Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Design and Control of Biologically Inspired Joints

Bachelor Thesis
in the Research Group
Knowledge Technology, WTM

Department of Informatics
MIN Faculty
Universität Hamburg

presented by
Nils Rokita
on
16.07.2015

Examiners: Dr. S. Magg
B. Schütz

Nils Rokita
Matrikelnummer: 6236557
Ecklоßberg 10 a
22391 Hamburg

Abstract

As robots have to do more and more human tasks they need more human like moving capabilities. In this thesis a biologically inspired joint operating with tendons is build. The controlling for such joints is examined on this example. It is tried to control the joint with learning algorithms like goal babbling and Neuronal Nets.

Joints operated by tendons have some advantages over motors directly in the joint. For example if the joint has not enough space as in a finger. Or if the elastic properties of the joint can protect the motor from the forces if the robot falls down. On the other side controlling a joint operated by tendons is much harder then a simple servo controlled by a PID-Controller.

Zusammenfassung

Da Roboter immer menschlichere Aufgaben wahrnehmen sollen brauchen sie auch zunehmend menschliche Bewegungsfhigkeiten. Daher wurde in dieser Arbeit ein beispielhaftes Gelenk entworfen welches wie das biologische Vorbild mit Sehenen angetrieben wird. Für dieses Gelenk wurden dann verschiedene Steuererungsalgorithmen ausprobiert, unter anderem die lernenden Verfahren Goal Babling und Neuronale Netze.

Sehnen getriebene Gelenke sind in einigen Bereichen besser als die Motoren direkt in das Gelenk zu bauen, zum Beispiel wenn wenig Platz ist, wie in den Fingern, oder wenn durch die elastizitt der Sehne die Hardware geschtzt werden kann, indem schwere Strze nicht direkt auf den Motor gehen. Dar ist die Steuerung eines Gelenkes welches ber Sehnen angesteuert wird koplizierter als ein einfacher Servo mit einem PID Controller.

Contents

List of Figures

Chapter 1

Introduction

Robots take more and more part in our world. They already have a great part in the industries, doing a lot of repetitive tasks. In this field specialized robots can exceed humans. There are a lot of fields which need much more human like features if robots should take part in this work too. For example multi purpose robots for the rescue area, which need to interact with the human world in a robust manner. One of the needed features are hand like grippers for interacting with the world.

The rescue robots in the Robocup Rescue League today are crawler-mounted vehicles with some simple gripper. They share a problem with robots from the @Home League: Their gripper is too big, so objects they have to get must stand sufficiently apart. This is often not given in a human environment, neither in rescue situations nor at home. One of the causes why the grippers are so big is that the motors for controlling them are directly inside of them.

A solution of this problem is to move the actuators out of the gripper into the arm moving it via tendons. But with the nonlinear characteristics of tendons the controlling of the grippers gets more complicated.

The human body has quite a few joints which have more than one dimension, like the hip or the shoulder. In most of the actual robots they are build out of three motors which are assembled as close together as it is possible, but the axis of the joints are not at the same spot as it would be with a real ball joint.

Another big problem for the humanoid robots is falling. By now humanoid robots have their motors directly in the joints. If they are falling, the kinetic forces act directly on the gear of the motor. These forces are strong enough to destroy the gears, which leads to failing motors and expensive repairs. Possible countermeasures are to add springs on the motor horn, [7] add a flexible coupling, ore use elastic tendons. Each of these countermeasures has some disadvantages on their own. I want to research tendons because it is the most human like form.

As the robots get more and more degrees of freedom, it gets more complicated to do the inverse kinematics. Elastic parts would be harmful on the mostly static kinematic calculation.

1.1 Related Work

As described above, the Robocup team WF-Wulfs has published a paper about adding springs to a joint [7]. They have implemented an additional gearbox on top of the motor which has a spring and a positional sensor inside. They have tried them in the competitions in 2014 but they do not seem to be ready yet as they do not write about them any more [4].

The problem of calibrating the kinematic model of the robot is discussed by Kastner et al. [13]. They used the camera and some visual calibration pads mounted beneath the feed. They state that the results are not perfect, but usable and better than without calibration.

There are some robots using tendons for some parts, mostly in the hands for example the iCup Robot has a nine DOF hand, operated via tendons [5]. Some joints are operated by a single tendon in cooperation with a return spring, the other ones have two tendons, but also only one motor. The iCup uses tendons in some other joints [10]. The tendons are steel cables with mostly one motor for two tendons.

The shadow hand [3] is operated by air muscles and tendons. Air muscles have the problem that they are limited to a maximum of 30 percent contraction. They can exert forces up to 70 kg at 4 bar. The problem with this system is that it needs a big pressured air supply, and in compression to motors the muscles are relatively slow. The actual position of the muscle is not as easy measured as in a traditional servo.

1.2 Thesis Goals

The goal of this thesis is to find a simple way to control joints moved by tendons with more than one motor and tendon per joint. I want to investigate if it is possible to build a self-learning application, which can operate the joint. Maybe the developed concept is able to regain precise control if the tendons length has changed after a repair or a heavy workload.

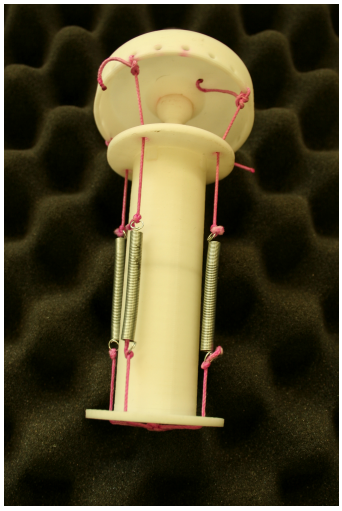
Chapter 2

Basics

In this chapter the basics used later in the approach are described.

2.1 Humanoid Robots

Most humanoid robots have a relatively low degree of freedom (DoF), so there are many movements they can not do. A typical humanoid robot has between 19 and 24 DoF. A human has 244 DoF in 230 joints controlled by 630 muscles [15]. For an idea about the major humanoid joints see figure 2.1.



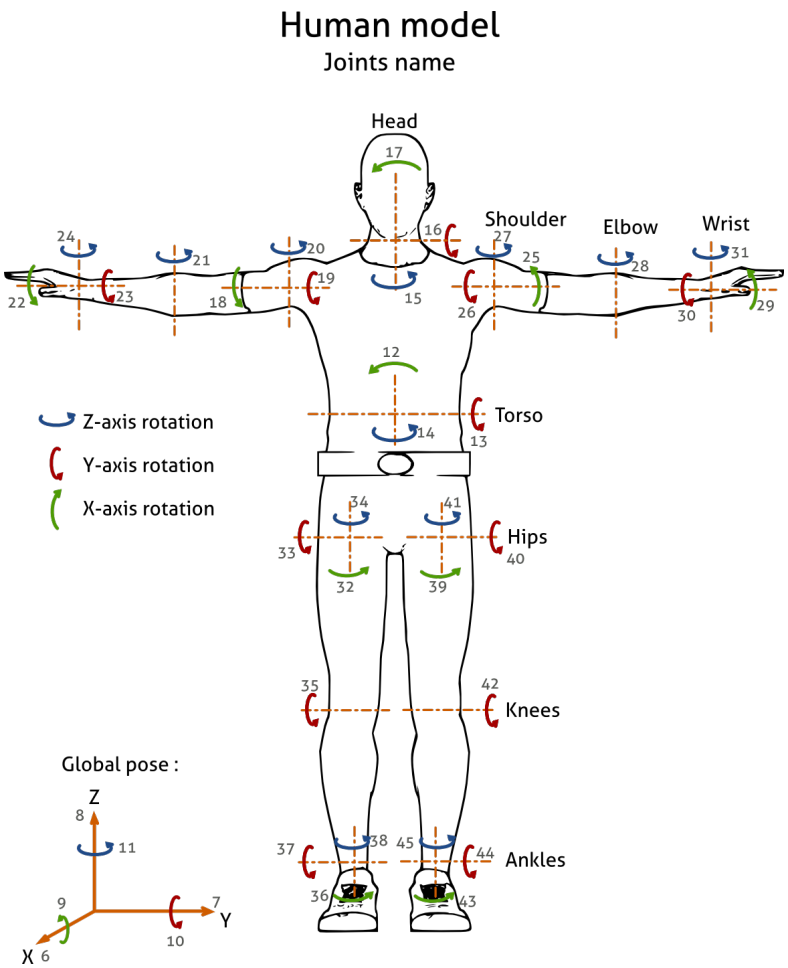
This is a model for a ball joint, operated by four tendons.

Figure 2.2: A model for a ball joint

2.1.1 Problems with Motors in Joints

Nowadays in humanoid robots motors and joints are basically the same thing [4] [2]. This is mechanically simpler, but have a few drawbacks, for example the large forces on the gear if the robot falls down. In multi dimensional joints like the hip there is also the problem that the three motors can not be on the same spot so the three movement axis are often slightly off, and not at the same point as in a ball joint. A ball joint can not be directly operated by motors, there have to be some tendons or bars.

A two or three dimensional joint can be done with motors, but the motors are reaching out of the object, often creating a problem because of collisions with other parts of the robot(see figure 2.4). In humanoid robots it is often not possible to do this in the shoulders, making the arm movement more difficult.



This figure shows a simplification of the joints in the human body. The 34 major joints are shown, the hands and feet are not shown in this image, so for gripping or advanced walking some more joints are needed. [Picture Source[1]]

Figure 2.1: Major joints in the human body

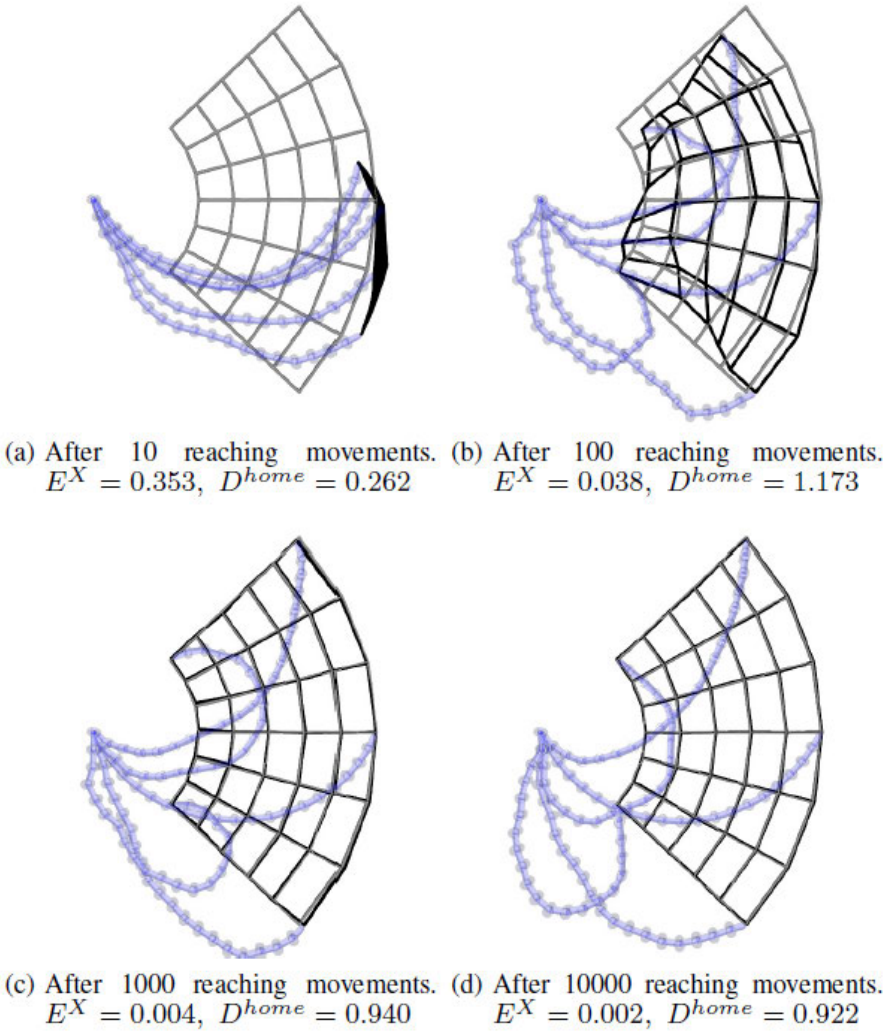
2.2 Kinematic and Inverse Kinematic

For most robotic control tasks the kinematic or inverse kinematic is used. The so called forward kinematic is used to calculate the actual position for a specific part of the robot, for example calculating the position of a hand or the position of the camera in the head. The inverse kinematic is used for calculate the necessary motor values to reach a desired position.

2.3 Motor Babbling

Motor babbling is a method to let a robotic arm explore and learn its workspace by itself [12]. For that purpose the algorithm tries randomly chosen values for all servos, and saves the correlation of the servo angles and the external survived

end effector position. After a lot of time the algorithm will have tried all possible servo angles and will know its workspace and in which way to reach a position.



(a) After 10 reaching movements. $E^X = 0.353,\ D^{home} = 0.262$

(b) After 100 reaching movements. $E^X = 0.038,\ D^{home} = 1.173$

(c) After 1000 reaching movements. $E^X = 0.004,\ D^{home} = 0.940$

(d) After 10000 reaching movements. $E^X = 0.002,\ D^{home} = 0.922$

Figure 2.3: Example for bootstrapping the inverse kinematic for a 20 DoF arm with online goal babbling [9]

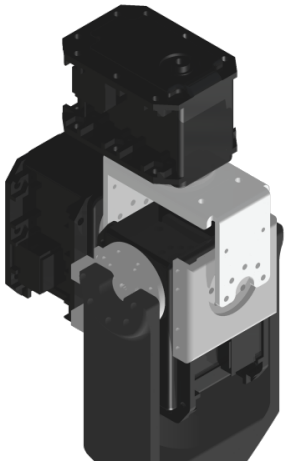


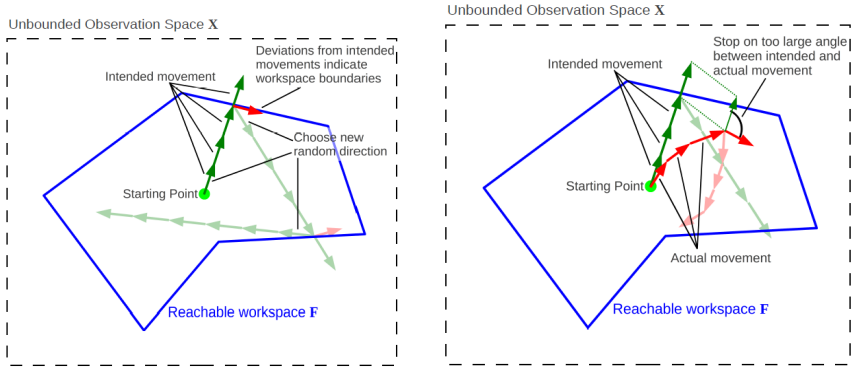
Figure 2.4: Typical motor positions for a hip joint

The first problem here is that, with a huge dimensional robotic arm, it takes a very big number of attempts, and therefore a lot of time to do training. The second problem is that it is possible to learn many sets of servo angles for the same end position, and the algorithm has to make a decision which is the best for using later on.

2.4 Goal Babbling

According to Hofstein [14] children try to reach for an object, even if they fail numerous times. This leads to

a rethinking of the simple motor babbling to overcome the limitations. Aiming for a goal instant of all possible motor angles like with motor babbling can reduce the dimension of the learning problem, for example from a ten degree of freedom arm to a two dimensional goal. In [8] Rolf et al. present a approach to learn the inverse kinematic with goal babbling. This approach was refined with online learning by the same authors in [9]. In this paper they showed that it is possible to learn the inverse kinematic in a reasonable time (see figure 2.3). In 2013 Rolf had refined the goal babbling to work with unknown ranges [11]. This is done while evaluating the desired movement against the actual movement. If the two differ too much, it is assumed that the range of the robotic arm is reached in this direction (see figure 2.5).



(a) Deviations from intended movements indicate boundaries.

(b) During learning, more deviations need to be accepted.

Figure 2.5: Detecting the boundaries of the workspace [11]

2.5 Neuronal Networks

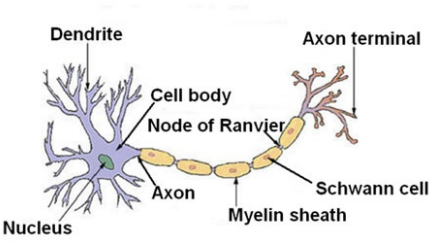


Figure 2.6: Model of a biological neuron
The dendrites are the input, the axiom the output Source: wikipedia.org

One technology for learning algorithms in artificial intelligence are Artificial Neuronal Networks (ANN). They are biologically inspired by the neurons in the human brain (see figure 2.6).

A biological neuron as explained by Hebb [6] has numerous electrical inputs called dendrites. If the accumulated input reaches a certain level, the cell is emitting an electrical impulse via the axon.

2.5.1 Single Layer Perceptron

The simplest form of a neuronal net is the Single Layer Perceptron (see figure 2.7), in which some input neurons are connected to some output neurons.

The value of each neuron is determined with the following formula:

$$o_i = f\left(\sum_{j=1}^{N} w_j x_j\right) \tag{2.1}$$

The output value o_i of a neuron is the sum of the weighted outputs from the neurons which are connected from the previous layer, w_j are the weights of the edges, x_j the output of the previous neuron. We use a function f for determining the activation of the neuron, so it is possible to implement a threshold as in the biological neuron. If we have more than one output neuron o_i we do the calculation for each of them separately. It is possible to use a simple step function

$$f(n) = \begin{cases} 1 & \text{if } n \geq 0 \\ -1 & \text{if } n < 0 \end{cases} \tag{2.2}$$

The learning is done by adjusting the weights of the edges w_i. After each learning step the error produced by the actual weights δ_i is determined (see 2.3). There t_i is the desired output of the neuron. The result is multiplied with the learning rate α and the output of the previous neuron x_i, and added to the actual weights. (2.4)

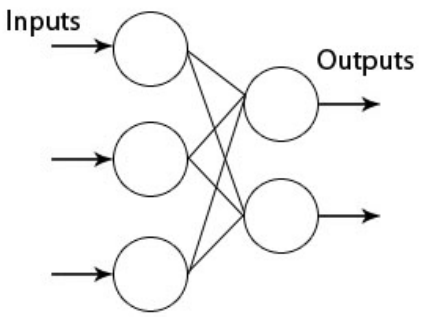


Figure 2.7: Single layer perceptron

$$\delta_i = t_i - o_i \tag{2.3}$$

$$\Delta w_i = w_i + \alpha\, \delta_i\, x_i \tag{2.4}$$

With a Single Layer Perceptron it is possible to model the logical AND or OR functions, but a SLP cannot model linear inseparable functions like XOR.

2.5.2 Multi Layer Perceptron

Multi Layer Perceptrons (MLP) can in contrast to SLPs handle linearly separable functions. A MLP has at least one hidden layer between the input and output (see figure 2.8) The neurons in a MLP are calculated like in the SLP, but calculating each layer from input to output separately. The output of the neurons is calculated with formula 2.5 and 2.6.

$$o_j = f(\text{net}_j) \tag{2.5}$$

$$\text{net}_j = \sum_{i=1}^{n} x_i w_{ij}. \tag{2.6}$$

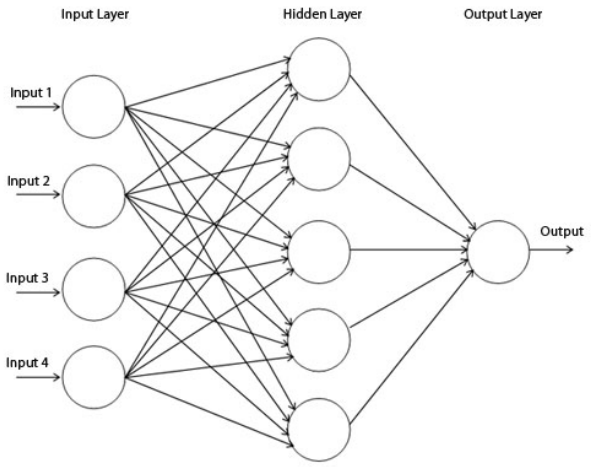


Figure 2.8: MLP

In a MLP the learning is done by back propagation. For the output layer the error can be calculated directly, for the hidden layers it is more complicated because the error of the neuron is not easily determined. Because the back propagation uses the derivative the activation function has to be differentiable. Usually sigmoid functions like 2.7 or 2.8 are used, the former ranging from -1 to 1, the later from 0 to 1.

$$f(n) = \tanh(n) \tag{2.7}$$

$$f(n) = \frac{1}{1 + e^{-n}} \tag{2.8}$$

For the update of the weights a gradient descent (see 2.9) is used. w_{ij} is the weight of the edge from neuron i to neuron j. E is the error function 2.10

$$\Delta w_{ij}(n) = -\alpha \frac{\partial E(n)}{\partial v_j(n)} y_i(n) = \alpha \delta_j x_i \tag{2.9}$$

$$E = \frac{1}{2} \sum_{i=1}^{n} (t_i - o_i)^2 \tag{2.10}$$

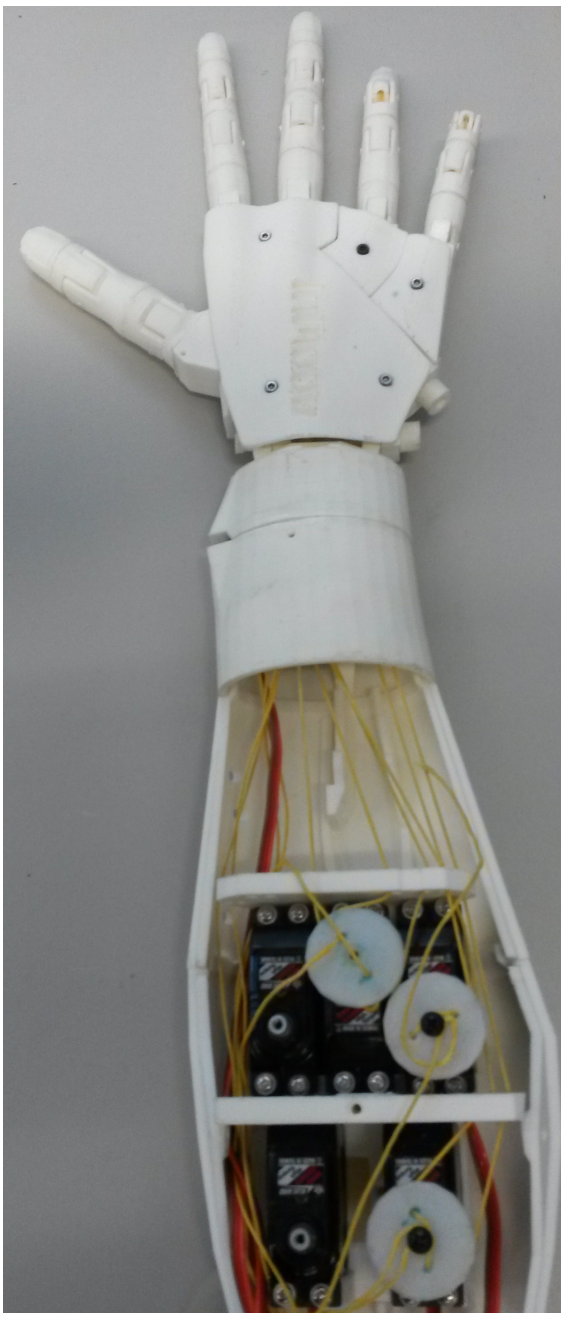
All together we get the formula (2.11) for the error of each neuron.

$$\delta_j = \begin{cases} f'(\text{net}_j)(t_j - o_j) & \text{if } j \text{ is a output neuron} \\ f'(\text{net}_j) \sum_k \delta_k w_{jk} & \text{if } j \text{ is a hidden neuron} \end{cases} \tag{2.11}$$

2.6 Tendons

Some of todays robots already have tendons, but often they use only one motor for the two tendons from one joint. Humans generally have one muscle for each tendon. Using one motor is done because with one motor the controlling gets easier, and it needs less space. The downside is that after a lengthening of the tendons there is some play in the motor movement, which most of the time is unwanted as it permits the movement of the effector without motor movement.

The human tendon muscle combination is a little elastic, so what not every fall leads to serious injuries. In robotics elastic materials are mostly avoided because the controlling with them is much more complicated, and often not yet possible.



It is visible that for each motor there are two joints. The Two left Motors are not jet connected.

Figure 2.9: The Inmove Hand

Chapter 3

Hardware and Software

3.1 Motors and Sensors

The testing model (see 3.2) uses the Dynamixel MX-28 servo motors from Robotis. They are digitally controllable over a simple protocol on a TTL bus system. The motors have a hall effect sensor and a magnet on the joint axis as a position encoder. The sensor has a resolution of 4096 positions on 360 degree, resulting in a resolution of 0.088 degrees. They understand a variety of commands, for example switch off the torque.

Figure 3.1: MX-28 Servo

A MX-28 motor with switched off torque is used as a position sensor in the joint because it is simple to get the current position from the motor over the framework (see 3.3) from Hamburg Bit-Bots, a Robocup team playing in the humanoid soccer league. For other positional sensors it would be necessary to implement the whole communication with the sensor.

Experience from the Hamburg Bit-Bots shows that the MX-28 motors have some play if they are switched to hold a position. If the motor is aged it can go up to several degrees. This positional offset can be read from the internal position sensor, but the motor is not able to hold the position that exact.

3.2 Testing Model

For the practical tests a simple model of a one dimensional joint with two tendons was build. The model is 3D printed, with some metal parts which are originally taken from the Darwin OP robot, but it would be possible to print them too. For the communication with the motors the USB2Dynamixel connector was used. It connects the motor bus from the Dynamixel to the USB port of any computer.

Plaited fishing line is used as tendons, because they are relatively robust not so elastic and simple to handle. The fishing line have a diameter from 0.8 mm and resists a force of 900 N.

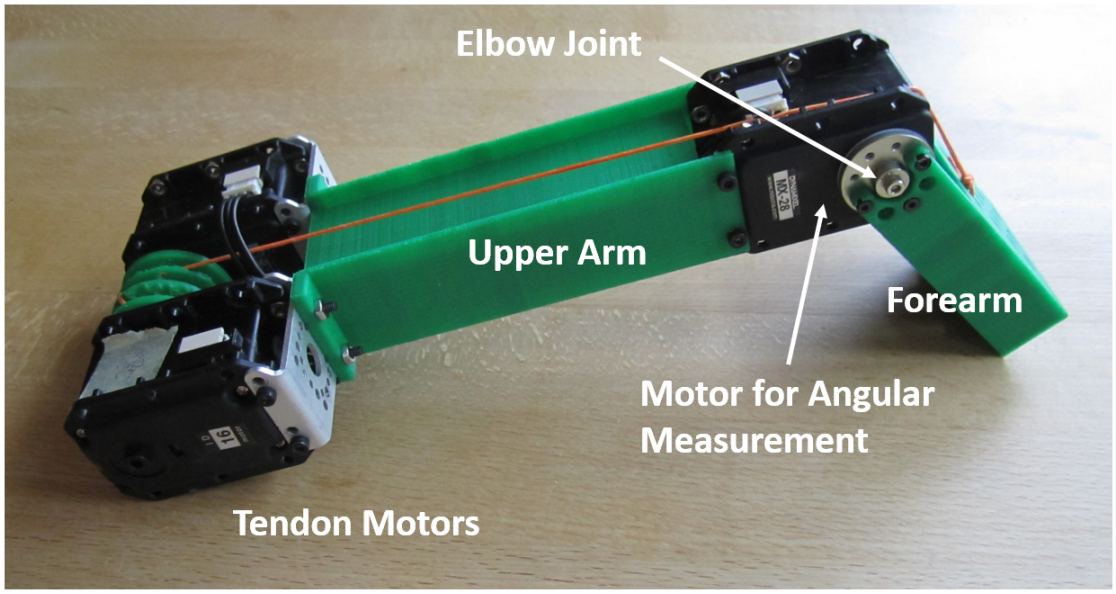


Figure 3.2: The testing model

3.2.1 Lengthening of the Tendons

Plaited fishing line have a relative low lengthening in comparison to other fishing lines. Measurements with the used fishing line have shown that it lengthen ca. 1.3% under 15 kg weight. A 30 cm part with a Bowline knot at one end lengthening to 32 cm, meaning a over all lengthening of 6%. The bowline was used in the model to fasten the fishing line to the forearm.

3.3 RoboCup Framework of the Hamburg Bit-Bots

The motor control software was written in Python using parts of the above mentioned Hamburg Bit-Bots framework [1] for the communication with the hardware over the serial bus. Hamburg Bit-Bots uses the Darwin-OP robot, which uses the MX-28 servos. Therefore the software from them already have the support for controlling MX-28 servos.

The software makes it easy to read and write the registers of the motors. For example it is not necessary to write the two bytes of the goal position separately. Instead the framework lets you write the goal position in degrees. In case of an error a appropriate Python exception is thrown, so error handling is quite easy to do.

3.3.1 Extensions to the Software

A simple wrapper for the hardware was written, so that the controlling algorithms can set positions to the tendons and read back the resulting angle from the joint in a simple matter. There are some classes for the different controlling and test algorithms.

Chapter 4

Approach

4.1 Implementation

All following approaches have in common that they have two motor commands as output and get the actual angle of the test joint back. The commands to a motor could either be a angle or the command to disable, enable or set the speed. At the beginning of a test an initializing sequence is done, including to set the joint on the left end of the reachable moment space and tightening the tendons.

4.2 Creating a Control Table

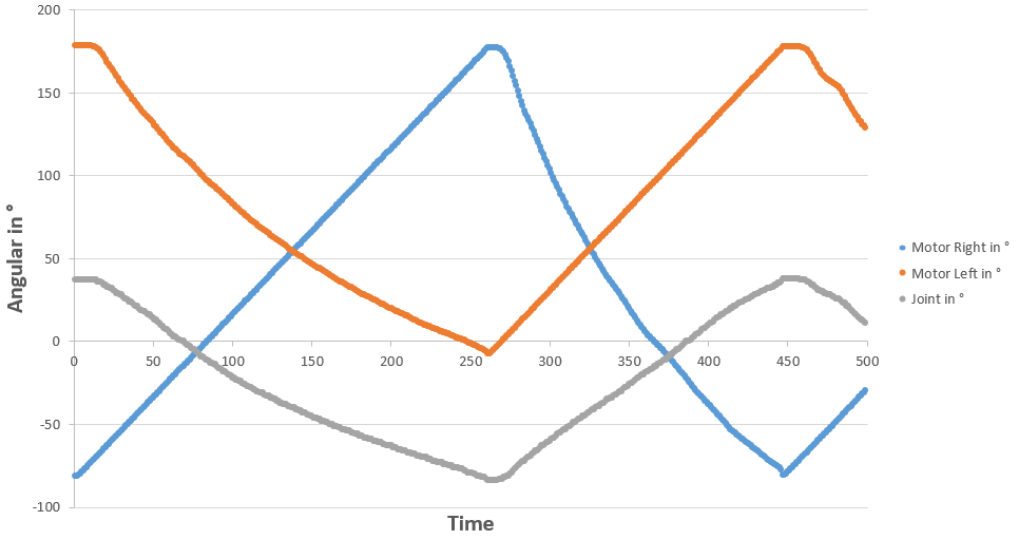
If the torque of one motor is switched off and pulled with the other motor, it is simple to record the positions of the switched off motor and the joint. At the end of the reachable angle the roles of the motors are changed, letting the other motor pull the former active motor (see figure 4.1). After some iterations the average over the recorded positions are calculated. A mapping from a joint angle to the corresponding two motor angles as seen in figure 4.2 is the result. Each motor position was taken one hundred times, the active motor there moved in one degree steps to minimize the recording time.

4.2.1 Limitations

This method can get a control table fast, but there are some drawbacks with it. At first the resolution is limited (see figure 4.3) because the pulled motor and the attached tendon do not have the same drag at all positions.

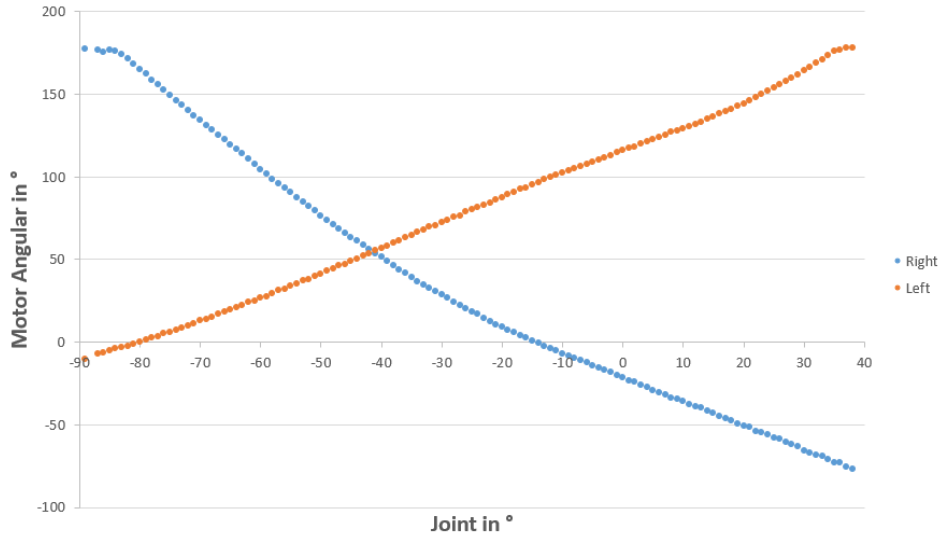
The peak in the error values on the left side of the control spectrum are most likely an effect of the slightly elastic plastic used in the model and the unequal length changes in the edge of the reachable room.

Over the time in use, the tendon and especially the knots lengthen (see chapter 3.2.1) and the table gets more and more inaccurate. To compensate this it would be necessary to record the tables again.



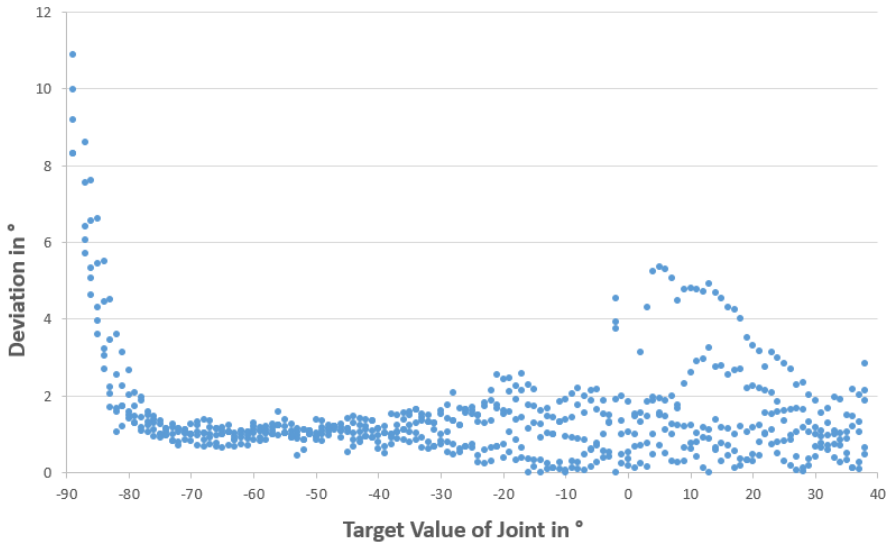
The graph shows the raw data of the servos and the joint. The segment where a motor line is straight corresponds to the time the motor pulled the tendon, in the other time the torque was switched off.

Figure 4.1: Part of the motor to joint angle data



This graph shows the angles the motors have to head for to archive a desired joint angle. The data was generated with an average over ten cycles.

Figure 4.2: The joint to motor angle table



This graph shows the error on different tries to reach the recorded points from the control table. The shown value is the absolute error in degrees.

Figure 4.3: The joint error table

If the joint has to operate in an up down direction at the time the tables are recorded, there is an additional problem: The moving part is falling down if the holding tendon is set to no torque (see figure 4.4).

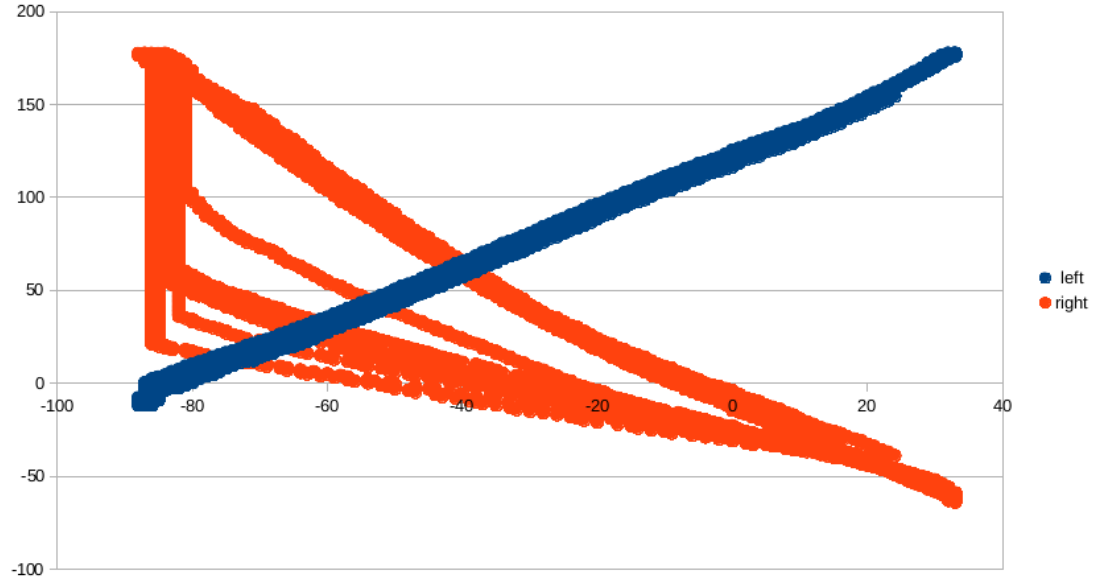
4.3 Goal babbling

As goal babbling is an approach with no initial knowledge about the robot itself (see chapter 2.4), it has to discover the possible movements by itself. It gets the space it has to explore, but in this case there are many motor positions which are not possible. In the original goal babbling there are static limits on each motor so that the robot does not collide with itself.

In this case however the limits change dynamically with each motor command, and even worse a movement of one motor can require a certain minimal movement from the other motor, or the possible movements would be greatly limited.

An attempt for a fix of the problems is to switch one motor off, like in the previous attempt, and let one motor pull the other. This approach has the problem that it has to determine which motor to switch of. This decision is not simple because it is possible that the algorithm decide to pull on both tendons at the same time.

Another problem with this fix is that the goal babbling would not learn to control both motors at the same time. It will record the pulled position like in the control table, or even worse, if not taken special care of in the implementation the



In this graph the raw positions for the motors are plotted against the joint angle. Each position was reached ten times. The right servo was the upper servo in this experiment.

Figure 4.4: Data recording of the controllable if the joint is vertical

position originally wanted by the algorithm.

An other idea is to represent both motors as one joint in the goal babbling algorithm, but as the motors do not act linearly to each other it is not simple possible to create a mapping.

4.4 MLP

Trying to bootstrap a MLP (see 2.5.2) with no prior knowledge has very similar problems to goal babbling. The untrained MLP just tries different motor positions, this again leads to motors which both try to pull on the tendons. It would be possible to train the MLP if the force in the tendons is measured the whole time, and then used as an error value for the MLP training. If the force is going to get too high the actual test would be aborted and a large error would be reported back to the MLP. This was not tested because sufficient force sensors are quite expensive and would need hardware modifications as well as a lot of new controlling software to communicate with the sensors.

4.5 Pre Trained MLP

As bootstrapping the MLP does not work on the current hardware without risking serious damage, the MLP was trained on the existing data from the first approach (see chapter 4.2). After the initial training the MLP was used to control the

model. Then the MLP was switched in online learning mode, by continuously back propagating the reached and given values. As an overall error value all errors from the training set were summed to get an easy to read overall error value. This value should be minimized.

4.5.1 Initial Training and Setup

The MLP was trained on the data from the first approach (see 4.2). For this the joint angle was taken as an input and each motor angle as an separate output. The formula (2.7) was used as activation function. The input and output values of the MLP are normalized from the original range of -180 to 180 to a range from -1 to 1. After some evaluation (see figure 4.5) as there are not much differences between the hidden neuron count five hidden neurons and 20000 iterations initial training where chosen. The learning rate α was set to 0.1. The learning rate was chosen because 0.3 is beginning to oscillate.

A test with the pre trained MLP lead to positional errors which are a little more distributed, but it seam to be at largely similar to the table approach from chapter 4.2 (see figure 4.6).

4.5.2 Online Learning

The trained MLP can control the model with nearly the same precision as the control table approach. For the online learning the MLP was used to calculate the necessary motor values for each joint angle. After the positions were reached, it was tried to reach the exact joint value switching one of the motors off and moving the other slowly. The difference between the positions from the MLP and the corrected position was then used as training error. This lead to slowly improving the errors.

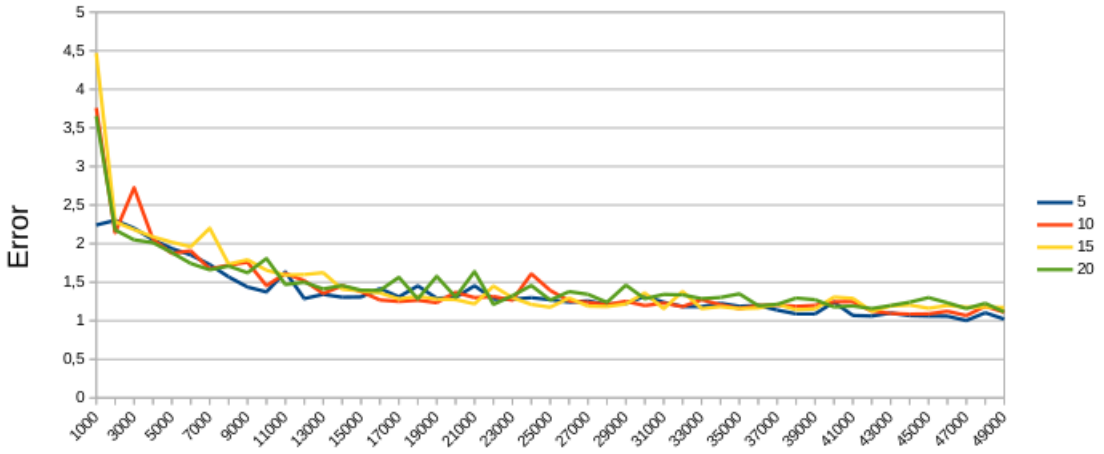
4.5.3 Change of the Model

For testing a change in the tendon length in a controlled manner an offset were put in one of the motors. There one degree offset roughly translate to a one millimeter longer tendon. Offsets below five degrees were compensated in under one-hundred movements for each position.

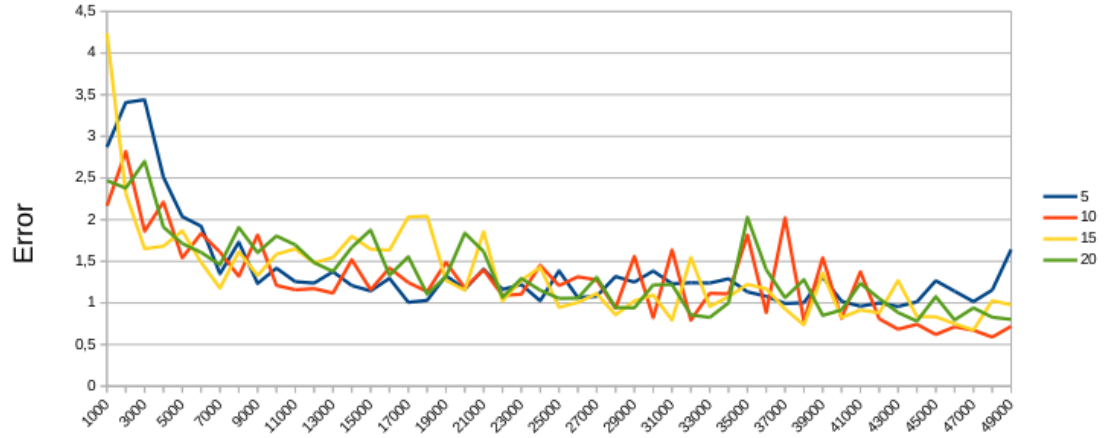
I only tested the lengthen of a tendon because if they shorten the motors would work against each other without the MLP being able to notice it in any way. This would not be good for the hardware.

4.6 General Limitations

It is necessary to use small steps in giving new positions to the motors. A jump from one end of the controllable space to the other one will command the motors to reach the indented position as fast as possible. Within that time there is the possibility



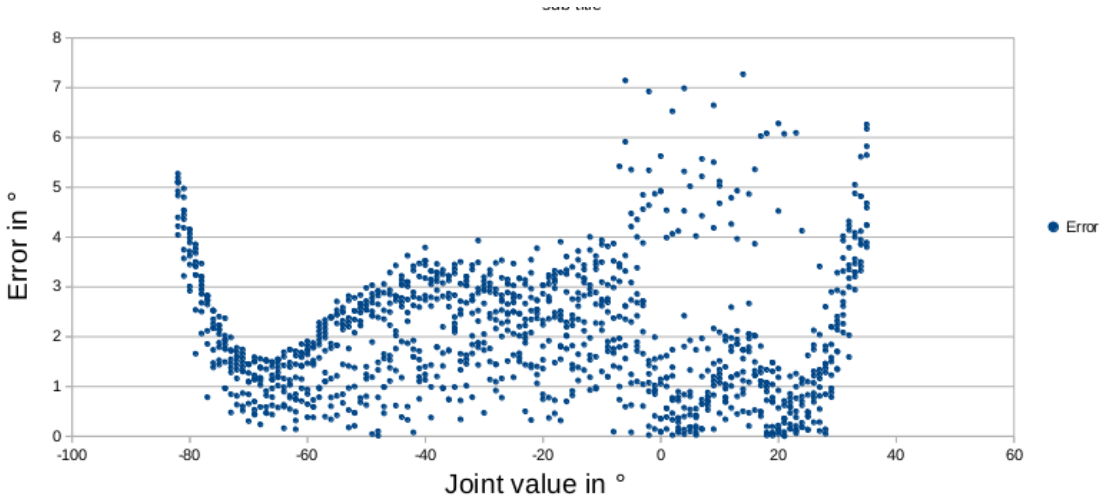
This graph shows the error with a learning rate α of 0.1 for MLPs with 5, 10, 15, or 20 neurons in the hidden layer



This graph shows the error with a learning rate α of 0.3 for MLPs with 5, 10, 15, or 20 neurons in the hidden layer

Figure 4.5: Comparison of different MLP configurations

that both motors together put too much tension on the tendon or loosening it too much. Both outcomes are not good, the former is bad for the motors themselves, because of the forces, the later is not so good because of the loose joint.



The MLP was used without online learning on the real hardware. it had to reach random chosen positions

Figure 4.6: The errors of the pre trained MLP

Chapter 5

Conclusion and Discussion

5.1 Discussion

5.1.1 Inaccurate Control Table

With the used approach for generating the control table there was a relatively big error in some regions for reaching the position. For some applications this might be enough accuracy, for most robotic arms it would not be sufficient. Fingers for example do not have to be this accurate, because at gripping it is mostly more necessary to know the force with which the object is hold.

After some time the MLP had reduced the inaccuracy so the setup can be used for more applications. This approach will need some time at the beginning for each joint to learn, but after some time the accuracy is good enough for most applications. The problem will be that each joint have do be trained separately, most likely after each other to avoid collisions with other parts of the robot.

5.1.2 Bootstrapping a MLP

As stated in 4.4 it would probably be possible to bootstrap a MLP without prior knowledge if additional sensors are integrated in the model. But the count of training iterations in 4.5 is a good hint about the necessary training time. One movement from one side to the other and back again needs round about 10 seconds. With the recorded training data at least 5000 iterations were necessary for reaching a reasonable error rate. This would mean a training time of about 14 hours, not calculated in that the hardware can not operate this long at once. Also the training would probably need more iterations, because in the beginning the higher output angles are not reached.

5.1.3 Use in a Real Arm

In a robotic arm it would be possible to use tendons controlled with the MLP from this thesis, but if a highly precise positioning is required, for example for

drilling holes, the positions have to be controlled externally or the accuracy of the controlling have to be a lot better than in the experiments.

One could say that the human is not capable of putting his joints in exact angles and control the arm mostly over visual feedback when gripping something. Dynamic controlling, like humans mostly do, would be possible if the controller just works with commands like five degrees more in the elbow joint. Then the main task of the MLP or control table would be to move both tendons in a way that they interact smoothly. This task could be done from the MLP, the control table have some points there the accuracy of the joints to each other is not this good, but this would be manually fixable if it is necessary.

5.1.4 Use in general Robots

If elastic tendons are used in the legs the walking algorithms have to be adapted, because at the moment they control they needs to control the positions very exact, as the whole kinematic and centre of mass needs to be computed. But in general there is nothing especially against tendons in the legs. The control algorithms has to get more accurate, and the walking has to deal with more inaccuracy.

5.2 Conclusion

The pre trained MLPs with online learning can be used in some environments, but for the most use cases it would need to include force sensors in the tendons for more accuracy and easier training. Goal babbling is a good approach for learning the reachable environment, when the joints are simply controlled by angles. It would be possible to use goal babbling for bootstrapping the kinematic of an arm using the MLP driven joints.

This thesis showed that it is not this simple to control joints driven by elastic tendons with hight accuracy. As discussed above for some use cases the reached accuracy is enough but there are a lot use cases where more accuracy is needed. Another drawback for the learning algorithms is the high learning time on real hardware.

5.3 Further work

A huge improvement for controlling a joint with tendons would be to include force sensors within the tendon. With force sensors it would be possible to make sure that the motors are not working against each other. It would be possible to generate more accurate control tables as it is possible to tighten the tendon which was not controlled in this experiments.

To further improve the shock resistance when the robot is falling down, more elastic tendons could be used, or springs could be included in the tendon. This would make the controlling even more complicated, but together with the force sensors the robot could tense the joint for more exact positioning or let it loose.

Another area of work is to expand this thesis to a multidimensional joint. Therefore it would be necessary to build a sensor which can be integrated in a multidimensional joint. The simple solution to take a motor as sensor, as done here, is not possible then. Another problem would be that the space of possible positions for all tendons is much larger, and the exploration and learning would cost a lot more time. Therefore it would be necessary to try to speed up the data recording and learning.

For the human hand there is the problem that the tendons for the fingers go over the wrist, the muscles are in the upper arm. This means that each movement in the wrist has to be countered in for the fingers, creating depending movements and again a higher dimension in the controlling problem.

Bibliography

[1] Hamburg Bit-Bots. Hamburg bit-bots code release 2013.

[2] Jan Draegert Simon Gene Gottlieb Roman Schulte-Sasse Gregor Barth Malte Detlefsen-Jan Bernoth Niklas Rughft Michael Pluhatsch Martin Wichner Daniel Seifert, Lutz Freitag and Ral Rojas. Berlin united - fumanoids team description paper for robocup 2015. 2015.

[3] Paul Tuffield Hugo Elias. The shadow robot mimics human actions. *Industrial Robot: An International Journal, Vol. 30 Iss 1*, pages 56–60, 2003.

[4] Oliver Krebs Reinhard Gerndt Stefan Krupop Tobias Bolze Frank Stiddien, Minda Xia and Tom Lorenz. Wf wolves & taura bots - humanoid kid size team description for robocup 2015. 2015.

[5] David Vernon Lorenzo Natale Francesco Nori Giorgio Metta, Giulio Sandini. icub: the open humanoid robot designed for learning and developing complex cognitive tasks.

[6] D. O. Hebb. The first stage of perception: growth of the assembly. *The organization of behavior: A neuropsychological theory*, pages 60–78, 1949.

[7] Reinhard Gerndt Leandro Tome Martins, Roberta de Mendonca Pretto and Rodrigo da Silva Guerra. Design of a modular series elastic upgrade to a robotics actuator. 2014.

[8] Michael Gienger Matthias Rolf, Jochen J. Steil. Goal babbling permits direct learning of inverse kinematics. *IEEE transactions on autonomous mental development, vol. 2, no. 3, september 2010*, 2010.

[9] Michael Gienger Matthias Rolf, Jochen J. Steil. Online goal babbling for rapid bootstrapping of inverse models in high dimensions. *IEEE Int. Conf. Development and Learning and on Epigenetic Robotics*, 2011.

[10] G. Sandini D. Vernon R. Beira F. Becchi L. Righetti J.Antos-Victor A. J. Ijspeert M.C.Carrozza N. G. Tsagarakis, G. Metta and D. G. Caldwell. icub: the design and realization of an open humanoid platform for cognitive and neuroscience research. pages 1151–1175, 2007.

[11] M. Rolf. Goal babbling with unknown ranges: A direction-sampling approach. *Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–7, 2013.

[12] Metta G. Sandini G. Saegusa, R. and S Sakka. Active motor babbling for sensory-motor learning. *IEEE International Conference on Robotics and Biomimetics (ROBIO2008)*, 2008.

[13] Tim Laue Tobias Kastner, Thomas Rfer. *Automatic Robot Calibration for the NAO*, pages 223–244. Springer International Publishing.

[14] Claes von Hofsten. Eye-hand coordination in the newborn. *Developmental psychology*, 1982.

[15] Vladimir Zatsiorsky. *Biomechanics of Skeletal Muscles*.

Erklärung der Urheberschaft

Ich versichere an Eides statt, dass ich die Bachelorarbeit im Studiengang 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 auf dem elektronischen Speichermedium entspricht.

Ort, Datum Unterschrift

Erklärung zur Veröffentlichung

Ich erkläre mein Einverständnis mit der Einstellung dieser Bachelorarbeit in den Bestand der Bibliothek.

Ort, Datum Unterschrift