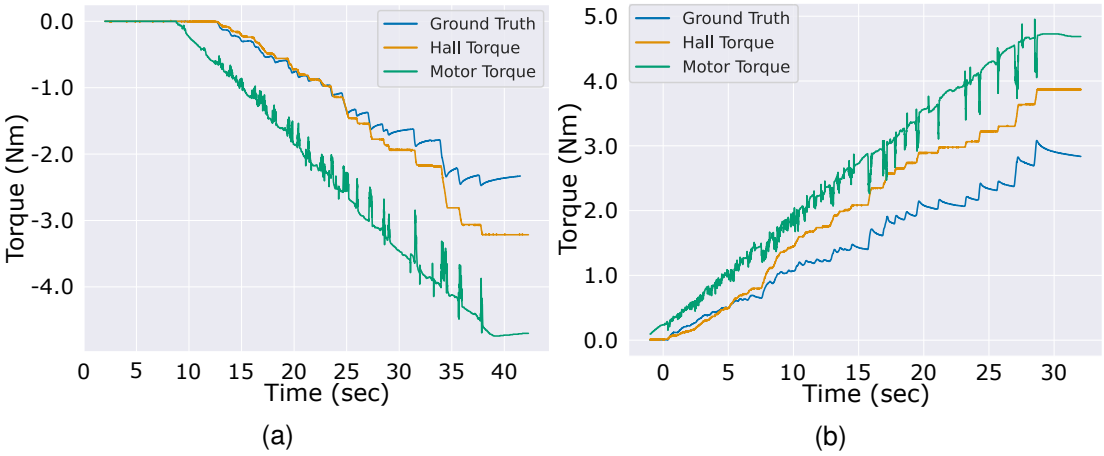


Figure 6.9.: Torque measured by Hall sensor and by servomotor, compared to ground truth measured by force/torque sensor. (a) shows a positive torque applied, while (b) shows a negative torque applied. Motor torque readings have been inverted to match the direction of the other two.

The results are shown in Figure 6.9. It can be seen that the calculated torque from the SEA matches the ground truth way closer than the predictions of the servo, especially for lower torques. Sudden increases in torque can be seen at points where the SEA gives in and allows more movement. This is accurately represented with the Hall sensor calculation, whereas the motor returns very noisy data at these points. The servo torque measurement is noisier overall and does not reflect the shape of the torque function. On average, the error of the torque predicted by the servo is 2.11 Nm for the positive direction, whereas it is only 1.18 Nm for the Hall sensor. For the negative direction, it is 1.75 Nm and 1.35 Nm, respectively. One important finding is that the servo motor returns a torque value before any torque is exerted at the motor horn. Since it relies on the displacement of the SEA, the Hall sensor measures this correctly instead. This difference between applied and perceived torque could be explained by the fact that the servo needs to apply a certain force to overcome the friction in the gearbox (compare section 3.1). During that time, the servo is applying a force to the gear train, but it is not strong enough yet to cause any noticeable torque at the motor horn.

6.5. Energy Consumption

In the final experiment, the potential benefits in energy consumption are evaluated. For this experiment, the robot was instructed to walk forward at a speed of 0.1 m/s for 30 seconds. During this time, the currents of one of the knee motors were monitored. As the robot is symmetric, the other knee motor is expected to consume a similar amount of energy. From this information, the electric charge can be estimated by assuming that the current does not change between measurements.

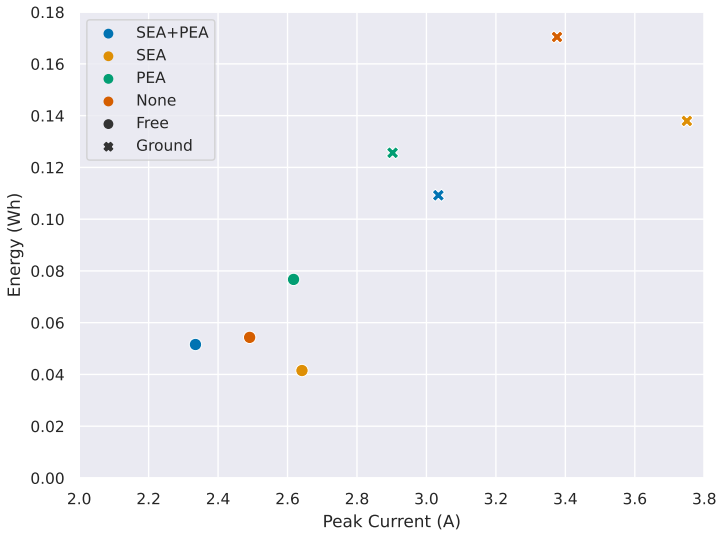


Figure 6.10.: Peak current and energy consumption of a single knee actuator while the robot is walking for 30 seconds in different configurations.

The electric charge can then be calculated as

$$Q = \sum_{i=0}^{n} I_i \cdot (t_i - t_{i-1}) \tag{6.4}$$

where I_i is the current at step i, and t is the timestamp. Given a high enough sample rate, this closely approximates the actual electric charge. As the voltage on the motor bus is constant, the energy consumption can easily be calculated as

$$E = Q \cdot U \tag{6.5}$$

where U is the voltage. The peak current of each trial was recorded simultaneously. Each combination of SEA and PEA was evaluated while walking on the ground and suspended in the air. The results are shown in Figure 6.10.

It can be seen that both SEA and PEA help to reduce the energy consumption of the knee actuators when the robot is placed on the ground. The SEA does so at the cost of increasing peak currents, whereas the PEA manages to decrease both. The combination of SEA and PEA reduces energy consumption by 35.89%, which is the most significant improvement of all candidates. Using only a PEA reduces energy consumption by 26.26% but reduces the peak currents further than using both.

For the case of the robot being suspended in the air, the PEA increases energy consumption. This happens as the actuator has to counteract the spring force when moving in one direction but does not gain any of the benefits when moving in the other (see also [25]).

Overall, the combination of both SEA and PEA performs very well in both scenarios. It achieves the lowest energy consumption and second lowest peak current when walking on the ground and second lowest energy consumption and lowest peak current when suspended in the air. Ultimately, any of the attachments bring an improvement over the default case when the robot is walking on the ground. The PEA by itself performs better than the SEA does by itself and thus should be preferred. Whether a SEA should also be added depends on whether peak current or total energy consumption is more critical to the use case.

These findings mostly align with the results of [2] and [1], who also attribute improvements in energy consumption to both elements. They confirm that PEAs are better at reducing peak currents and that combining both elements results in the lowest energy consumption. What cannot be confirmed from these findings is that SEAs reduce energy consumption significantly better than PEAs. This might be related to the shape of the motion. It might also be explained by the fact that the PEA was specifically selected to reduce energy consumption in the 60 degree orientation [25]. In contrast, the SEA was selected to perform best overall. As the robot spends most of its time with its legs bent at a 60 degree angle while walking, the PEA might have an advantage.

7.Discussion

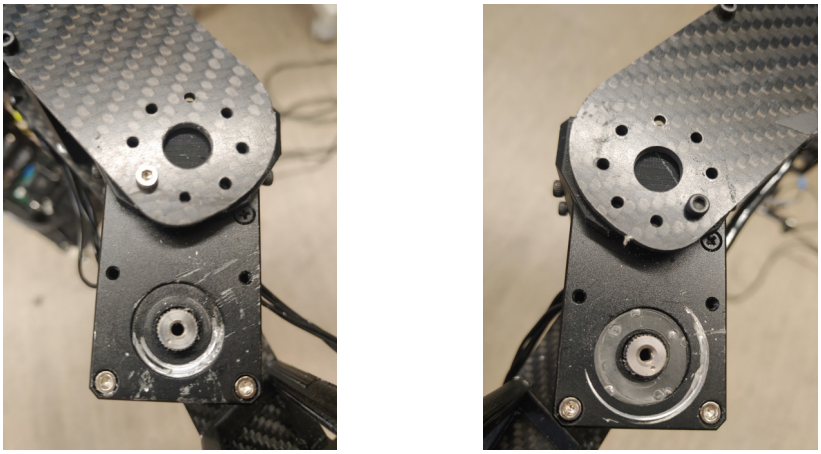
The results shown in the previous Section indicate that the use of SEAs in the knee motors of a humanoid robot is possible and provides several benefits. It can reduce energy consumption and peak currents in the actuators, protect the gearboxes, and soften impacts. It is also a valuable step in the direction of soft robotics. As an additional effect, the SEA, in combination with a Hall sensor, can be used to estimate the torques of the actuator. These estimations are significantly more precise than those of the Dynamixel actuators.

Nonetheless, the introduction of SEAs in the robot's legs cause several issues. The main problem is that the SEA is barely capable of supporting the robot's weight. As the robot spends most of the time with its legs slightly bent, most of its mass acts as a torque on the knee, using the upper leg as a lever. This gets better when the proposed neural networks are used to mitigate the error as the actuator moves further to compensate for the displacement. However, even with this measure, the improvement is not significant enough to keep the robot upright, eventually causing it to fall. This problem could be reduced by using a more powerful spring as a SEA. However, stronger springs would reduce the benefits of adding the compliant element. According to the previous analysis, the spring used as a SEA in this thesis was already the strongest spring that could be manufactured with this design and material. Stronger springs also usually take up more space and have higher prices. Both of these factors should be considered when designing a low-cost small humanoid robot. One option would be using a more rigid filament, such as the NinjaTek Cheetah[1], which has a shore hardness of 95A, compared to the NinjaFlex filament, which only has a shore hardness of 85A.

As the SEAs used in this thesis are 3D-printable, they are very cheap and readily available. However, it also brings a vast disadvantage. As 3D printers work by heating a plastic, the filament needs to be very sensitive to heat. This is problematic, as the SEA absorbs energy when displaced, which is partially transformed into heat. That means that the SEA gets softer during extended use. This does not seem to cause the failure of the part, and it returns to its original properties after cooling down. However, after using it for a certain amount of time, the part turns so soft that it is no longer capable of supporting the robot. Even before reaching this point of failure, this can lead to issues, as the properties of the SEA change over time. This means the spring constantly changes unpredictably, making the control problem significantly harder. As a RoboCup humanoid league game only takes ten minutes per half with a five-minute break to cool down, this should not be a problem in that use case, but when looking at other use cases, such as companion robots, this is problematic.

The elasticity of the SEA also appears to be a problem concerning shear forces. As the robot does not hold its legs straight in the lateral dimension to give itself more stability, the forces are not spread evenly across the leg. Usually, this is not a problem, as the entirety of the leg is rigid enough to deal with these forces. However, adding an elastic element to only one side of the leg causes the leg to shift, resulting in a rotation

[1] https://ninjatek.com/shop/cheetah/



(a) Left knee (b) Right knee

Figure 7.1.: Wear to the knee motors from extended use.

of the SEA around the sagittal axis. This is problematic for two reasons. The first reason is that this rotation also causes the magnetic field measured by the Hall sensor to rotate. This, as described above, can result in significant errors in the measurement. The effect of this problem is increased when the temperature of the SEA increases or when the screws holding the SEA loosen. This problem could be combatted by adding a secondary Hall sensor on one of the other two axes of the magnet. This sensor could then measure the rotation and use that information to correct the sensor readings of the first sensor. Another approach to solving this problem would be adding two proximity sensors to the robot that measure the distance of the SEA. With the known size of the SEA, the rotation can be calculated from the distance of the SEAs edges to the proximity sensor. This information can then be used to correct the Hall sensor readings. However, the second problem caused by the rotation of the SEA cannot be countered that easily. As the SEA is rotated, it forces the motor horn into the encasing of the servo. As shown in Figure 7.1, this can cause severe damage to the motor over time. Solutions for resolving this issue will be discussed in the following section.

Another factor that should be discussed here is the geometry of the compliant elements. The design used in this thesis is based on the works of Martins et.al. [34] and was refined by Bestmann et al. [36]. Both papers show that the component works as intended, and the part has been thoroughly evaluated during this thesis. However, no comparison between this design and other variants has been drawn so far. By now, a vast array of different SEAs have been developed [43]. It is possible that some of these candidates perform better than the one used here. In future work, this should be further investigated.

The final problem the current setup causes is the space required to assemble all parts. Adding the SEA to the robot makes each knee approximately 2cm wider, and another centimetre if the Hall sensor is attached. This again increases shear forces, as the distance between the two carbon fibre plates of the leg is significantly increased.

Furthermore, it hinders the movement of the robot. As the torsion springs need to be moved to the inside of the legs, they are almost touching. This did not cause any problems during the evaluations of the thesis but could be problematic for some more dynamic motions.

7.1. Hardware Modifications

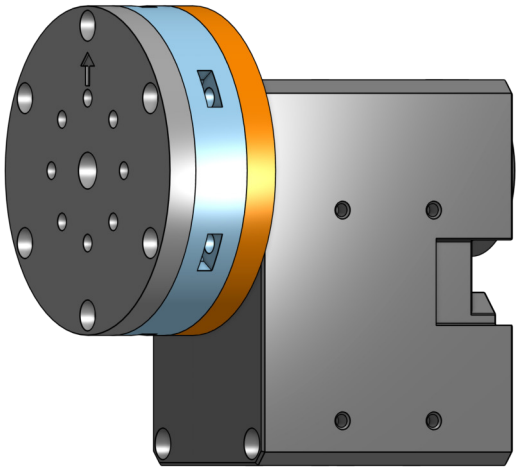


Figure 7.2.: SEA (**light blue**) with stabilizing back plate (**orange**).

The design could be significantly improved by changing the type of SEA that is being used. In order to support the weight of the robot more reliably, an even harder SEA would be needed. The proposed design, however, cannot generate springs with smaller displacements. One solution would be using springs made from maraging steel, as proposed in [12]. The springs proposed in that paper have a similar footprint that could probably be downscaled to this robot's size. They are also capable of resisting significantly stronger torques. However, using a spring like that would result in higher manufacturing costs and lower availability, as these springs cannot simply be 3D printed. Using a less elastic filament, such as NinjaTek Cheetah, could be a way to improve spring strength while still keeping the benefits of fast manufacturing.

Another solid component could be introduced to combat the problem of the SEA rotating sideways. [44] and [45] both use a solid central shaft to wind the spring around. Due to the shape of the motor horn and the flat but wide SEAs, this is not easily possible or useful in this setup. Instead, a solid component the size of the SEA could be added between the SEA and the motor horn (compare Figure 7.2). This would prevent the SEA from rotating in that direction. It should be taken care that the component does not break from the stresses applied to it. A design similar to that of the NUBots, who use

needle roller bearings[2] in their joints could also be used, achieving a similar function. Robotis also produces a bearing set that can be added to the servo to reduce radial load [3]. However, this would be the most expensive of the solutions shown.

Finally, additional sensors could be added to measure the displacement and temperature of the SEA. As the components necessary for communication are already in place, the sensor board could easily be extended without requiring much additional space. The ideal setup would add a secondary Hall sensor orthogonal to the first, which measures the rotation around the other two axes of the SEA. Alternatively, two proximity sensors can be placed between the servo and the SEA to gain the same information. A thermistor could also be added to measure the temperature, and thus the change in material properties.

7.2. Network Results

Finally, the results of the error correction approaches should be discussed. The results of the free-hanging experiments show that using a neural network to correct the introduced errors is valid and more effective than using a PI controller. In further experiments, it could be verified that these networks can also perform well when provided data from walking on the ground. Here it became clear that the velocity and effort of the actuator are the most effective modalities to solve the problem. This is very useful, as this information is provided directly from the actuator and does not require any additional sensors.

Executing the network on the real robot showed that the trained networks are real-time capable and can calculate the necessary corrective actions on the fly. While using the network yields an improvement, compared to taking no action at all, it still cannot sufficiently correct the error for the robot to be able to walk. This has multiple reasons. The biggest issue, caused by the use of a too weak SEA, has already been discussed in the previous Sections. As the SEA cannot hold the robot upright, a large offset is constantly present in the sensor readings. Apart from that, the main problem is that the amplitude of the generated motion, albeit more prominent than without taking an action, is still not large enough to generate a proper walking motion. This can be explained by two reasons. First, the network was trained in a way where it only ever sees the immediately next time step. As the changes between time steps are tiny, the network only ever learns to take small corrective actions. These, however, may not be sufficient to correct the errors encountered in this situation. Second, in this scenario, the feedback loop has been closed. This means the actions of the network immediately influence the following sample. This means the network may gravitate towards not taking any action if the input of the network is close to the output. As the training data mostly consisted of samples where the input and output were similar, it is likely, that the network learned this connection. The Siamese net counters this by using the command as an input instead of the previous position. Still, the problem remains, in part, for velocity and effort.

[2] https://robocup.informatik.uni-hamburg.de/en/2022/07/results-of-humanoid-league-survey-2/

[3] https://www.robotis.us/epx540-br101-set/

The problems could be reduced by conducting iterative training. The trained network could be used to record a new dataset, which would contain different observations. These observations can then be used to refine the network. It would also be possible to consider the relationship between further apart samples or provide a history of the past steps to the network.

8.Conclusion

This thesis explored ways to apply SEAs to a humanoid robot's knees. A SEA was modified from an existing design to achieve different spring constants, and a sensor board was developed to measure knee displacements effortlessly. Several neural networks were trained using this setup to combat the error introduced by the SEA. The thesis results show that using a neural network to solve this task is possible and comparable to PI controller-based approaches. The networks necessary for such a task are tiny and are easily real-time capable. The computational power required is small. While the network generally performs well, several improvements are still necessary to fully use the system. Still, the results provide a reasonable baseline to keep building upon.

The system is easily expandable, which allows the quick evaluation of new approaches to the problem. This adds an easy way to compare different solutions to each other, which could be extended to quickly evaluate new solutions in the future. The approach shown in this thesis also enables the accurate measurement of the torques applied by the servo through the displacement of the SEA. It has been shown that this measurement is more accurate than the estimates provided by the Dynamixel actuators.

Finally, it has been confirmed that the added SEAs improve the energy consumption of the servos.

8.1. Future Work

To further improve the results shown in this thesis, the hardware design should be modified as described in the discussion section. A more reliable hardware system is expected to result in more reliable networks and better overall error correction. The use of a pre-loaded spring should also be evaluated, as it can further improve the efficiency of the SEA. As the robot spends most of its time with the legs bent at a 60 degree angle, the spring could be modified to rest in that position naturally. This would result in a more considerable improvement in energy consumption, as the robot would require less energy to stay in this position. It would also help support the robot and allow for stronger springs.

The networks should also be further evaluated. First, iterative training should be conducted to evaluate whether the results improve. The effect of adding a history to the network's inputs should also be evaluated.

Finally, the OgmaNeo architecture should be evaluated[1]. OgmaNeo is an implementation of the Feynman machine [46]. It uses online predictive hierarchies to predict the following action. The approach enables continuous learning, which would be beneficial to this problem, as the network could adapt to different scenarios on the fly. This could allow the robot to traverse challenging terrain. The OgmaNeo architecture is very low weight and can be run directly on a microcontroller and could thus be run on the Hall sensor board.

[1] https://ogma.ai/

Acknowledgements

I would like to thank the members of the Hamburg Bit-Bots RoboCup team, who supported me in any possible way and were always available for questions. I especially thank Jasper Güldenstein and Niklas Fiedler for helping me debug some hardware problems. I also want to thank Oubaida Errogui for taking his time to discuss various physical concepts with me.

Bibliography

[1] Tom Verstraten et al. “Series and parallel elastic actuation: Impact of natural dynamics on power and energy consumption”. In: *Mechanism and Machine Theory* 102 (2016), pp. 232–246.

[2] Martin Grimmer et al. “A comparison of parallel-and series elastic elements in an actuator for mimicking human ankle joint in walking and running”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 2463–2470.

[3] Gill A Pratt and Matthew M Williamson. “Series elastic actuators”. In: *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*. Vol. 1. IEEE. 1995, pp. 399–406.

[4] Matthew M. Williamson. “Series Elastic Actuators”. Technical Report. Massachusetts Institute of Technology.

[5] Hiroaki Kitano et al. “Robocup: The robot world cup initiative”. In: *Proceedings of the first international conference on Autonomous agents*. 1997, pp. 340–347.

[6] Jerry Pratt, Ben Krupp and Chris Morse. “Series elastic actuators for high fidelity force control”. In: *Industrial Robot: An International Journal* 29.3 (2002), pp. 234–241.

[7] Kyoungchul Kong, Joonbum Bae and Masayoshi Tomizuka. “Control of rotary series elastic actuator for ideal force-mode actuation in human–robot interaction applications”. In: *IEEE/ASME transactions on mechatronics* 14.1 (2009), pp. 105–118.

[8] Mohammad Javad Fotuhi, Zied Ben Hazem and Zafer Bingül. “Modelling and torque control of an non-linear friction inverted pendulum driven with a rotary series elastic actuator”. In: *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. IEEE. 2019, pp. 1–6.

[9] Gordon Wyeth. “Control issues for velocity sourced series elastic actuators”. In: *Proceedings of the 2006 Australasian Conference on Robotics and Automation*. Australian Robotics and Automation Association. 2006, pp. 1–6.

[10] Heike Vallery et al. “Compliant actuation of rehabilitation robots”. In: *IEEE Robotics & Automation Magazine* 15.3 (2008), pp. 60–69.

[11] Dino Accoto et al. “Robomorphism: a nonanthropomorphic wearable robot”. In: *IEEE Robotics & Automation Magazine* 21.4 (2014), pp. 45–55.

[12] Fabrizio Sergi et al. “Design and characterization of a compact rotary series elastic actuator for knee assistance during overground walking”. In: *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. IEEE. 2012, pp. 1931–1936.

[13] José de Gea Fernández et al. “Design, modelling and control of novel series-elastic actuators for industrial robots”. In: *Actuators*. Vol. 9. 1. MDPI. 2020, p. 6.

[14] Gerardo Bledt et al. “MIT Cheetah 3: Design and control of a robust, dynamic quadruped robot”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 2245–2252.

[15] Jemin Hwangbo et al. “Learning agile and dynamic motor skills for legged robots”. In: *Science Robotics* 4.26 (2019), eaau5872.

[16] Dmytro Pavlichenko and Sven Behnke. “Flexible-Joint Manipulator Trajectory Tracking with Two-Stage Learned Model utilizing a Hardwired Forward Dynamics Prediction”. In: *International Journal of Semantic Computing* (2022).

[17] Pulkit Agrawal et al. “Learning to poke by poking: Experiential learning of intuitive physics”. In: *Advances in neural information processing systems* 29 (2016).

[18] Luu Anh Khoa Lanh et al. “Hybrid adaptive control for series elastic actuator of humanoid robot”. In: *International Journal of Intelligent Unmanned Systems* (2022).

[19] Marco Hutter et al. “Scarleth: Design and control of a planar running robot”. In: *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2011, pp. 562–567.

[20] Yin Wei et al. “Position control of Series Elastic Actuator based on feedback linearization and RISE method”. In: *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2016, pp. 1203–1208.

[21] Junho Chang et al. “Verification of Mechanical Improvement for Bipedal Robot Walking”. In: *Transaction of the Japan Society for Simulation Technology* 14.1 (2022), pp. 28–35.

[22] George Ellis. *Control system design guide: using your computer to understand and diagnose feedback controllers*. Butterworth-Heinemann, 2012.

[23] HP Olesen and RB Randall. “A Guide to Mechanical Imped ance and Structural Response Techniques”. In: *Briiel & Kjaer Ap plication Note* 17-179 (1977).

[24] Ryojun Ikeura and Hikaru Inooka. “Variable impedance control of a robot for cooperation with a human”. In: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*. Vol. 3. IEEE. 1995, pp. 3097–3102.

[25] Marc Bestmann et al. “Wolfgang-OP: A Robust Humanoid Robot Platform for Research and Competitions”. In: *IEEE Humanoids 2021*. to appear. July 2021.

[26] Edwin H Hall et al. “On a new action of the magnet on electric currents”. In: *American Journal of Mathematics* 2.3 (1879), pp. 287–292.

[27] Edward Ramsden. *Hall-effect sensors: theory and application*. Elsevier, 2011.

[28] Neha Wadehra et al. “Planar Hall effect and anisotropic magnetoresistance in polar-polar interface of LaVO3-KTaO3 with strong spin-orbit coupling”. In: *Nature communications* 11.1 (2020), pp. 1–7.

[29] S Nandy et al. “Chiral anomaly as the origin of the planar Hall effect in Weyl semimetals”. In: *Physical review letters* 119.17 (2017), p. 176804.

[30] ams, ed. *AS5600 Datasheet. 12-Bit Programmable Contactless Potentiometer*. Version v1-06.

[31] Mark C Hopkins Sr. “The state of the art in hall effect technology and its implications for appliance design and development”. In: *on line] July* (2004).

[32] Marc Bestmann et al. “WF Wolves & Hamburg Bit-Bots Team Description for RoboCup 2018–Humanoid TeenSize–”. In: ().

[33] Max Schwarz et al. “Humanoid teensize open platform nimbro-op”. In: *Robot Soccer World Cup*. Springer. 2013, pp. 568–575.

[34] Leandro Tomé Martins et al. “A polyurethane-based compliant element for upgrading conventional servos into series elastic actuators”. In: *IFAC-PapersOnLine* 48.19 (2015), pp. 112–117.

[35] Marc Bestmann, Jasper Güldenstein and Jianwei Zhang. “High-Frequency Multi Bus Servo and Sensor Communication Using the Dynamixel Protocol”. In: *RoboCup 2019: Robot World Cup XXIII*. Springer. 2019.

[36] Marc Bestmann and Jianwei Zhang. “Bipedal Walking on Humanoid Robots through Parameter Optimization”. In: *RoboCup Symposium*. July 2022.

[37] Sebastian Stelter et al. “Fast and Reliable Stand-Up Motions for Humanoid Robots Using Spline Interpolation and Parameter Optimization”. In: *IEEE ICAR*. Dec. 2021.

[38] Richard H MacNeal. *The NASTRAN theoretical manual*. Vol. 221. Scientific, Technical Information Office, National Aeronautics and Space . . ., 1970.

[39] Shady Farah, Daniel G Anderson and Robert Langer. “Physical and mechanical properties of PLA, and their functions in widespread applications—A comprehensive review”. In: *Advanced drug delivery reviews* 107 (2016), pp. 367–392.

[40] Eddo Hobert. “3D Printed flexible fingertip strain sensor”. B.S. thesis. University of Twente, 2017.

[41] Jane Bromley et al. “Signature verification using a" siamese" time delay neural network”. In: *Advances in neural information processing systems* 6 (1993).

[42] Maxon. *Product Range 2022/23*. Maxon.

[43] Cornelius Irmscher et al. “Design, optimisation and testing of a compact, inexpensive elastic element for series elastic actuators”. In: *Medical engineering & physics* 52 (2018), pp. 84–89.

[44] Chan Lee and Sehoon Oh. “Configuration and performance analysis of a compact planetary geared elastic actuator”. In: *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2016, pp. 6391–6396.

[45] Umberto Scarcia et al. “Optimal design of 3D printed spiral torsion springs”. In: *Smart Materials, Adaptive Structures and Intelligent Systems*. Vol. 50497. American Society of Mechanical Engineers. 2016, V002T03A020.

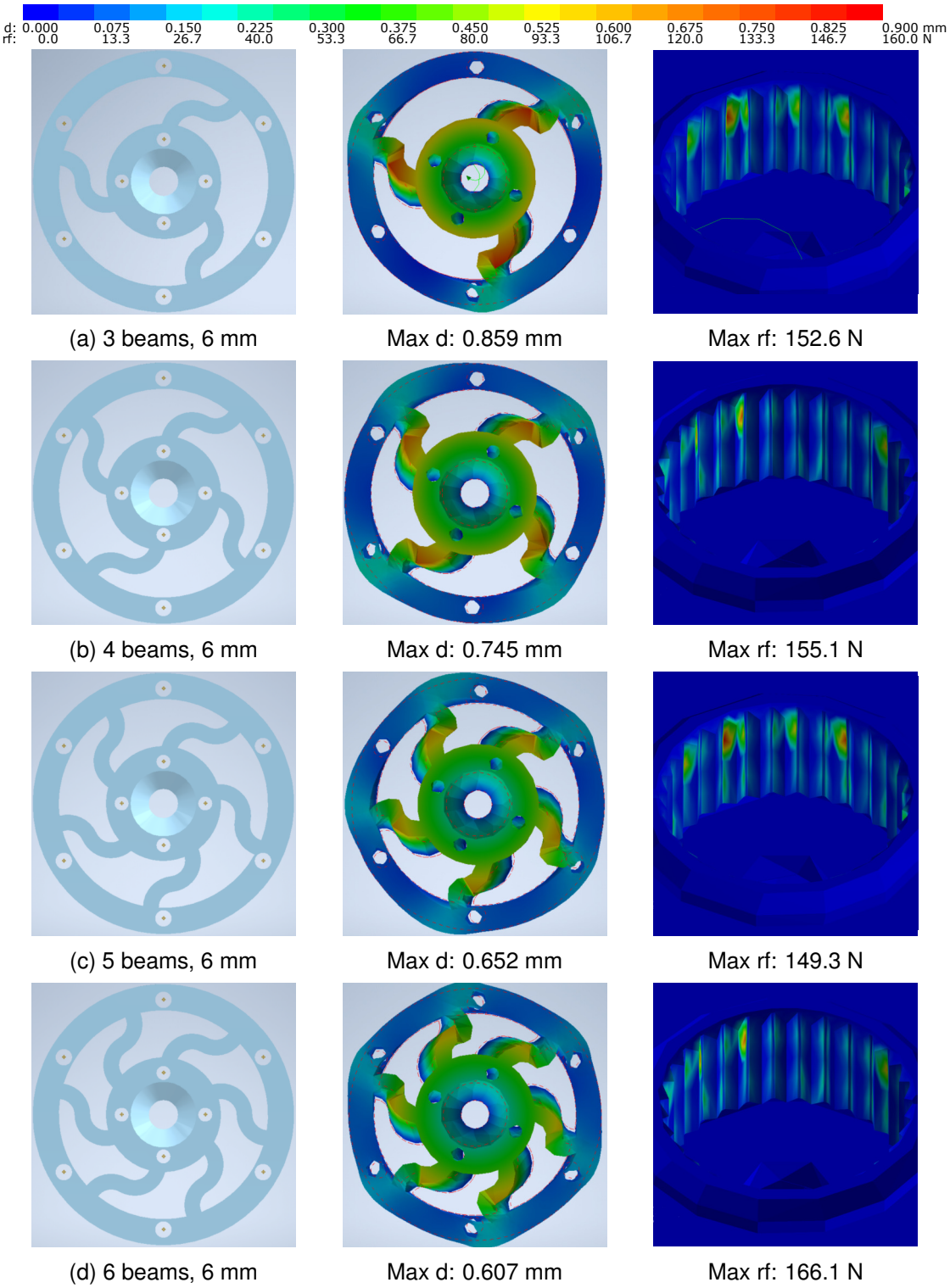
[46] Eric Laukien, Richard Crowder and Fergal Byrne. “Feynman machine: The universal dynamical systems computer”. In: *arXiv preprint arXiv:1609.03971* (2016).

Appendices

A.Finite Element Analysis

NinjaFlex			
	unit	value	source
Mass density	g/cc	1.19	Datasheet
Young's Modulus	GPa	0.147	matweb (generic TPU)
Poisson Ratio	-	0.48	[40]
Tensile Limit	MPa	26	Datasheet
Yield Limit	MPa	4	Datasheet
PLA			
	unit	value	source
Mass density	g/cc	1.28	matweb (generic PLA)
Young's Modulus	GPa	2.27	matweb (generic PLA)
Poisson Ratio	-	0.36	[39]
Tensile Limit	MPa	59	matweb (generic PLA)
Yield Limit	MPa	70	matweb (generic PLA)
Brass (soft yellow)			
	unit	value	source
Mass density	g/cc	8.47	Inventor Material Library
Young's Modulus	GPa	1096	Inventor Material Library
Poisson Ratio	-	0.331	Inventor Material Library
Tensile Limit	MPa	275	Inventor Material Library
Yield Limit	MPa	103.4	Inventor Material Library

Table A.1.: Material properties used in the finite element analysis, along with the corresponding sources.



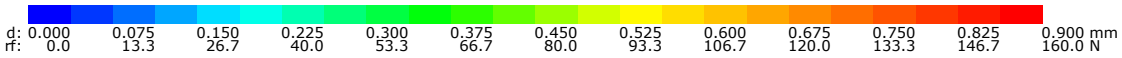
(a) 3 beams, 6 mm — Max d: 0.859 mm — Max rf: 152.6 N

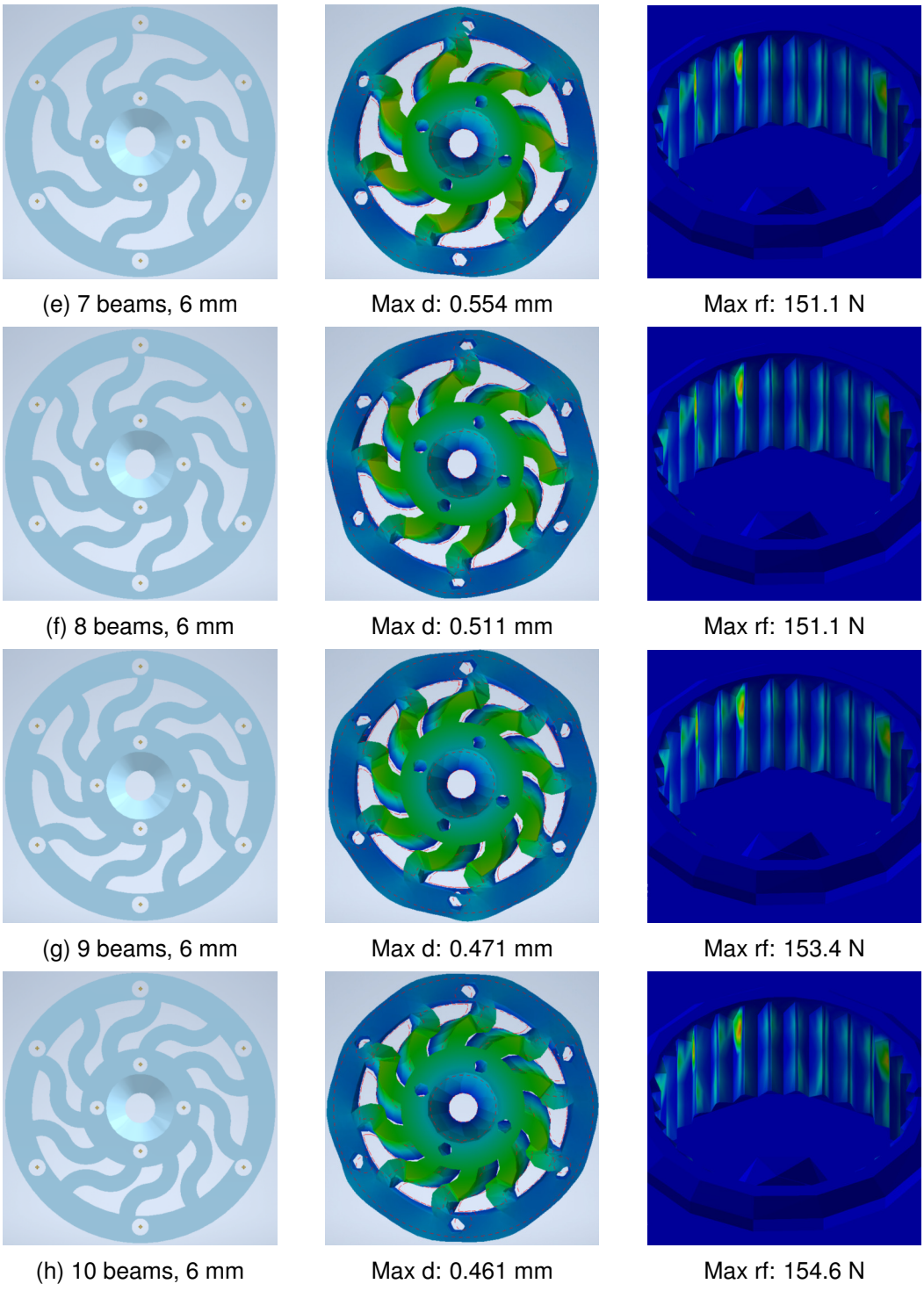
(b) 4 beams, 6 mm — Max d: 0.745 mm — Max rf: 155.1 N

(c) 5 beams, 6 mm — Max d: 0.652 mm — Max rf: 149.3 N

(d) 6 beams, 6 mm — Max d: 0.607 mm — Max rf: 166.1 N

Figure A.1.: CAD model, displacement and reaction force for the models with 6 mm beams. Red dashed line in displacement graph indicates original shape. Reaction force graph shows the inner ridge of the motor horn. Legends are provided at the top.





(e) 7 beams, 6 mm | Max d: 0.554 mm | Max rf: 151.1 N

(f) 8 beams, 6 mm | Max d: 0.511 mm | Max rf: 151.1 N

(g) 9 beams, 6 mm | Max d: 0.471 mm | Max rf: 153.4 N

(h) 10 beams, 6 mm | Max d: 0.461 mm | Max rf: 154.6 N

Figure A.1.: CAD model, displacement and reaction force for the models with 6 mm beams. Red dashed line in displacement graph indicates original shape. Reaction force graph shows the inner ridge of the motor horn. Legends are provided at the top.

B.Hall Sensor Board

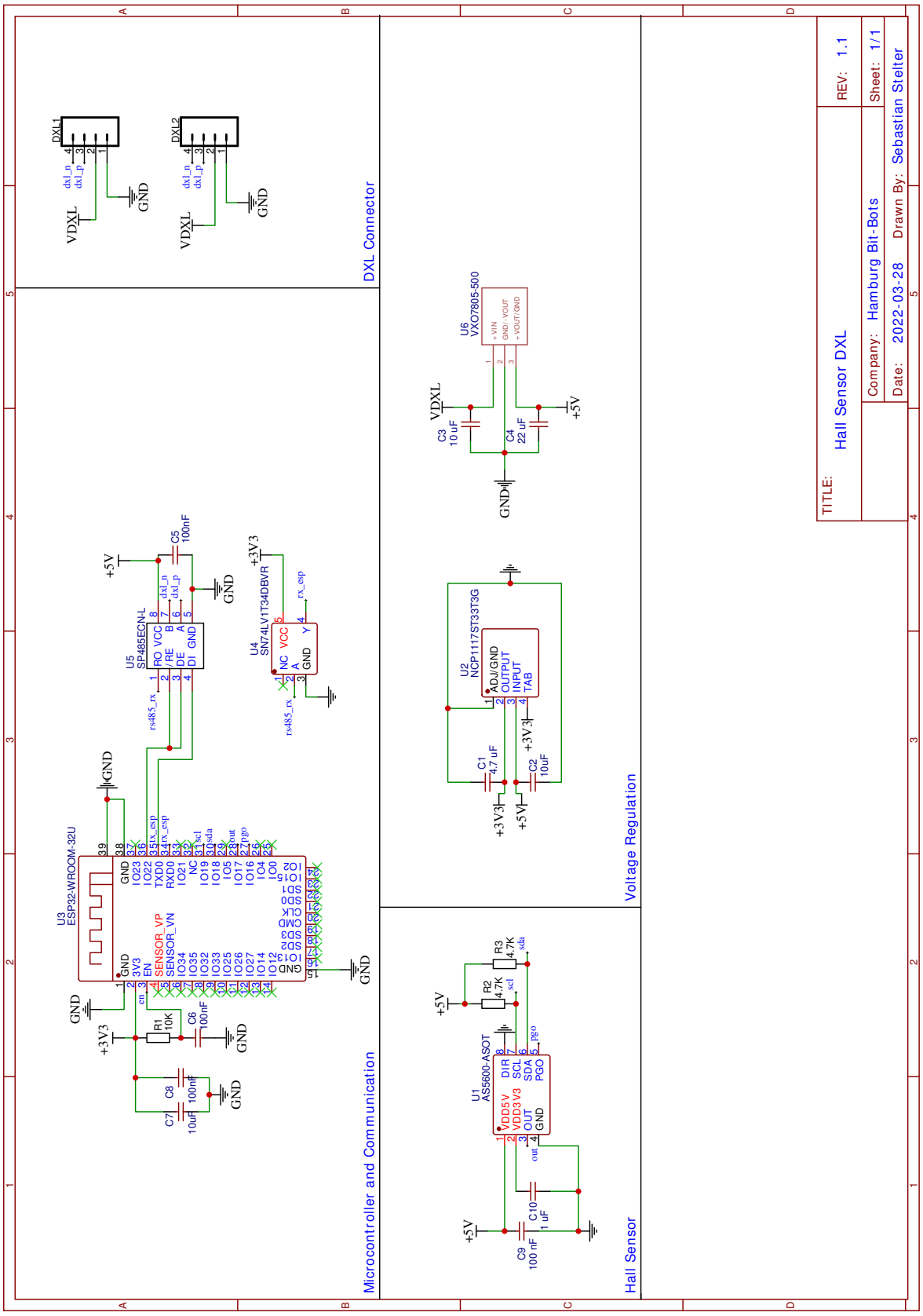
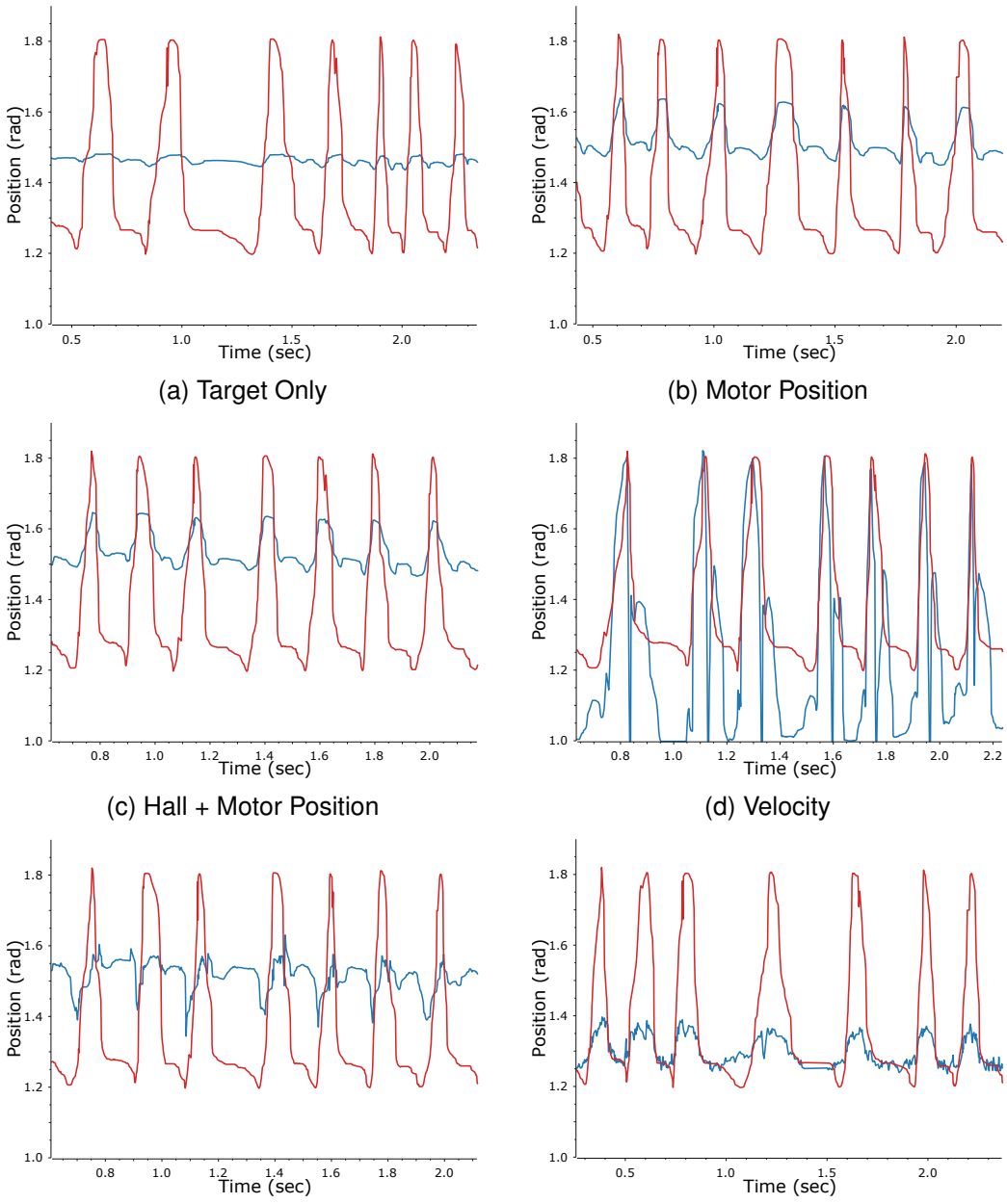


Figure B.1.: Schematic of the hall sensor board.

C.Ablation Study Results



(a) Target Only

(b) Motor Position

(c) Hall + Motor Position

(d) Velocity

(e) Effort

(f) IMU

Figure C.1.: Example predictions of the candidates for the left knee. Targets are shown in **red**, predictions in **blue**. All graphs show the same seven steps taken from the test dataset. X axis represents experiment time and is not related to real life robot speed.

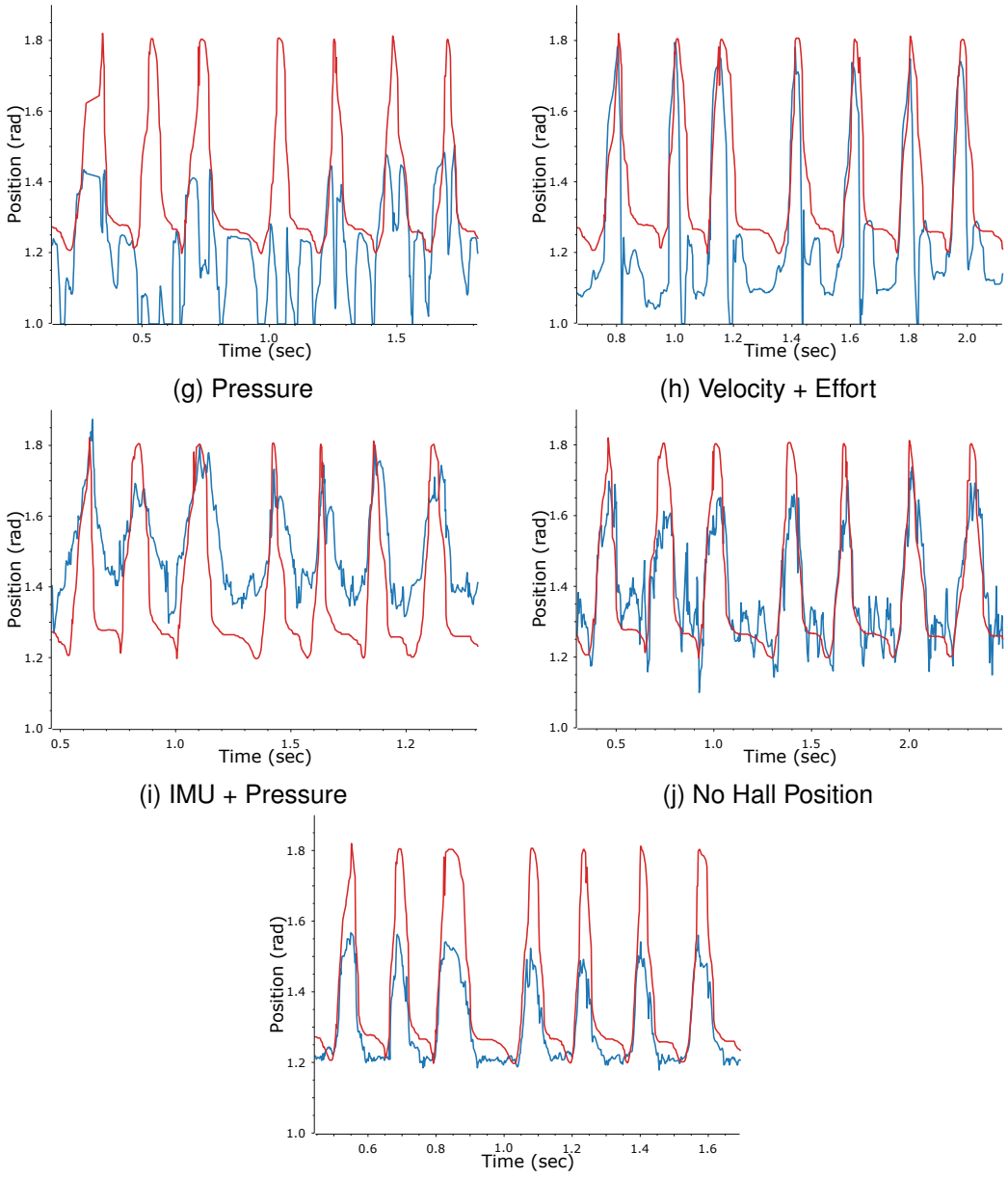


Figure C.1.: Example predictions of the candidates for the left knee. Targets are shown in **red**, predictions in **blue**. All graphs show the same seven steps taken from the test dataset. X axis represents experiment time and is not related to real life robot speed.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der elektronischen Abgabe entspricht.

Hamburg, den 21.10.2022 Sebastian Stelter

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den 21.10.2022 Sebastian Stelter