Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

DHEST
Department of
Health Sciences and Technology

Sensory-Motor Systems Lab

Steve Maassen

# New training strategies and experimental tasks to enhance motor learning

**Bachelor Thesis**

Sensory-Motor Systems Lab
Swiss Federal Institute of Technology (ETH) Zurich

**Supervision**

Dr. Laura Marchal-Crespo

**Thesis Advisor**

Prof. Dr. Robert Riener

Spring Term 2015

# Declaration of Originality

I hereby declare that the written work I have submitted entitled

**New training strategies and experimental tasks to enhance motor learning**

is original work which I alone have authored and which is written in my own words.[1]

**Author(s)**

Steve Maassen

**Supervision**

Dr. Laura Marchal-Crespo

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`http://www.ethz.ch/students/exams/plagiarism_s_en.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date

Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

# Acknowledgements

I want to thank my supervisor Laura Marchal-Crespo for the professional and kind supervision during the whole Bachelor Thesis. Further I want to thank Kilian Baur for his kind support and advice he gave in any situation we were stuck due to technical problems. Also, I want to thank Tanja Baumann for the well functioning cooperation in order to combine both Bachelor Thesis'.

# Contents

# Abstract

Having done a review of literature in the field of motor learning, especially analyzing the effects of error amplification and haptic guidance on timing and spatial aspects, it became clear that guidance is favorable for timing aspects whereas resistant feedback improves spatial components. In order to test this hypothesis, a good test environment that would keep the motivation of the patients up by distancing them from the idea of fulfilling a dumb tracking task, was needed. With the game developer environment and game engine Unity, an "Ice Cream Maker" game was implemented with three different tasks and all the adjusting options needed. After that an assisted-as-needed controller, with multiple decaying rules, was implemented in Simulink for the upper limb exoskeleton robot ARMin. The communication between the robot and the game was established with the UDP-Protocol. Functioning of the assisted-as-needed controller in combination with the game was tested. Results were satisfactory, but the exact values for the decaying rules should be defined by the criteria of a possible future study later on.

# List of Figures

# List of Tables

# Symbols

## Symbols

| | |
|---|---|
| $k_i$ | proportional gain for the coordinate $i$ [-] |
| $d_i$ | differential gain for the coordinate $i$ [-] |
| $G_i$ | percentage of initial gain value at time step $i$ [-] |
| $\delta$ | decreasing factor [$s^{-1}$] |
| $f$ | robot forgetting factor [-] |
| $g$ | robot learning gain [-] |

## Acronyms and Abbreviations

| | |
|---|---|
| AAN | Assisted-As-Needed |
| DoF | Degrees of Freedom |
| EA | Error Amplification |
| HG | Haptic Guidance |
| NA | Noise Amplification |
| SMS | Sensory Motor Systems |

# Chapter 1

# Introduction: Studies on Mechatronics

## Skill and Tasks Specific Control Methods to enhance Motor Learning, A Review

## 1.1 Introduction

To improve motor learning of stroke patients or even healthy subjects, many strategies have been considered. It seems logical that you have to provide some kind of feedback to the subject in order to make him/her learn a novel task. This is the reason why many studies have analyzed different methods of augmented feedback, namely visual, haptic and auditory feedbacks in many variations and combinations. However, in most instances, these studies did not take into account the skill levels of the subjects or the nature of task (e.g. discrete or continuous). Although there are some reviews on the benefit of different feedback strategies on motor learning [14] [22], nobody tried to summarize the results of different strategies in function of the subjects' skill level and task specification. Specific tasks characteristics seem toplay an essential role in learning a task. One of the biggest distinction made by the community is the one between temporal and spatial aspects. This is due to the fact, that neurological investigations show that temporal aspects and spatial aspects are handled by different regions of the brain [8]. In this chapter, I will try to recapitulate the latest results and come up with efficient feedback methods in a specific task. Due to the fact that augmented feedback in general makes up a huge field of research, I will focus mainly on haptic feedback and look into the various control strategies proposed by the experts in this field.

## 1.2 Main Concepts in Haptic Feedback

Haptic feedback in motor learning has been studied for several years now. It consist in a robot-human interaction in which feedback is provided from the robotic device to the human. Especially Robotic-assisted-rehabilitation has come up more and more in the last years due to the increasing level of technology and the need for therapeutical assistance. Looking at the statistics of The Internet Stroke Center, only in the US, each year, approximately 795,000 people suffer from a

stroke [1]. This causes a lot of physical demanding work for physiotherapists, which could be taken over by assisting devices. Therefore, the most logical way to replace the therapeutical feedback would be with haptic devices (e.g. robots) which can provide the same or even superior feedback to the subjects. Furthermore, haptic feedback is unique in due to a characteristic called bidirectional property of the haptic sense [22]. This is is used very often in the early days of a human being to learn the first movements because it enables us to interact with our surroundings and perceive informations from it at the same time. Something that is essential for the learning of every movement is, on the one side, the processing of the movements completed with help of the robotic device, and on the other side, the building of an inner model. The processing of the feedback information is very distinct in different cases, meaning that timing and spatial aspects are treated in different ways when you have to do a discrete movement or a continuous movement. [4] In the case of resistive feedback methods, model building is taking place because the subject feels where he is supposed not to go. To analyze all these findings in different cases and put them in the context of different control strategies, I will go through the main methods that were used in the last years and I will try to point out the different effects they generate on motor learning.

In general, there are many different strategies concerning motor learning with haptic feedback devices. Some of them have been shown to be beneficial in certain conditions while others showed no change or even detrimental effects. At a top level, you can assign nearly all haptic feedback methods to one of two main categories, namely assistive guidance or resistive haptic feedback also known as haptic disturbance. In the first case, the forces generated by the haptic device push the subject's concerned limb into the right position or inherit the wanted velocity. In the resistive case, exactly the opposite is achieved by generating forces that bring the subject away from the wanted position or velocity. In both categories, there are many different methods to generate the wanted effect, in which some of them showed beneficial in specific cases and impairing in other ones.

## 1.3 Resistive Haptic Feedback Methods

Resistive Feedback does not try to show how a certain movement or task is completed, but tries to prevent subjects from doing it. This is motivated through the fact that error drives the learning of motor tasks and is beneficial for building an internal model as shown by Thoroughman and Shadmehr [24]. Another thought behind this method is that the subject increases his/her ability to recognize small errors due to the amplified error feedback he gets [26]. However, it has been shown that this method is most beneficial with subjects who are of at least intermediate skill level [15]. To generate the forces that achieve these wanted errors to a certain desired position or velocity, experts in the field came up with different methods and algorithms, which I am going to discuss briefly in the following subsections.

### 1.3.1 Error Amplification

A common method to realize an error-augmentation setup is simple error amplification (EA). This is realized by measuring the error that has most meaning for the the specific task, and multiplying it by a negative gain, often called error amplification gain. A simple example of this method can be seen in equation (1.1) where $k > 0$ and $x_{des} - x_{act}$ corresponds to the task specific error measurement.

$$F_{mot} = -k \cdot (x_{des} - x_{act}) \tag{1.1}$$

[1] Internet Stroke Center - Statistics: http://www.strokecenter.org/patients/about-stroke/stroke-statistics/, last checked: 06.08.2015

A study by Reinkensmeyer et al. showed that EA was not beneficial for a discrete task consisting of a virtual golf putting movement with a wanted velocity profile [5]. They even found that it was decreasing the motivation of the subjects throughout the whole study, which is something that should be avoided because motivation plays a crucial role in motor learning. However, other studies, like the one of Powell et al., who looked into main control methods and applied them to two tracking tasks, showed that EA led to significantly better results than haptic guidance methods [3]. Another study by Lee and Choi, also about a 2D tracking task, found the same results [10]. Unfortunately there are only a few studies that take into account the initial skill level of the subject. A study of Milot et al. showed that EA was beneficial in a flipper like game, consisting of a discrete tasks, for initially skilled subjects [15]. Furthermore a therapeutical study by Cesqui et al., in which patients had to fulfill a 2D point to point movement, showed that, for patients with mild level impairment, EA was more beneficial than HG.

### 1.3.2 Noise Amplification

Another strategy to generate disturbing haptic feedback is noise amplification. In this case, the forces, generated to prevent the subject from reaching the desired position or velocity, follow a noise-like pattern and seem random to the subject. One of the few studies on that topic was the one of Lee and Choi, which consisted of a 2D continuous tracking task [10]. They compared a progressive form of HG, EA and a noise-like disturbance and found that the later method had significantly better results than HG in the retention trials. The reason for this is probably that the subject's attention is kept up throughout the whole experiment due to the random disturbances. This way the subject's response to errors and changes in the trajectory is more immediate. Unfortunately skill level was not taken into account in this study, because no baseline tests were run.

### 1.3.3 Beyond EA and NA

There are a few other methods to achieve a resistive haptic feedback, which follow more complex algorithms to generate the desired effects. One example is a method by Lee et al. who designed a hybrid controller that starts as a Guidance Controller and fades into an EA controller [9]. However, there was found no statistical significant advantage compared to normal EA or Haptic Guidance. Another study was done by Morasso et al. in which a mixture between guidance and resistance was implemented [16]. In figure 1.1 represents their design of a manifold that generates assistive forces in the normal direction and resistive forces in the tangential direction of the desired movement. They showed that, in the case of a discrete tracking task, the method clearly improved the building of the inner model, which, as we learned, is crucial for motor learning.

## 1.4 Haptic Guidance

To teach a new and complex task to a novice, it seems logical that the first step would be to show him/her the task by manipulating his concerned limb and reproducing the desired movement. That is, basically, the main idea behind assistive guidance by robotic devices, especially considering subjects that have some kind of severe impairment and are not able to fulfill the wanted task by themselves. However, haptic guidance is discussed controversially and after many years of research, some experts showed that it might even impair motor learning. One of the main arguments is the guidance hypothesis. It states that the subject that is supposed to learn the task with assistive guidance, relies too much on the feedback and becomes inactive. After the learning phase, the subject's performance in retention tests could be even worse than before because of the passive

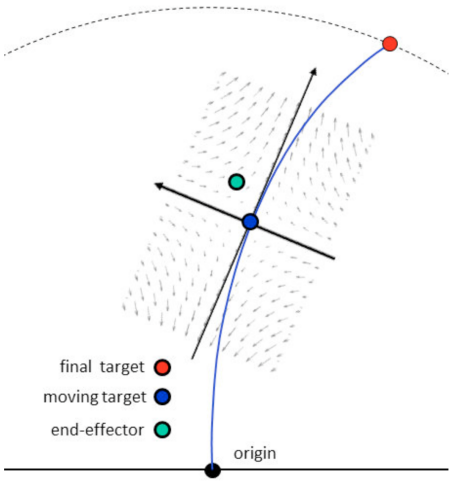



Figure 1.1: Manifold Designed by Morasso et al. [16]

behavior. In a certain way, guidance forces could cloud the internal perception, called proprioception, during a movement and so the important information would get lost. This hypothesis was proven by several studies like the one from Schmidt et al. in 1991 [21].

Nevertheless, in very early stages of the learning process or in the case of very complex tasks, the subject can benefit from haptic guidance. This is especially true for more complex forms of haptic guidance (e.g. Assistive as needed, path control etc.). In this subsection, I will cover the different implementation methods used.

### 1.4.1 Fixed Gain Position Control

The first and the simplest approach to haptic guidance is fixed gain position control. This means that the robot tries to minimize the error without taking into account the behavior of the subject, who is seen as disturbance by the robot. The input for these controllers are trajectories, meaning that the desired position is fixed to a certain point in time. Therefore, position control is very restrictive concerning the movement of the subject. Furthermore, this method prevents the subject from making errors, which has been shown that could be essential for motor learning [20]. However, in the case of very severely impaired subjects or pure novices to a complex task, fixed gain position control could serve as a base in the first phase of the learning. This is partly motivated by the fact that motivation and the reward of finishing a task successfully enhances learning effects [5]. At the same time, the challenge point theory states that if the task level is too challenging and the subject is not able to complete it, the learning of the task could be hindered [7]. Nevertheless, in the case of skilled subjects or in a later learning phase, fixed gain position control has been shown to be the worst method in nearly all the studies on that topic.

### 1.4.2 Path Control

Path control is a common method that is implemented in many motor learning devices. The main idea behind path control is that the subject is allowed to make a certain amount of error but react if the error gets too big and push him/her back to the desired position. This is often implemented as a threshold tunnel, in which the subject can move freely, and soft walls, that produce slight control forces once they are reached. Often, this type of controller does not take into account

timing aspects because the subject is allowed to move freely on the path. Such a controller was, for example, implemented by Rauter et al.[19]. They compared different kinds of path controllers with common position control in a complex, continuous rowing task and they found that position control enhanced the timing aspects while path control improved spatial aspects. Concerning the skill level, their main goal was to investigate the effects on low skill participants. Therefore a statement about skilled subjects can not be concluded from this study.

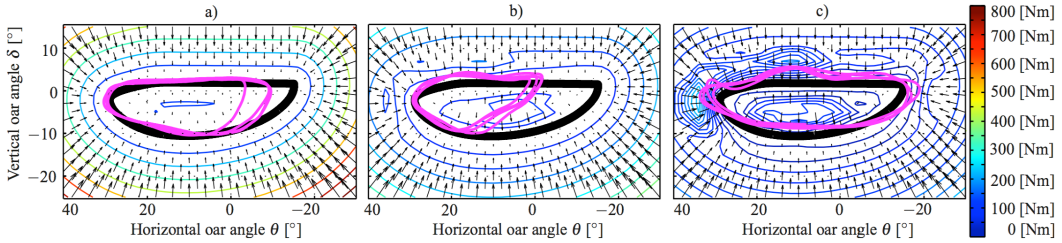



Figure 1.2: AAN path control. a)-b) shows the evolution of the force field and the desired trajectory (black) [19]

An interesting method they tried was a path controller that adjusted the boundary force fields in function of the performance, meaning it reduced guidance forces in regions where the subjects stayed in the threshold for multiple times and vice versa. This strategy could be referd to as an AAN path control.

A path controller can also be enhanced by imposing little timing constraints. This is done by adding a force field, also referred to as flux, in the inner of the threshold tunnel. This way, the subject is pushed gently along the path. An example for this kind of controller is a study by Cai et al [2], who performed tests on mice, which had spinal cord injuries, with different kinds of AAN path controls. They learned that path control was more effective than simple position control. A possible reason for this might be the fact that a training with a fixed trajectory without possibility of error could drive the spinal circuitry towards a state of learning helplessness because it can not explore potential solutions for the movement pattern.

### 1.4.3 Force Control

Instead of controlling the robot with a classical position control approach, a force controller looks at the difference between the interaction forces, between the subjects and the robot, and the desired task forces. This is then used in a closed loop control to generate the forces that are applied by the robot. Training with this stategy has the advantage that kinematic aspects of the movement fluidity are enhanced, meaning that the number of velocity peaks is lower and the average velocity higher after training [1]. These results by Bluteau at al. were found in a study that compared a force controller and a position controller in a 2D drawing task of arabic letters and ellipsoids. Furthermore, if a force controller is implemented into other forms of controllers it enhances the transparency of the robot, meaning that all friction, weight or inertial forces of the robot can be compensated and the movement in the robot feels much freer and more fluid [25]. This can be of advantage in motor learning because the desired task forces are not shadowed by unwanted friction or inertial forces.

### 1.4.4 Shared Control

Another interesting concept is the method of shared control. Here a subject and a virtual teacher, who can be controlled by a real world expert like a physiotherapist, create a link between each other. This means that the input of the teacher and the input of the subjects are averaged, resulting in a common output. This average also can be weighted to ensure more dependency on the teacher than on the subject. However, Powell et al. found a major problem with this method and even with general haptic guidance (HG) algorithms. They claim that a subject cannot benefit from HG if he/she does not know if the forces generated by the haptic device are task forces (e.g. inertia of a tennis racket or the collision with the ball during a tennis task) or guidance forces (e.g. forces resulting from the error to the desired position) [3]. To overcome this problem Powell et al. took the idea of shared control and combined it with a proxy model . This proxy algorithm, proposed by Zilles and Salisbury [27], consists of a massless "god-object" often called "avatar" that is bound to the physical laws of the virtual environment, like walls or friction. This "avatar" or"proxy" is then connected via a spring damper system to the haptic device. If now, the shared control concept is added by also connecting the virtual expert by a spring damper coupling to the proxy you end up with a constellation as shown in fig 1.3.

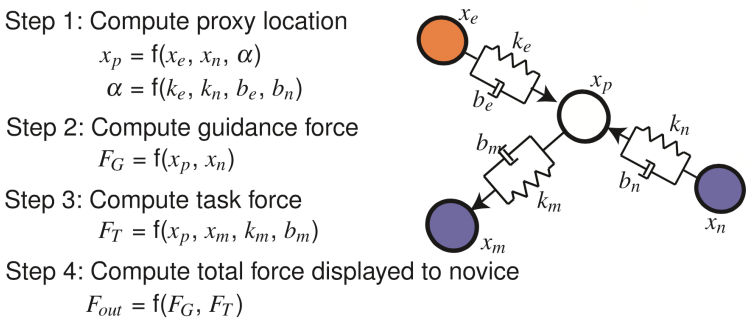



Figure 1.3: Shared Control Proxy Model by Powell et al. [3]

By only allowing forces to go in the direction of the arrows, task forces and guidance forces can be split. The average between the expert and the object is calculated and represents the location of the proxy. The distance between this calculated proxy and the novice produces the guidance forces. By looking at the distance between the mass an the proxy, task forces can be generated. Powell and his group made a study in which they compared several common control methods with this shared proxy method by outputting the task forces to one joystick used for the input and guidance forces to another one. The task consisted in continuous tracking tasks. However, they found that during training and in evaluation trials the shared proxy method performed significantly worse compared to normal guidance or resistive feedback respectively. This may be because of the enhanced cognitive workload or the unintuitive set up, that uses both arms instead of one.

### 1.4.5 Temporal Fixture Control

Another idea is to provide haptic guidance only in certain timing or spatial cues. Powell et al., for example, tried a position controller, which was enabled during 100 ms followed by a total absence of feedback of 400 ms [3]. This is motivated with the idea that the subject's reliance on feedback decreases and he/she sees the haptic guidance more as an indicator where to go next than a complete guide throughout the task, which he/she can passively follow. However, in this study of a continuous tracking task, they found that the "nudging" (e.g. the interruptions of the robot)

was irritating and frustrating, which provoked that the retention tests of constant haptic guidance were better than those of the subjects trained with the temporal separated guidance.

### 1.4.6 Assisted-as-needed Control

As discussed before, the general problems with haptic guidance, especially with fixed gain position control, is the reliance of the subject on the guidance and the lack of freedom to make errors. To solve this problem, a few experts came up with a method called assisted-as-needed (AAN). To reduce the reliance on guidance, it is decreased gradually over time if the subjects errors are relatively small. This way the subjects get more freedom to preform the movement and therefore could create more errors. The subject has to concentrate more because the haptic feedback is not doing the entire task for him/her. As a result, the subject is training in his/her personal "sweat-spot", where he/she can just manage to hold the error small with the assistance he/she needs. To achieve this reduction of the guidance, control gains are faded according to a certain decay algorithm. This algorithm takes into account a task specific performance measurement. A common algorithm is the one, proposed by Marchal-Crespo et al., based on Emken et al. [13][6], described in equation (1.2)

$$G_i = f \cdot G_{i-1} + g \cdot E_i \tag{1.2}$$

$G_i$ represents the values of the gain, which should be reduced (e.g. proportional gains). $f$ is called the robot forgetting factor that ensures an exponential decrease in the case of no error and $g$ is the robot learning gain, which is a scale factor for the consideration of the error $E_i$. $i$ stands for the current time step. As a result of this equation, the Gain $G_i$ is reduced while $E_i$ is near to zero, whereas the gain goes up once $g \cdot E_i > f \cdot G_i$. An important property of this algorithm should be its reactiveness. Even if the error has been small for several minutes, once the error gets bigger, the gains should respond immediately.

Based on this equation, you could implement methods, where different gains are dependent on different errors, which was done by Marchal-Crespo et al. in a study analyzing the motor learning of a continuous steering task [13]. With the help of a force feedback steering wheel, a virtual wheelchair was driven along a predefined track. In order to give a fluent and a foresightful guidance, the control is done in relation to a point at distance $d$ ahead on the track. Therefore, three different errors were used for the controller, namely the look-ahead distance error ($e_{dis}$), the look-ahead direction error ($e_{ang}$) and the derivative of the last one ($\dot{e}_{ang}$).

The controller used, described in (1.3), had three different proportional error gains. Each of these were adapted with the help of the algorithm described in equation (1.2), and the corresponding error. $K_d$ for example takes $e_{dis}$ as performance measure.

$$F_{assist} = K_d \cdot e_{dis} + K_a \cdot e_{ang} + B_a \cdot \dot{e}_{ang} \tag{1.3}$$

First results showed that the AAN control decreased the error significantly compared to the control group. [13] A later study with same control methods investigated the effect of assisted-as-needed control on people with different initial skilled level as well as the effects on elderly or younger test subjects [12]. In general, they found that young subjects performed significantly better long term retention test whereas the benefits on long term retention skills of older subjects were better but not significantly. Concerning the improvements on timing aspects of the task, Crespo et al. analyzed turn initiation and straightening timings and found that these were improved. The analysis of the initial skill level showed that the benefits of the assisted-as-needed controller were better the less skilled the subject was, as can be seen in figure 1.5.

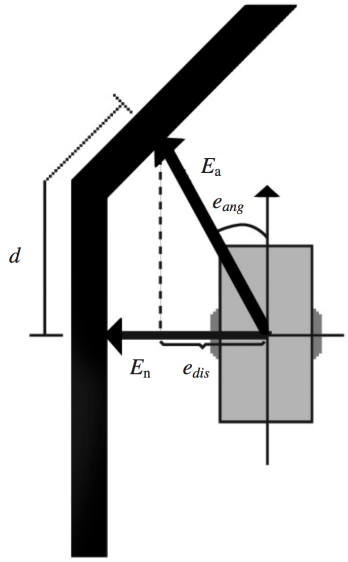



Figure 1.4: Different Errors of the Wheelchair Task [13]

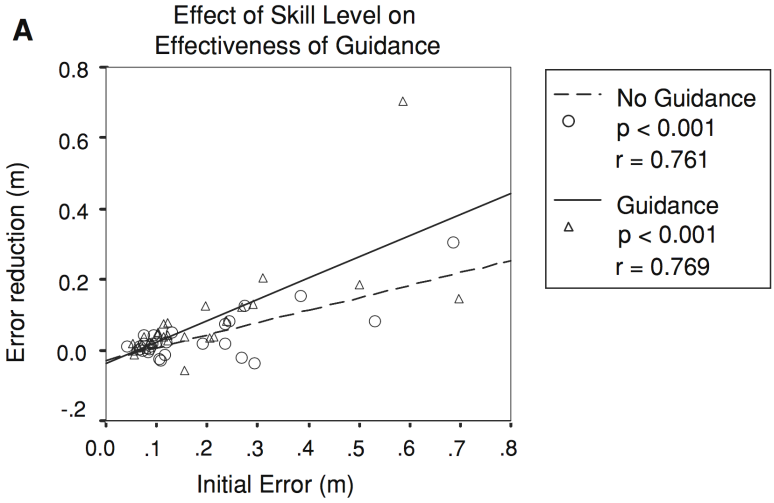



Figure 1.5: Effect on Haptic Guidance Based on Initial Skill Level for all Participents [13]

## 1.5 Discussion and Hypothesis

### 1.5.1 Discussion

Having reviewed a large part of the literature in haptic feedback in motor learning and robot-assisted methods, it becomes clear that the skill level, as well as the task characteristics are essential parameters, that can vastly influence motor learning. This aspect has unfortunately been omitted in many studies. Baseline tests are essential to correlate the results with the initial skill level, and thus the effectiveness on the method used. Furthermore, it is not clear if skill level can be taken as the same as different training phases. One could say that the same beneficial or detrimental effects that occur in the case of a skilled subject right at the beginning of training could be seen also in a late training phase of an initially less skilled subject.

In general, as a conclusion of the few studies that take into account the initial skill level, one could say that, for less skilled subjects, simple position control and path control showed good effects, while resistive methods in general turned out to be more efficient for more skilled subjects [3] [15]. A good way to guarantee beneficial effects for both situations could be to use an AAN controller that would work in the "sweat spot" of the subject. One could even go further and try to extend the AAN controller to a hybrid between guidance and resistance by allowing the gains to turn negative once a certain performance has been reached, which was done by Rauter et al [18] This way, they managed to get a controller that automatically adjusts to the skill level and to the learning phase of the subject .

Looking at the task characteristics, a difference between continuous and discrete, as well as a difference between strongly dynamic tasks and tasks with strong spatial aspects, should be taken into consideration to find the right control method.

Concerning timing aspects, two ideas are confirmed nearly in all the studies that take into account time dependencies. First, the restrictive character of fixed gain position controllers provoke that dynamic characteristics of a task are displayed clearly on haptic devices. Therefore this type of controller improves timing aspects far better than it enhances spatial characteristics of the task. Secondly, the exact opposite seems to be true for assistive methods that allow error (e.g. path control), or resistive feedback methods. The inner model of a path or a location is build in a better way due to the small errors that are amplified and recognized by the subjects [4].

Considering the distinct learning patterns in the case of continuous or discrete movements, the literature is far less clear about specific results, due to these characteristics. For most of the studies, either discrete or continuous task were implemented and it is hard to say if findings hold or break in the other case, when all the other conditions (e.g. skill level, control method) stay the same. Therefore, future studies should, in my eyes, realize test environments where this could be systematically analyzed. Nevertheless, I tried to summarize some of the work, reviewed in my studies, in table 1.1. A "Yes" means that the control strategy was significantly better than the control group and a "No" means that no beneficial effects could be shown. A "/" means that the control method was not tested by the papers cited in this work. A minor remark could be made about the Force Controller. It is basically improving transparency and fluidity of the haptic devices, which could enhance every robot-assisted rehabilitation experience. Therefore, the Force controller should be used in combination with other controllers.

Another point is it that nearly all the studies held with HG or Resistive Feedback (RF) cited in this report consisted of simple task (e.g. $< 2$ DoF). Therefore, for the skill dependency as well for the different effects on timing or spatial aspects, it would be interesting to see if the findings in the simple tasks hold for more complex tasks.

| | | Task characteristic | |
|---|---|---|---|
| | | Discrete | Continuous |
| Control Method | Error Amplification | Yes [15] | Yes [3] |
| | Noise Amplification | / | Yes [10] |
| | Manifold Method | Yes [16] | / |
| | Fixed Gain Pos. Control | Yes [16] | / |
| | Path Control | / | Yes [2] |
| | Force Control | Yes [1] | Yes [1] |
| | Temporal Fixture | No [3] | No [3] |
| | Shared Proxy Control | No [3] | No [3] |
| | Assisted-as-needed | / | Yes 1.5 |

Table 1.1: Beneficial effects of different control strategies

### 1.5.2 Hyptothesis

At the end, all the findings discussed in section 1.5.1, lead me to the hypothesis of my bachelor thesis:

*Haptic guidance improves time dependent components of motor learning tasks and not necessarily spatial aspects. This is especially true for an assist-as-needed controller based on position control. A development of a test environment of a complex task (4 DoF) is needed to see if the findings hold.*

# Chapter 2

# Methods

## 2.1 Designing the Game: "The Icecream Maker"

As a first task of this bachelor thesis, I had to design a user-friendly environment to study effects of different controllers on motor learning. The easiest way to create such a setting is virtual reality in which the subject tries to complete a virtual task. The input was given through a 7 DoF robotic device called ARMin. This upper limb exoskeleton is described further in detail in section 2.2.1. Furthermore, as discussed in the introduction, motivation plays a key role in the successful learning of a task. Therefore we decided to implement more than just a virtual reality environment, in which a simple tracking task is realized by displaying two dots or something similar. Instead, we chose to design a game, which motivates the subject throughout a series of tasks. As the hypothesis states that an assisted-as-needed controller, based on position control, shows the best results in enhancing timing aspects of a task, the game concept should take into account that the task to be completed should have strong timing aspects. It would make, for example, no sense to design a game, in which a serious of positions should be reached without any timing constraints, as it is found in the games that are already implemented for the ARMin.

### 2.1.1 Game Concepts

I came up with 3 game concepts, always keeping in mind the two main criteria, namely time dependency of the task and motivation and fun throughout the task. I specified the following points for each game concept:

- General Idea
- Game Procedure
- Trajectory/Velocity profile
- Feedback
- Personal Opinion/Motivation/General Thoughts

Out of these concepts, we chose the one that fitted best to the duration and motivation of the thesis and I started implementing it.

#### 2.1.1.1 Concept 1: "The Orchestra"

**General Idea**

Playing an instrument or being a conductor includes various time dependent tasks, that are not necessarily bound to position (e.g. a conductor must do his strokes in a specific time pattern but the way he does it can vary vastly). One could implement a game, where the player takes several roles in an orchestra. He/she could be :

- Conductor
- Cellist
- Trombonist

These 3 roles consist in moving the arm in a specific, time dependent, pattern and could be used to study continuous or discrete motor learning tasks.

**Game Procedure**

Step 1: The game begins with the player in the role of the conductor, starting the count for a piece and conducting the first few measures. After the player has achieved the goal movements, the role switches to keep the players attention.

Step 2: The role switches to one of the instruments. This setup could be similar to all the guitar hero games, where notes, in form of bars, fly towards the player and the player needs to perform the right movements.

**Trajectory/Velocity profile**

**For Cello:** The player only needs to focus on the rhythmic aspect of the tones, not on the pitch. Short bars would indicate short notes, while long bars would indicate long notes to play. With the distance between the bars, the pause duration could be controlled. Thus, the task could vary between continuous and discrete.

**For Trombone:** The player needs to focus more on the pitch than on the rhythmic aspects of the tone because he has to put the arm at the right place, at the moment the tone should be played. However, he can leave it there if no tone has to be played. The paths could include discrete (single tones, staccato) or continuous (glissando) tasks.

**Feedback**

Visual: The game would give concurrent visual feedback because, in my eyes, this type of game would make no sense without it.
Haptic: As specified in my thesis, the system would give haptic feedback in form of as assisst-as-needed (e.g. fading feedback, frequently reduced feedback).
Audio Feedback should be implemented as well as the game depends on music.

#### Personal Opinion/Motivation/General Thoughts

In my eyes, this game will be much more complicated to implement, but it might be worth a try because of the complex cello and trombone movements that could be investigated in motor learning aspects. The subjects would probably have more difficulties getting into the game or may have a complete lack of musical feeling, so that strong haptic assistance would be absolutely necessary at the start.

### 2.1.1.2 Concept 2: "The Orbiter"

#### General Idea

Gravitational Orbits is a great demonstration how a change of velocity can have an impact on the position path. One could imagine a game, where you have one or multiple gravitational attractors and the player has to place a satellite into a certain orbit or manipulate it afterwards.

#### Game Procedure

Step 1: The player is confronted with a certain situation concerning one or multiple attractors. Then the player is shown a certain goal orbit.

Step 2: Haptically, the system shows the player how to reach a certain velocity.

Step 3: The player has a try to reproduce the trajectory and reach the needed final velocity.

Step 4: Repeating Step 2 and 3 until the player got it right.

Step 5: Changing the desired orbit into a different shape.

Step 6: The player has to adjust the current orbit into the new desired orbit by "sweeping" over the satellite and adjusting its velocity.

#### Trajectory Velocity profile

This game would be mostly limited to discrete movements. One could implement an artificial interface that could control the speed of the satellite by completing continuous motions but it

would probably confuse the player and provoke a cognitive overload, creating an unhealthy motor learning environment.

#### Feedback

Visual concurrent and terminal feedback would be favorable at the beginning as well as strong haptic guidance during the first attempts.

#### Personal Opinion/Motivation/General Thoughts

I find the concepts of orbital physics very interesting and therefore see a big potential in making a game out of it. It would motivate people a lot by showing them how velocity changes can have an impact on the orbits. The task is focusing heavily on discreteness of the movements and emphasizing on the final velocity. Further, tasks will be difficult even in 2D and could be frustrating at the beginning.

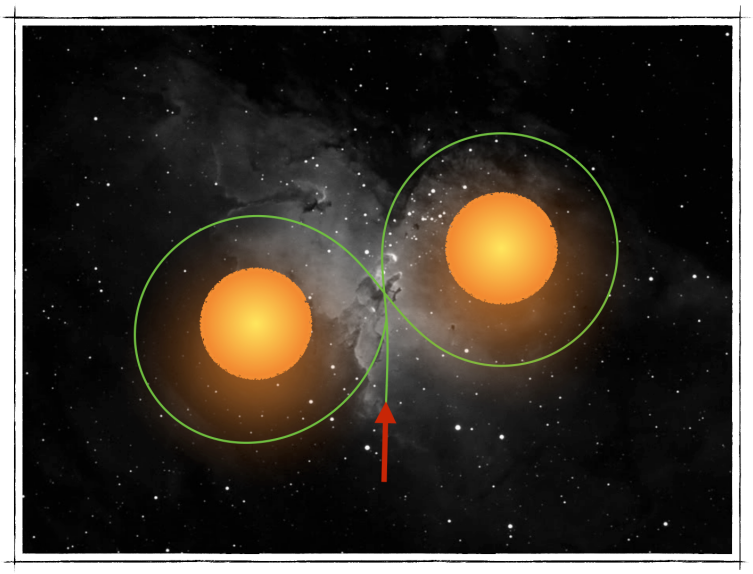


Figure 2.1: Illustration of the Game Concept "The Orbiter"

#### 2.1.1.3 Concept 3: "The Ice Cream Maker"

**General Idea**

The player is an ice cream maker in his ice cream van. He/She can look out of the window and can take orders from children showing up. He/She has to chose the right ice-cream flavor, sauce, and toping, and fill the cup with the right ingredients. This is done by following an ingredient distributor that follows the desired trajectory. After he/she has collected the right amount of all the ingredients, the player returns the ice cream cup to the child and puts the money into the cash box.

**Game Procedure**

Step 1: Player is looking to the window and sees the order. By reaching the arm for an empty cup, the player turns to the left by 90°.

Step 2: The player has to choose the right flavor of ice cream and then follows the path under the right distributor. The distributor is moving so that the player has to follow its trajectory. When a certain amount is successfully collected, the player turns again 90° to the left.

Step 3: Same procedure as in step 2, just with different sauces.

Step 4: Same procedure as in step 3, just with different toppings. After completion, the player turns back around to the child and gives him the ice cream cup.

Step 5: The player gets the money and has to put it into the cash box (optional).

**Trajectory/Velocity profile**

The general idea would be to let the movement path, in terms of position, fixed and vary the velocity path, according to the difficulty. One could imagine these difficulty associations:

| Ingredients | Easy | Medium | Hard |
|---|---|---|---|
| **Ice Cream** | Vanilla | Chocolate | Strawberry |
| **Sauces** | Chocolate | Strawberry | Vanilla |
| **Toppings** | Cherry | Biscuit | Chocolate Chips |

Each ingredient would correspond to another position path and with increasing difficulty, the total $\Delta v$ of the path increases. For example, the ice-cream distributor could follow an oval path in the x-y-plane, shown in Figure 2.2.

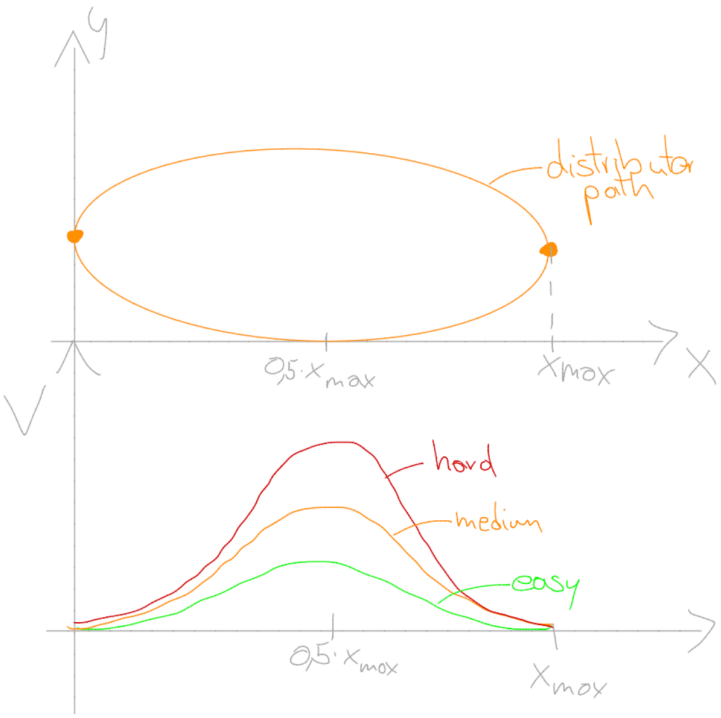



Figure 2.2: Example for a Velocity profile and Trajectory

The second ingredient distributor could move in another plane, while the third could be a 3D path, in order to have an increase in difficulty, even in the easy stage (e.g. vanilla + chocolate sauce + cherry). The velocity profiles above could be continuous (distributor is immediately going backwards in the same way, once it reached $x_{max}$) or discrete (the distributor waits for a few second in the corners) This could allow direct comparison between discrete and continuous motor learning aspects. The velocity profiles could also take very different forms (Gaussian, Linear exponential, etc. . . ).

### Feedback

Visual: The game would give concurrent, visual feedback because, in my eyes, this type of game would make no sense without it.
Haptic: As specified in my thesis the system would give haptic feedback in form of an assisted-as-needed (e.g. fading feedback, frequently reduced feedback).

**Personal Opinion/Motivation/General Thoughts**

In my eyes, this kind of game, if properly implemented, would provide a good motor learning environment because the subject would have immediate and frequent reward, due to the fact that he/she is completing one task in a couple of minutes and he/she is making the child happy. Thus the subject will probably be motivated for a longer time and will try to reach more difficult goals every time (i.e. increase the difficulty). The fact that any velocity path could be implemented, increases the attention of the subject, who could be faced with a different velocity profile every time.

### 2.1.2 Discussion about Game Concepts

Having discussed all the advantages and disadvantages of each game concept, we came up with the decision to chose **The Ice Cream Maker** as the final idea to implement a good environment for basic research in motor learning. This decision was mainly motivated by three arguments. First, the idea of "The Orchestra" was rejected because it would blow the time frame of this thesis, due to the fact that I could not dedicate all of the time to the game but would have to implement the controller as a second step. However, it has to be underlined that the study of musical movements is a very interesting topic as shown by Maes et al [11]. Their study showed that a discrete movement is more cognitively demanding than continuous movements by analyzing cello players playing a staccato or legato piece. Secondly, the concept of "The Orbiter" did not fit the thesis either, because it was clear, from the beginning, that the game should work with other controllers besides mine, namely an EA controller. The discrete movement characteristics of the task would not make a good testing environment for this type of controller. At last, with "The Ice Cream Maker" game, the lab would have an appropriate tool to test different velocity profiles and different controllers. One could easily change the velocity profiles or trajectories without reasoning it in the game. Another argument for this concept was that it would make people feel happy and satisfied because they could make a virtual child happy by preparing the ice cream. As a matter of fact, most people love ice cream.

Nevertheless, there are a few changes between the final game and the concept I first came up with. For space reasons, I decided to just place one distributor in every stage of the game instead of one for each difficulty. This way, the patient would not have to chose the right distributor under which he would have to hold the ice cream cup. The ice cream distributor, for example, automatically distributes the right ice cream flavor depending on the difficulty chosen. Furthermore, step 5 was not implemented at all in order to eliminate unnecessary complexity and to put the focus more on the main tracking tasks. For the same reasons, the reaching movement for the cup to start the game was not implemented.

### 2.1.3 Creation Process and Design Methods of "The Ice Cream Maker"

#### 2.1.3.1 Game Engine: Unity© 5.0

Unity is a cross-platform game engine, which was developed by Unity Technologies. Before version 5.0, there was a free personal version and a professional version that was available for a fee. Since version 5.0 that came out on March 3, Unity Technologies made all professional features available in the free version. My Bachelor Thesis started in February 2014 so the beginnings of the project were implemented in Unity 4.0 and I upgraded to Unity 5.0 around March 15. This enabled me to use all the features, especially some new light settings that enhanced in-game depth perception.

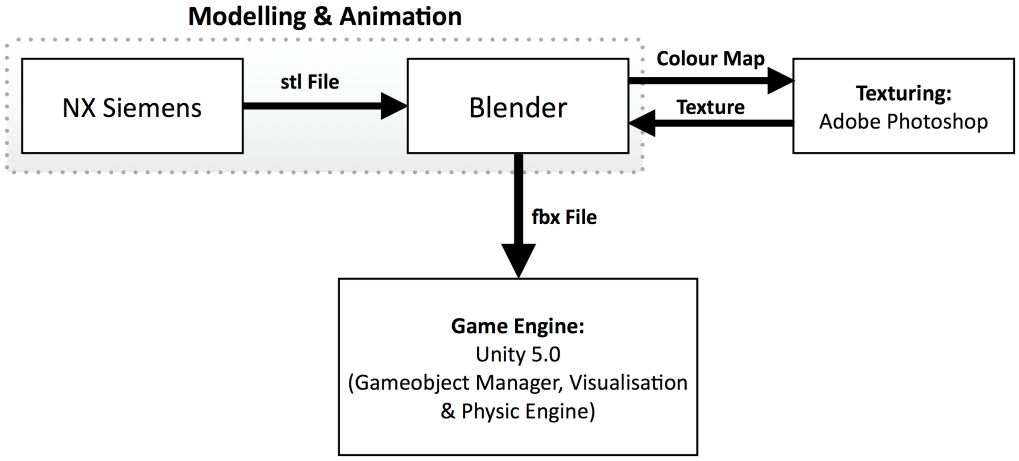



Figure 2.3: Schematic View of the Creation Pipeline

In this project, Unity is used to implement the game and the game physics. A detailed description of the Unity structure, all the game objects, scripts and interaction between those is found in the Appendix A.

#### 2.1.3.2 Modelling Software: NX Siemens© 8.0 and Blender© 2.74

To create all the game content and game objects I used two modeling softwares. On the one hand, I used NX Siemens 8.0, which is a widely used CAD Engineering software. I used it for all content that was strongly constrained in geometry because it is easy to define these in a coherent way. This enabled me to make easy dimension changes, where the whole geometry changed in dependency of the first change. The ice cream truck is a good example, where dependency between the geometry is important and an exact mathematical description is needed for the later design of the tracking trajectories.

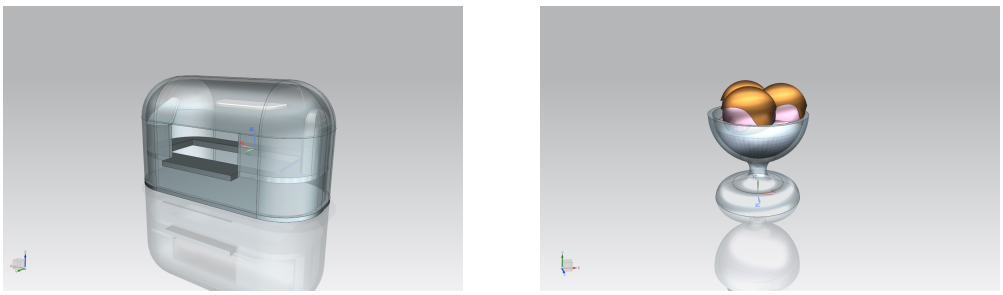


Figure 2.4: Van and Filled Cup Model in NX Siemens

For all the game content that was not so restricted in geometry, I used Blender 2.74. This is a free software for freeform modeling that is used to create high-end polygon meshes and renderings. Further, the files generated by Blender can be directly imported into Unity which is the reason why I also had to import the objects, created in NX Siemens, into blender and then pipeline them through to Unity.

#### 2.1.3.3 Texturing

After the geometry of the object is defined, it still has no color. To change that, I used Blender to create something that is called a UV-Map. This is basically the mesh of the object, unfolded onto a 1024 by 1024 pixel image. This image can be edited and painted in an image software like Photoshop and then reimported into Blender. There, the image gets folded back on the 3D object to color the faces. For more detail, please refer to the example of a creation process of the kid character in section 2.1.4.

#### 2.1.3.4 Rigging and Animation

In order to animate a 3D mesh object, it needs a Rig, which can be seen as a skeleton of the model. This rigging process is also done in Blender. You define a limited number of bones that go inside the mesh and then associate certain mesh surfaces to certain bones so that, when the bone moves, the associated mesh surface moves with it. For an example please refer to section 2.1.4. Once I had finished rigging, I could animate the skeleton by defining certain movements with keyframes. In these keyframes, you define your wanted position in this specific moment in time. Afterwards, the software does a Bezier Interpolation between these key positions and you will end up with a smooth animation.

#### 2.1.3.5 Importing into Unity

Once the creation of a certain object was finished, I imported the object with all its information about the texture, rig, and animation into Unity as an .fbx file. The object could then be manipulated according to some logic (e.g. position, rotation, size).

### 2.1.4 Case Example of a Creation Process: The Kid Character

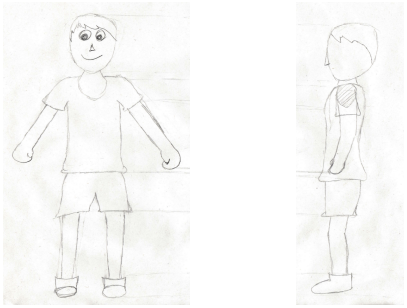
From the beginning, it was clear that the game would start with a scene, where the subject stands in his/her ice cream van and waits for the kid, that is coming to the van and orders an ice cream. Therefore, one of the earliest models I made was the kid Character, where I learned to use Blender with the help of a youtube tutorial series by Sebastian Lague[1]

To start, I made two concept drawing of my Character model; one from the side and one from the front, as seen in figure 2.5 . This is a good way to have a reference while modeling the 3D character, in the same way how sketches are important in the beginning of every design process in mechanical engineering.

After importing those two drawings into Blender and defining them as front and side view, I could start modeling; first a raw shape and afterwards more and more details. Once I was happy with the resolution of the mesh and detail features of the character, I applied a smoothing algorithm that makes the sharp edges disappear and the polygonal look of the body. The results can be seen in figure 2.6.

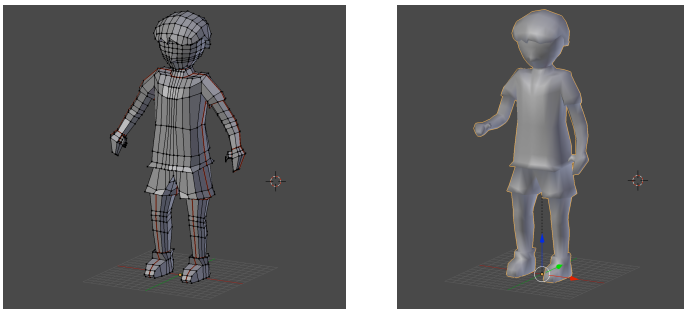
Once I was happy with the geometry of the body, I started texturing the character. First I had to mark seem lines, which are the lines that are cut in order to unfold the mesh onto a flat surface (red lines in figure 2.6a). After that, I could create the UV Map, which can be seen in figure 2.7a. Before importing that 1024 by 1024 pixel map into an image editor, I also created an Ambient

[1]Blender Character Creation: Modelling - https://www.youtube.com/watch?v=DiIoWrOlIRw, last checked: 09.07.2015



(a) Front View (b) Side View

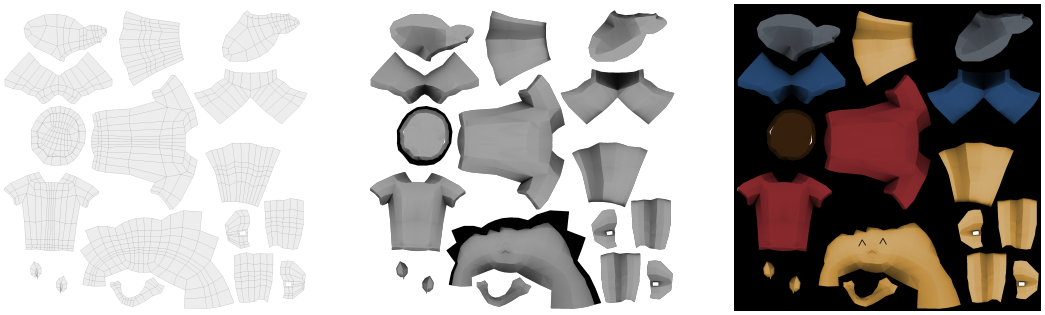
Figure 2.5: Design Sketches of the Kid Character



(a) Raw mesh (b) Smooth mesh

Figure 2.6: Mesh's of the Kid Character

Occlusion Map (AO-Map). This map holds information about the shading of the character if it would get lightened by a directional light from above (figure 2.7b). Creating another layer in the final texture with this Occlusion map, turns the texture much more realistic and good looking. After painting the different parts of the character in an image editor and combing it with the AO-Map, I ended up with the final texture, that could be reimported into Blender (figure 2.7c).



(a) UV-Map (b) AO-Map (c) Final Texture

Figure 2.7: Different Stages of the Texture

After this was done, I could start rigging the Body in order to be able to animate the character in a later stage. I used the same youtube tutorial series as before and ended up with a decent, not unnecessary complicated character rig, which I could use to do basic animation moves. As described in section 2.1.3.4, for every animation, namely winking, going, taking the cup and turning,

I defined several keyframes until I was happy with the interpolated movements. You can see the different key-positions of the "turning" animation in figure 2.8.

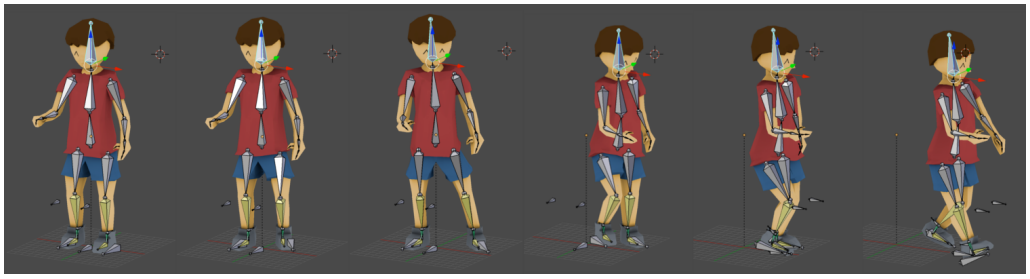


Figure 2.8: Keyframes of the "Turning" Animation with Displayed Rig

Finally, the whole model, with its mesh, texture, rig, and animation, could be imported into Unity. There, I could program the game logic to trigger the animations with the help of a state machine and move the body according to the animation that is playing.

### 2.1.5 Final Structure and Features of the Game

When you launch the game, you get into the Main Menu (figure 2.9), where you have several main options that you can change. You can chose between 3 game difficulties which correspond to different velocity profiles. With increasing difficulty, the velocity peaks and accelerations get more pronounced. Furthermore, you can enable or disable visual feedback, which is explained separately in section 2.1.5.1. Finally, you have to select with which arm you are playing (right or left) in order to visualize the end-effector in the right way.

If the player clicks on the "Advanced Settings" button he gets to another screen with more settings (figure 2.9). There you have several other options to set:

- You can set the duration of each of the tasks.
- You can chose which task should be played and which task should be skipped by ticking the wanted tasks.
- You can select between the different controllers implemented in the Simulink© model:
  - Zero Force Controller
  - Haptic Guidance:
    - Constant Gain
    - Constantly Decaying Gains
    - Exponentially Decaying Gains
    - Error based Decaying Gains
  - Error Amplification
  - Mixed EA and HG Controller
- You can enable or disable the force control for each controller separately.

By clicking on the "Submit" button, you get back to the Main Menu, where you can press on the "Start" button to start the game.

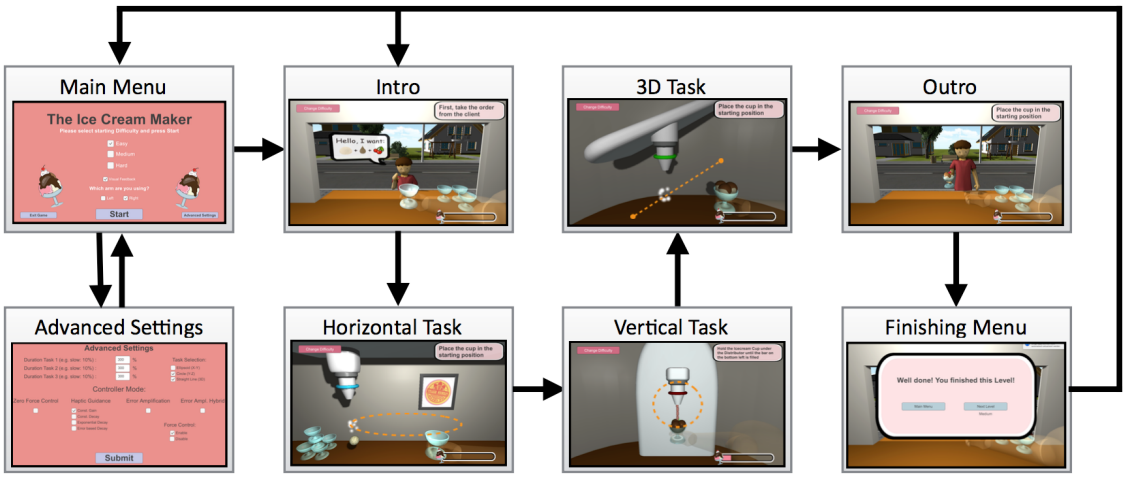



Figure 2.9: Different States during the Game

First, the subject sees an intro scene, in which he/she sees the kid approaching and a speak bubble appearing, which contains the order that the player has to fulfill. After 5 seconds, the player turns around to the first task, where he/she is asked to put the cup under the distributor, which starts moving in the predefined trajectory, after the player has reached the starting position. Once he/she has played the task for a certain duration, he/she turns to the next task. The same rules apply for this task as for the the task before. After that, the subject gets to the third task, where he/she has to follow, again, the same rules until he/she is finished. The trajectories and the velocity profiles used are described more in detail in section 2.3. After finishing these three tasks or by skipping some of them as defined in the settings, the subject gets to an "Outro" scene, where he/she has to put the cup in a sparkling spot to give the cup to the kid and end the game. Once finished, a finishing menu pops up, where you can chose if you want to play the next difficulty or get back to the Main Menu. During the whole game, the player can chose to end the current difficulty and to get back to the Main Menu to change some settings or just exit the game. All these different states of the game can be seen in figure 2.9.

#### 2.1.5.1 Visual Feedback

In order to make the subject aware of where exactly to hold the cup beneath the distributor, meaning at which vertical position, I implemented a region with a sparkling particle effect. One of the biggest problems in Unity is depth-perception, due to a lightning system that is not optimally implemented. Therefore, it is very hard to realize in which position something is situated in the depth-coordinate. To solve this, I programmed it in a way that, once the cup is held in the correct position, the particles turn green. When the subject leaves the wanted region, the particles turn white again. This can be seen in figure 2.10.

Furthermore, another visual feedback was implemented to help the subject realize if he/she has caught an ingredient. This was done by turning the cup green for half a second every time something was caught.

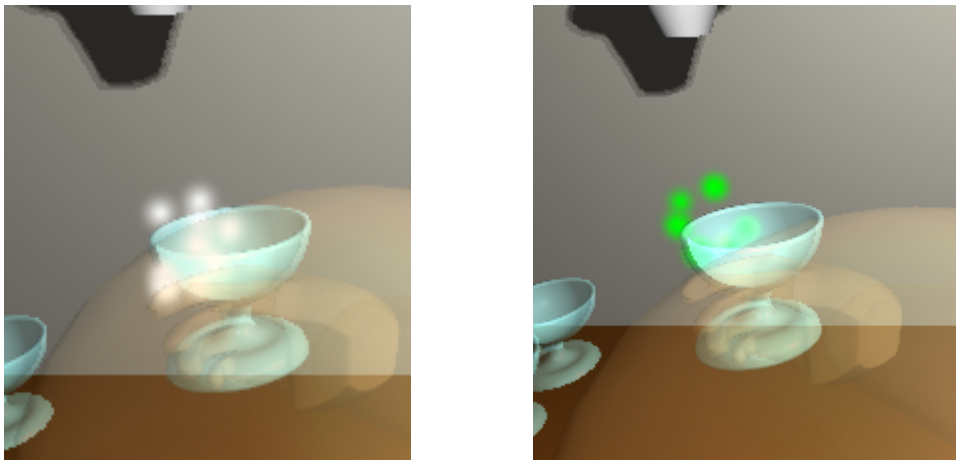


Figure 2.10: Left: Without Feedback; Right: With Feedback

## 2.2 Assisted-as-needed Controller

As a second part of my thesis and as conclusion of my studies on mechatronics, I implemented an assisted-as-needed controller for a robotic device, called the ARMin. The goal of this part of my thesis was to create a Simulink model, that is running on a realtime xpc-target and is communicating with "The Ice Cream Maker" game via UDP-Protocol. I implemented multiple control settings which can be chosen in the game as described in section 2.1.5. A lot of the structure of the game can be seen in the controller implementation, so the controller is specifically designed for the game. Furthermore, in a last phase, this assisted-as-needed controller with various control options was combined with the error amplification controller, implemented by Tanja Baumann. This was done to create a complete test environment for the basic research on motor learning in the SMS lab. The overall goal is to compare EA and HG in tasks with strong timing aspects.

### 2.2.1 The ARMin IV

The ARMin IV is a 7 DoF robotic device for upper limb rehabilitation, designed by the SMS-lab at ETH Zurich in collaboration with the University Hospital Balgrist [17]. The current version is the fifth generation and consists of 7 actuators with encoders and potentiometers. The encoders are used for closed loop control and the potentiometers serve as absolute measuring units to initialize the encoders. Although it has 7 DoF, it is a non-redundant robot because axis 7 consists in closing and opening the hand.

Furthermore, the ARMin IV has 3 force sensors with 6 DoF (i.e. x,y,z forces and x,y,z torques). These are located beneath three cuffs, where the subject is attached to the robot to measure the interaction forces between the user and the robot. These can be used to implement closed loop Force Control, which can enhance transparency of the robot.

There are a lot of applications that are already implemented, namely a setup to train Activities of Daily Life, several path and position controllers as well as a Zero Force Controller. Further compensation forces are calculated in nearly all these controllers. These forces compensate the weight of the joints as well as friction and some spring components of the robot. As in most robotic devices, the controllers can be designed in the joint space or in the end-effector space and you can get from one to the other using forward or inverse kinematics.

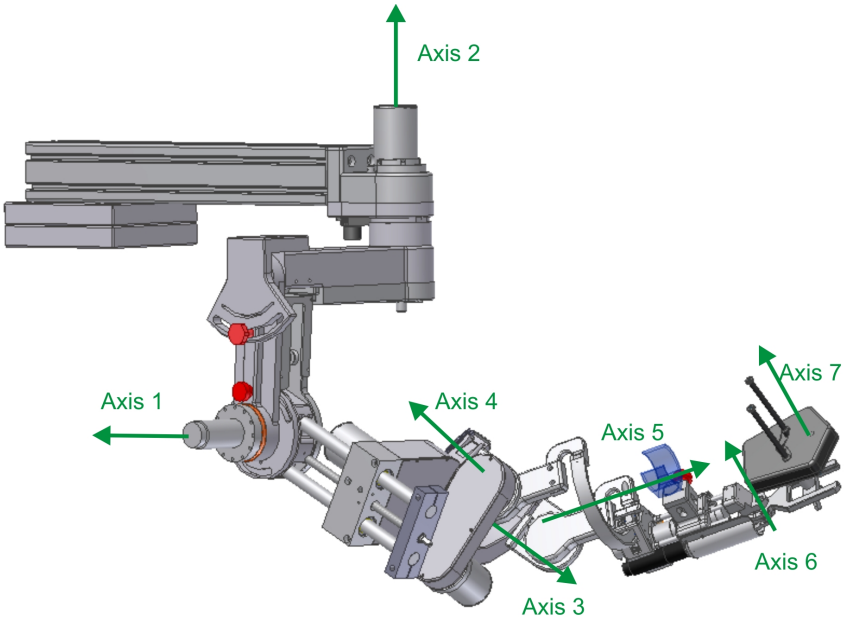



Figure 2.11: Axis Setup of the ARMIn IV

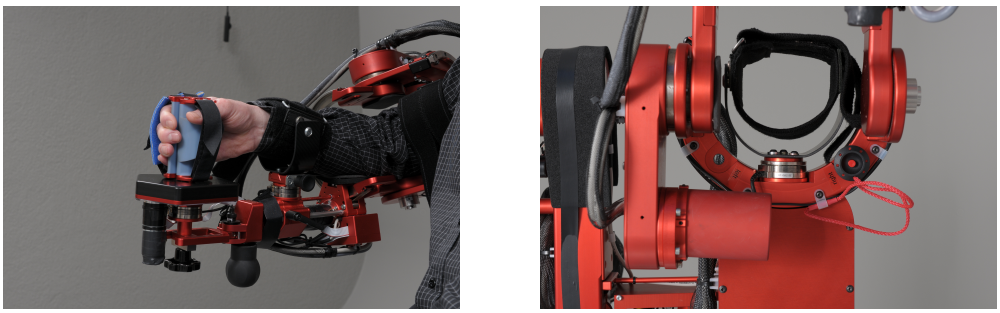


Figure 2.12: Photos of the Force Sensors of the ARMin IV

### 2.2.2 Main Structure

The first part of the controller is done in the end-effector space. It consists of a simple PD-controller in $x$, $y$, $z$, with different gains in each direction. The transfer function looks as follows:

$$PD_i = k_i + d_i \cdot s \tag{2.1}$$

where the index $i$ represents one of the three coordinates, namely $x$,$y$ or $z$ and $s$ is the complex variable resulting from the Laplace Transform.

In a first phase, I tried to find the optimal proportional and differential gains in the case of a fixed gain controller to have a reference and starting values in the case of decreasing gains. After trying some values and looking for a stable control as well as a control that guarantees following the velocity profile precisely, we came up with the following values for the proportional gains:

$$k_x = 15\,\mathrm{cm}^{-1}$$

$$k_y = 15\,\mathrm{cm}^{-1}$$

$$k_z = 30\,\mathrm{cm}^{-1}$$

Further, the differential gains were derived from the proportional gains with the thumb rule

$$d_i \propto k_i \cdot 0.1 \tag{2.2}$$

and fine tuned afterwards such that the following values were taken:

$$d_x = 0.2\,\mathrm{cm}^{-1}$$

$$d_y = 0.2\,\mathrm{cm}^{-1}$$

$$d_z = 0.12\,\mathrm{cm}^{-1}$$

We decided not to use an integrator part because the appearance of steady state errors is not important for the use of the controller or the study, namely to investigate timing aspects of a tracking task. Furthermore, we could avoid unnecessary instability brought by the integrator part because of its phase shift and unstable nature.

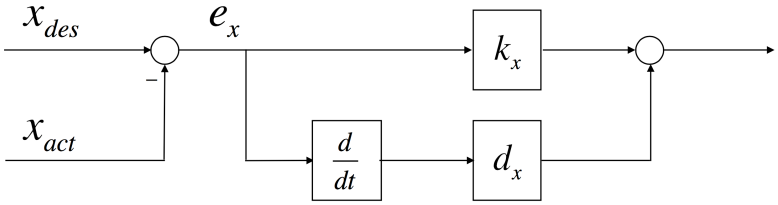



Figure 2.13: PD Controller for the $x$-axis

Before sending the signals to the motors, inverse kinematics are applied to these control signals in order to represent them in joint-space.

### 2.2.3 Gain Decaying Rules

After I have got the above designed constant gain PD controller working on the ARMin without encountering any instabilities, I started to implement three different decaying rules for the proportional and differential gains.

#### 2.2.3.1 Linear Decay

As a first method to decrease the controller gains over time, I took a linear decay over time. This means that the gains start with an initial value, which is the value found in section 2.2.2, and decrease a certain amount for each incremental time step. Because the proportional gains, as well as the differential gains, are decreased at the same time and rate, I used a simple pre-multiplier for these gains with a "percentage constant" that starts with 1 and goes down to 0.

In this way the calculations are done just once and I could easily switch between different decaying rules. Therefore, the equation governing the linear gain decay could be described as

$$\begin{cases} G_i = G_{i-1} - \delta \cdot \dfrac{1}{f_s} \\ G_0 = 1 \end{cases} \tag{2.3}$$

where $G_i$ represents the percentage of the initial gain value at time step $i$, $\delta$ is the decreasing factor, which is the drop rate per second and $f_s = 1\,\text{kHz}$ is the sampling rate of the ARMin. Having tried some values and considering a task duration of 3 minutes, we came up with $\delta = 0.02\,\text{s}^{-1}$. If expressed as function of time, you could also write equation (2.3) as follows:

$$G(t) = 1 - \delta \cdot t \tag{2.4}$$

As a result the value for the proportional gain in $x$ direction, for example, can be calculated as described in equation 2.5, where $k_{x_i}$ is the value of the proportional x gain at time step $i$ and $k_{x_{intial}}$ is the intial value. All other proportional and differential gains are calculated in the same manner.

$$k_{x_i} = G_i \cdot k_{x_{initial}} \tag{2.5}$$

#### 2.2.3.2 Exponential Decay

I implemented a second decaying method, which decreases the gains exponentially over time. As in the case of the linear decay, a pre-multiplier percentage constant is decreased without taking into account the subject's performance. The equation that was implemented in the Simulink model is described as follows:

$$\begin{cases} G_i = f \cdot G_{i-1} \\ G_0 = 1 \end{cases} \tag{2.6}$$

where, again, $G_i$ represents the percentage of the initial gain value and $f = 0.99998$ is the robot forgetting factor, which, in this case, is the same for all the gains. Explicitly the gain percentage,

as a function of time, could be written as:

$$G(t) = e^{\log(f)\cdot t} \tag{2.7}$$

#### 2.2.3.3 Performance based Decay

This third method of decaying gains is often called assisted-as-needed. The goal is to take information about the performance of the subject and find a law that decreases or increases the gains using this information. In this way, the controller or the assistance helps the subject more in situations where the performance is bad and he/she is not able to fulfill the task. Furthermore, the assistance is weaker in situations in which the subject is near to the desired position or velocity and he can not rely on the haptic feedback. In an ideal case the subject's training is held in a sort of "sweat spot" in which motor learning could be optimal.

As a first design step, I had to think about which information would be the best to measure the performance of the task. I chose the spatial error, meaning the 2nd degree vector norm of the error. This seems a bit counterintuitive because the main goal of this project should be the analysis of the timing aspects and not necessarily the spatial aspects, but, in fact, an error in timing shows best in the spatial error. If, for example, the subject is to late for a certain event on the trajectory (e.g. dropping of an ingredient) this results in an spatial error as well because the desired position lays ahead of the actual position. Therefore, I calculated the error of the actual position and the desired position $E$ every time step $t_i$, meaning every millisecond, as follows:

$$E(t_i) = \sqrt{\Big(x_{des}(t_i) - x_{act}(t_i)\Big)^2 + \Big(y_{des}(t_i) - y_{act}(t_i)\Big)^2 + \Big(y_{des}(t_i) - y_{act}(t_i)\Big)^2} \tag{2.8}$$

In the same way as in the two sections before, I implemented a law that described a percentage constant in the interval $[0, 1]$ because I do not want the gain to exceed the initial values nor to turn negative. I defined an algorithm, that is based on an adaptive algorithm by Emken et al. and resulted in equation (2.9). [6]

$$\begin{cases} G_i = f \cdot G_{i-1} + g \cdot E_i \\ G_0 = 1 \end{cases} \tag{2.9}$$

where $f$ is again the robot forgetting factor and $g$ is the robot learning gain.

It was important to find the appropriate values for $f$ and $g$, so that a decent decreasing of the gains takes place when the error is small. However, once the error gets bigger, the gains should react fast enough to guarantee a training in the "sweat spot" as discussed before. Furthermore, because the last task is a discrete task, as described in section 2.3.3, where the subject needs to get back to the starting position every time, the overall time the discrete line trajectory is followed is less than in the first or the second tasks. Therefore, the values for $f$ and $g$ have to change in function of the task that has to be accomplished. After a few trials we came up with the following values:

- $f_{elip} = f_{circle} = 0.989$
- $f_{line} = 0.9955$

– $g = 0.0003$

### 2.2.4 Force Controller

After a few tests we encountered a problem concerning the hardware of the robot. Even though we added the compensation forces, as described in section 2.2.1, once the gains were nearly fully decreased, the task was not doable. This was due to the inertia and the complex 6 DoF architecture of the robot, which hindered the subject to make smooth and continuous movements. To solve this problem, we decided to take into account the force sensors and implement a supplementary force controller to enhance transparency. This force controller, together with the PD controller, forms the final structure of the controller as can be seen in figure 2.14. This combination is often called impedance control.

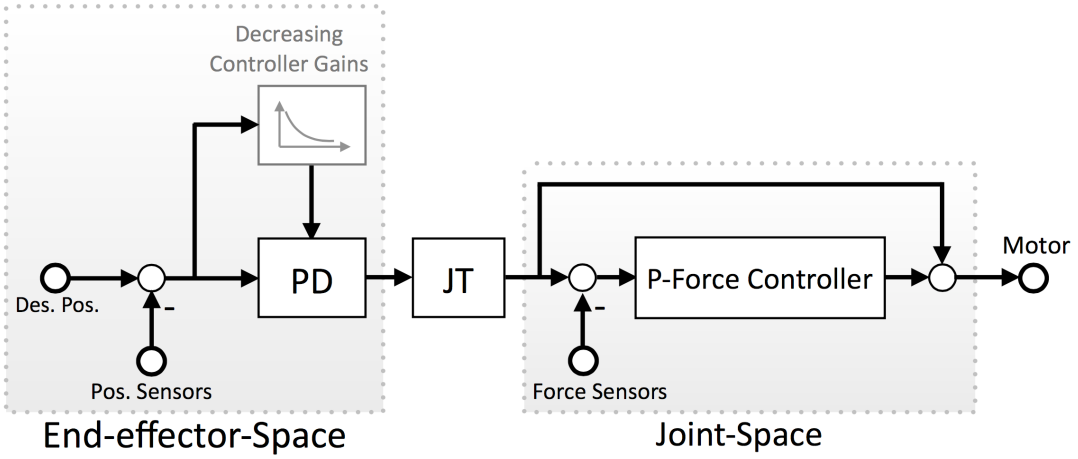



Figure 2.14: Block Diagram of the Controller Structure

In general, the task forces are calculated by the PD controller in the end-effector space according to the current gain values. Afterwards these forces, as well as the values of the force sensors, are mapped into the joint-space by the inverse kinematics. This is done with the Jacobian Transform represented by the block labeled with "JT". The force controller then looks at the difference between those and calculates the resulting,, final forces. These are then sent to the motor.

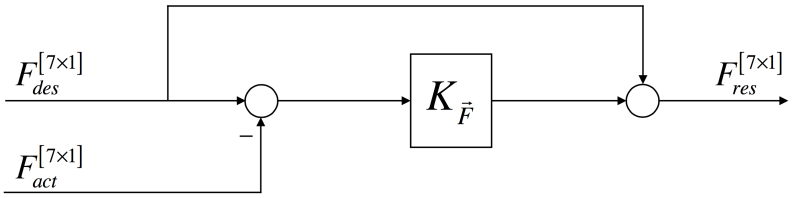



Figure 2.15: Block Diagram of the Force Controller

Concerning the force controller by itself, it consists in a simple proportional gain controller for each of the 7 axis's. This kind of controller, obviously, needs a feed forward of the input forces in order

to result in combination of the tasks forces and the forces that help improving transparency. The detailed structure of the Force Controller, as implemented in the Simulink model, can be seen in figure 2.15. The block $K_{\overrightarrow{F}}$ represents a left hand matrix multiplication by a $[7 \times 7]$ matrix, which contains the proportional gains for each axis in the diagonal.

$$K_{\overrightarrow{F}} = \begin{pmatrix} 1.7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.10}$$

The two last values are equal to zero because they belong to the wrist and the hand palm axis, which are not actuated and therefore, not needed in this case.

## 2.3 Trajectories

An important part of this research project and, therefore, for this thesis, was the definition of the trajectories, meaning its location in space and the velocity profile. Since my background was purely from mechanical engineering, we consulted two physiotherapist students, who helped us finding an adequate trajectory corresponding to important movements in physiotherapy. Even though this project serves for basic research, the final goal should always be the application in physiotherapy. Results from the conversations with the physiotherapist led to the conclusion that the movements should be as large as possible and the velocity profile should resemble to a Gaussian curve, which represents the optimal profile done by the the human body itself. This was found by Tamar et al [23]. Further, one of the physiotherapists thought it would be nice to have a straight line that goes from the bottom left to the upper right, which would consist in an extension of the elbow joint. This advice was applied in task 3 (subsection 2.3.3). For simplicity the plots and equations in the following sections are straight in relation to the coordinate system, meaning that the axis of the ellispoid and the circle coincide withe axes of the coordinate system. As the reader will learn in section 2.4.2, in reality these trajectories are turned by 45° around the $z$-axis and offsetted because the optimal workspace of the ARMin is not straight in relation to the coordinate system.

### 2.3.1 Task 1: Ellipsoid

As a first task, we decided to take an ellipsoid in the $x$-$y$-plane as trajectory and to keep the inherited sinusoidal velocity profile that resembles to a gaussian curve. The nature of this task is continuous if it is performed several times without interuptions. Equation (2.11) describes the trajectory of task 1. We defined the values $a = 20\,\text{cm}$ for the x-axis and $b = 15\,\text{cm}$ for the y-axis. The angular velocity $\omega$ is changed in function of the difficulty chosen in the game, meaning it can

take the values 0.5, 1 or 1.5 rad s$^{-1}$. In figure 2.16 the trajectory as well as the different velocity profiles can be seen. Here the term "extremities" describes the point with the biggest absolute value in $x$ (e.g. $x = 20\,\text{cm}$ or $x = -20\,\text{cm}$) and could be seen as the "turning points" of the trajectory. Figure 2.17 shows the trajectory in the context of the game. However, the trajectory can't be seen in the game itself.

$$\overrightarrow{r} = \begin{cases} x = a \cdot \sin(\omega \cdot t) \\ y = b \cdot \cos(\omega \cdot t) \\ z = 0 \end{cases} \tag{2.11}$$

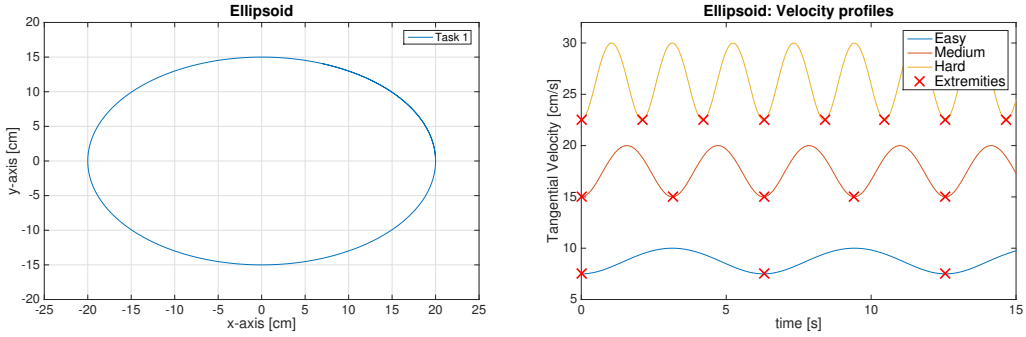



Figure 2.16: Plots of Trajectory and Velocity Profiles of Task 1

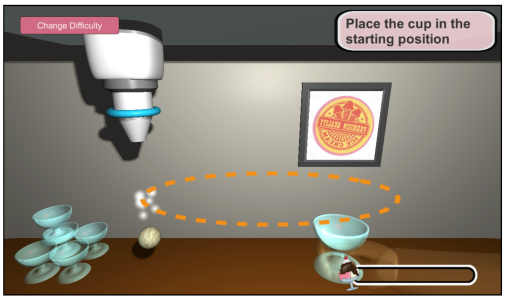



Figure 2.17: Trajectory 1 in the Game

Equation (2.12) shows the expression for the tangential velocity. It can easily seen that you can increase average speed, acceleration and total differences of the velocity by changing the value of $\omega$.

$$|\overrightarrow{v}_{tang}| = \omega \cdot \sqrt{(a \cdot \cos(\omega \cdot t))^2 + (b \cdot \sin(\omega \cdot t))^2} \tag{2.12}$$

### 2.3.2 Task 2: Circle with ellipsoid velocity profile

For the second task, we chose a circle trajectory in the $x$-$z$-plane, as described in equation (2.13). As in Task 1, this tasks consists in a continuous movement if repeated for several times without interruption. For the radius, we took the value $r = 20\,\text{cm}$. However, the inherited velocity profile of a circle is constant by definition and a function of the angular velocity $\omega$. Because this would make no sense to study the effects of haptic guidance on timing aspects, we changed this velocity

profile into an elliptic velocity profile to increase the complexity of the task. This was done with a method called time warping, in which time space is warped so that the time increase is not constant. Most of that work was done by Tanja Baumann and will be described in her thesis. Therefore, I will not go into detail on this topic.

$$\overrightarrow{r} = \begin{cases} x = r \cdot \sin(\omega \cdot t) \\ y = 0 \\ z = r \cdot \cos(\omega \cdot t) \end{cases} \tag{2.13}$$

Figure 2.18 shows the trajectory in the $x$-$z$-coordinate system as well as the velocity profiles after the time warping. The difficulty can be varied again through $\omega$ to increase the average velocity, acceleration and the total difference of velocity. Figure 2.19 shows the trajectory in the context of the game. As in during task, the trajectory can not be seen in the game,

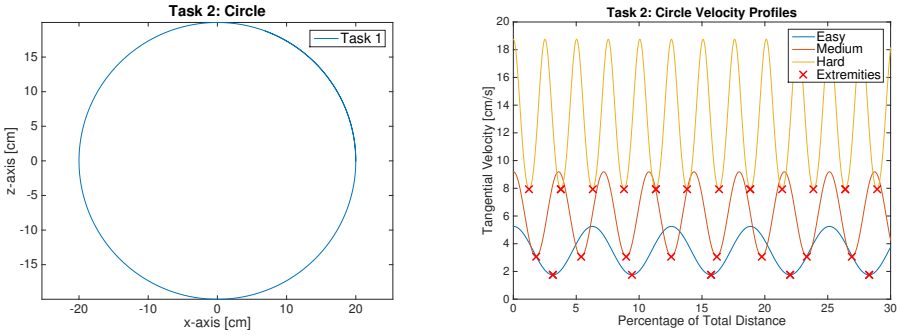



Figure 2.18: Plots of Trajectory and Velocity Profiles of Task 2

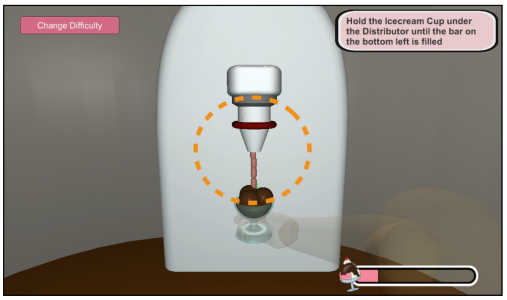



Figure 2.19: Trajectory 2 in the Game

### 2.3.3 Task 3: Line in 3D space

Task 3 is a line in 3 dimensions and it is very different from the first two tasks for multiple reasons. First, it consists of a discrete task, that gets repeated once the finishing point has been reached. Second the line has components in the 3 dimensions, which is more difficult for the subject to accomplish. Further, the velocity profile is not trivial and consists of an addition of multiple trigonometric functions with different frequencies. The trajectory, described in equation (2.14), consists of a starting point $\overrightarrow{P_1} = [40, 20, -17]^T$ and an end point $\overrightarrow{P_2} = [66, -16, 2]^T$. All these coordinates are in centimeters.

$$\overrightarrow{r} = \left(\overrightarrow{P_2} - \overrightarrow{P_1}\right) \cdot t + \overrightarrow{P_1} \tag{2.14}$$

The velocity profile was mostly implemented by Tanja Baumann and I will not go into detail on the time warping. Both, trajectory and the velocity profiles for different difficulties can be seen in figure 2.20. Figure 2.19 shows the trajectory in the context of the game but it is not shown in the game itself.

While the dropping of the ingredients in task 1 and task 2 are considered as continuous (as they at a rather high frequency), the dropping in task 3 is triggered in specific positions. The topic gets dropped once a minimum of the velocity profile has been reached, meaning around 33%, 66% and 100%.

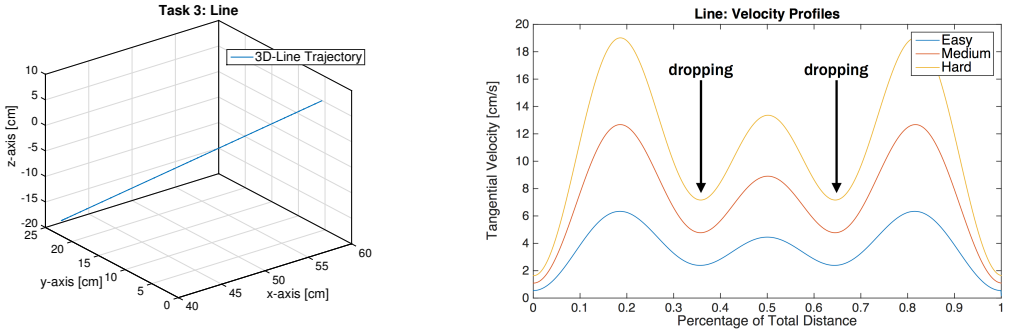



Figure 2.20: Plots of Trajectory and Velocity Profiles of Task 3

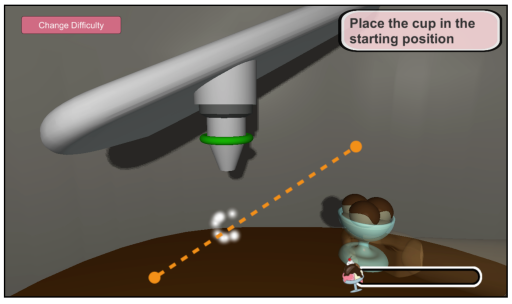



Figure 2.21: Trajectory 3 in the game

## 2.4 Communication between Game and Controller

A key point of this thesis was to implement a working communication between the game environment and the Simulink model, containing the controller and the trajectory calculations. While the game is running as en ".exe" file on a host PC of the ARMin setup, the Simulink model gets compiled and is running in a real-time environment on an xpc-target, which is connected through drivers to the motors of the robot. As it can be seen in figure 2.22, the Target PC is waiting for instructions about the current task number and a trigger to enable the control. After that, the trajectory is generated and the needed control forces are calculated and send to the motor. At the same time, the Simulink model is monitoring each motor position with the help of the encoders, and calculates the actual position of the end-effector with the inverse kinematics. These have to

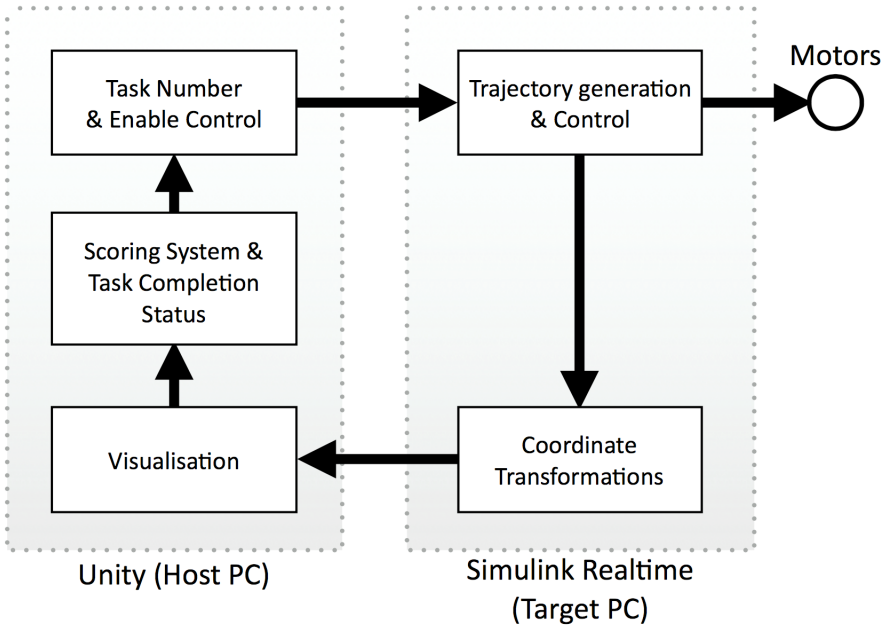



Figure 2.22: Schematic Interpretation of the Communication Process

be transformed into a coordinate systems that maps perfectly onto the one in the game. This is done with a serious of matrix transformations that are described in section 2.4.2. The transformed actual end-effector position as well as the transformed desired position of the end-effector are then send back to the host Pc and the game. There, the appropriate visualizations are made, meaning that the distributors are placed in the desired position and the cup is visualized at the actual postion. Further a scoring system consisting in a progress bar on the bottom left of the screen is rendered according to the amount of ingredients caught by the subject. After a certain amount of time (in our case 3 minutes), the next task is initiated and the process starts all over again.

### 2.4.1 UDP-Protocol

All the communication is done via UDP-Protocol, which is an internet protocol that works in a usual internet network environment. You can send data to specific IP-Addresses or to all members of the network. Data is converted to doubles and packed as a byte package. Then, it is send and unpacked by the receiver. This protocol is often used by computer programs, however it does not ensure delivery, ordering or duplicity. In our case, the Host PC as well as the Target PC function as senders and receivers at the same time. The game running on the Host PC, is running a separate thread to receive the following data from the Target PC with a frequency of 100 Hz:

- $\overrightarrow{r}_{des} = [x_{des}, y_{des}, z_{des}]$
- $\overrightarrow{r}_{act} = [x_{act}, y_{act}, z_{act}]$

These are the position vectors of the current desired position on the trajectory $\overrightarrow{r}_{des}$ and the actual position of the end effector of the ARMin $\overrightarrow{r}_{act}$.

At the same time, the game is sending the following data every frame that is rendered, which is around 50 Hz when no other tasks need a lot of processing power:

- Current trajectory number (e.g. 1,2,3 or -1 for no trajectory)
- Difficulty selected in the main menu
- Main controller mode (e.g. HG, EA or Mixed)
- Sub-controller mode (e.g. constant, exponential or error based decay)
- Force Controller Enabled/Disabled variable (e.g. 1 or 0)

### 2.4.2 Coordinate Transformations needed for matching the Different Coordinate Frames

Before the desired and the actual position of the end-effector can be send to the Game, a series of transformations are necessary to map the coordinates to the coordinate system of the game. An important point to mention is that the trajectories in section 2.3 are displayed in a coordinate system that is turned 45° around the the $z$-axis in relation to the ARMin coordinate system. Furthermore, the trajectories in 2.3 are centered around the origin for simplicity while in reality the trajectories have an offset in order to be, again, in the optimal workspace of the ARMin. Figure 2.23 shows the trajectory in ARMin coordinates in the case of the ellipsoid.

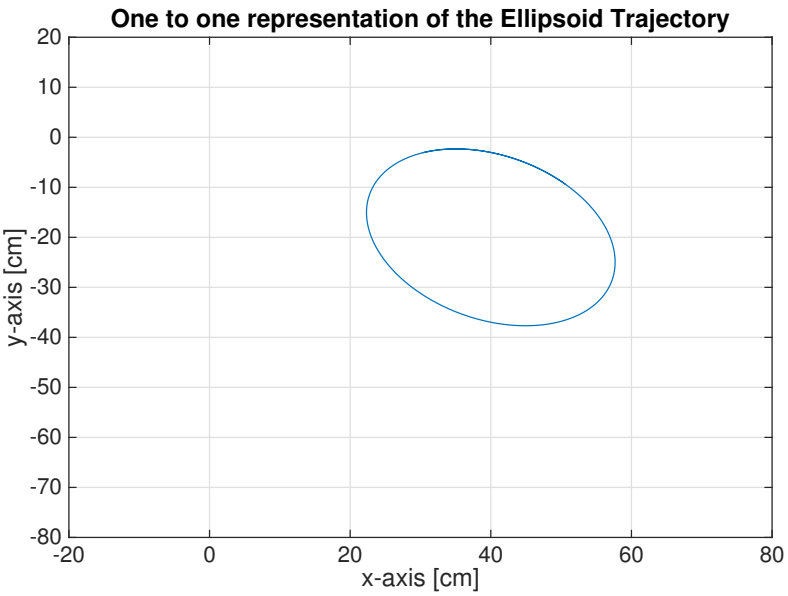



Figure 2.23: Ellipsoid Trajectory Used in the ARMin

The following transformation steps were applied to the desired and actual end-effector position:

1. The first step is to centre the trajectories (e.g. ellipsoid, circle, and line) to have everything referenced to a point. This way every trajectory can be treated the same in the following steps. This step is done by a simple translation of the vector (in cenitmeters)

$$\overrightarrow{T} = [-42.42, -14.14, 10]^T \tag{2.15}$$

2. The second step is to rotate the coordinates by $\alpha = -45°$ or $\alpha = 45°$ (depending on the arm,which the subject is using) around the z-axis because the optimal workspace of the ARMin is rotate by $\pm 45°$ in relation to the coordinate system. Therefore coordinate vectors are multiplied by the following matrix:

$$R_z = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.16}$$

3. As a third step, the $y$ and $z$ axes are swapped because Simulink uses a right handed coordinate system, whereas Unity uses a left handed coordinate system. This is done by a multiplication with the following swap matrix:

$$S_{y-z} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{2.17}$$

4. After that, a task specific rotation is done. This is due to the fact that the visualization takes place in the absolute coordinate system of the game. Because the rotation of the player is dependent on the task, the visualization is dependent on that rotation. Therefore, a rotation of $\beta = f(i)$ in function of the task number i=1,2,3, around the $z$-axis, is needed as a last step before sending the coordinates. $f(i)$ is defined as follows:

$$\begin{cases} f(1) = -90° \\ f(2) = 0° \\ f(3) = 180° \\ f(-1) = 90° \end{cases} \tag{2.18}$$

The case of $i = -1$ corresponds to the moment where the player looks out of the ice cream truck and no trajectory is enabled. Further the case specific rotation matrix looks as follows:

$$R_{spec} = \begin{pmatrix} \cos\beta(i) & -\sin\beta(i) & 0 \\ \sin\beta(i) & \cos\beta(i) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.19}$$

The final resulting transformation can be expressed as follows:

$$\overrightarrow{r}_{trans} = R_{spec} \cdot S_{y-z} \cdot R_z \cdot (\overrightarrow{r}_{init} + \overrightarrow{T}) \tag{2.20}$$

# Chapter 3

# Results and Discussion

## 3.1 Results

To begin with, I had to check if the trajectories and the velocity profiles are tracked correctly if the different gains have values near to the initial ones. Figure 3.1 shows the tracking of the trajectory as well as the tracking of the velocity profile of the first task. The $z$-coordinate is not shown because it is not of much importance in task 1. Figure 3.2 shows three passes of the 3D Line trajectory and the tracked velocity profile. All these plots were made with gains that are above 90% of their initial values.

To test if everything is working as expected, I made a pilot study with myself as subject to test task 1. In this way I could see if the gain reduction laws work as expected and the robot is behaving in a stable manner. In figure 3.3 shows the trend of the gain percentage in the case of the constant gain decay rule as well as the errors in the 3 dimensions. The same information can be seen in figure 3.4 for the exponential decay rule. Greater errors at a late stage of the test were made on purpose to check if it is possible to make errors, once the gains are nearly fully faded.

For the error based decay rule (AAN), I first tried with a value for the forgetting factor of $f = 0.95$ and for the robot learning gain a value of $g = 0.005$. The results with these values can be seen in figure 3.5.

A second pilot study, with the values as described in section 2.2.3.3, was done by my supervisor. This trial consisted of task 1 and 3 and the assisted-as-needed controller. The duration of each task was 3 minutes. The results can be seen in figure 3.6 and 3.7. The peak in task 3 around second 80 is probably due to a measuring or calculation error.

Furthermore, the functioning of the game was tested rigorously until all bugs were spotted and everything was working fine.

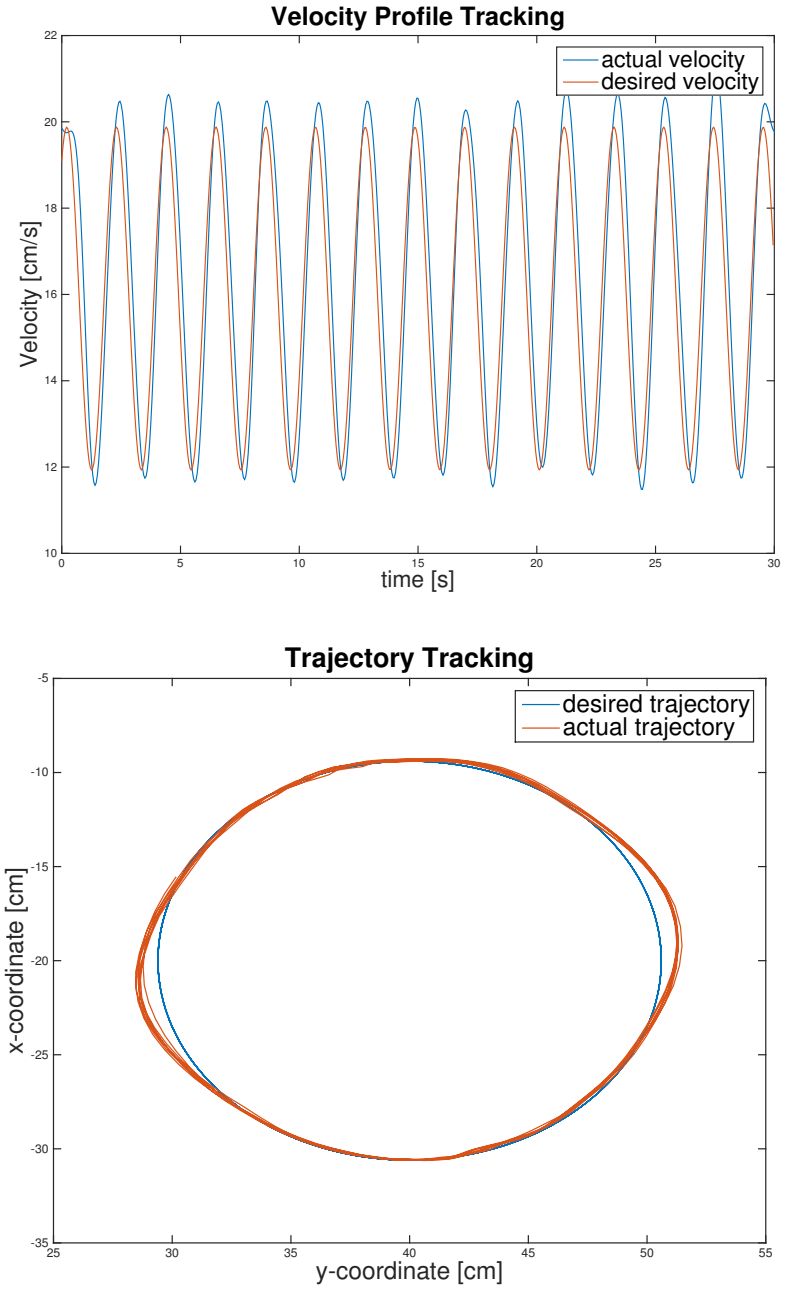



Figure 3.1: Trajectory and Velocity Profile Tracking of Task 1

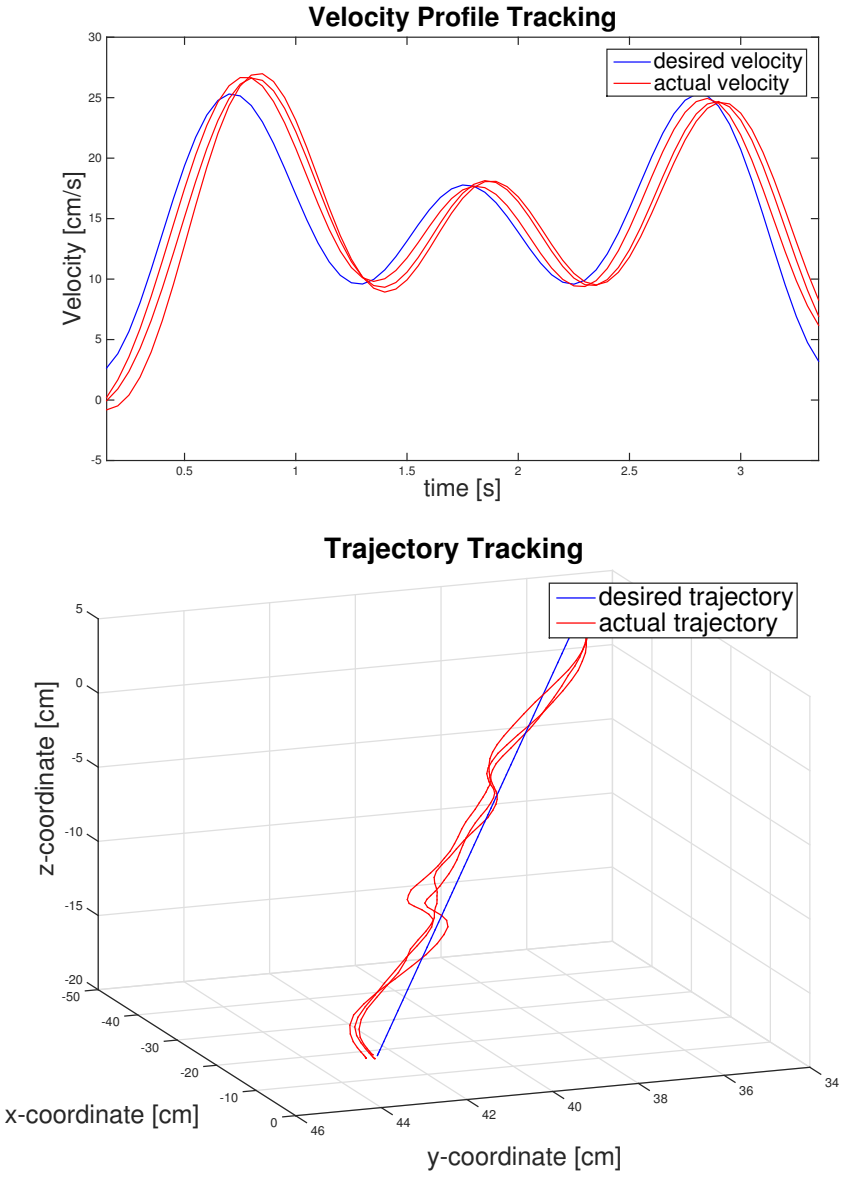



Figure 3.2: Trajectory and Velocity Profile Tracking of Task 3

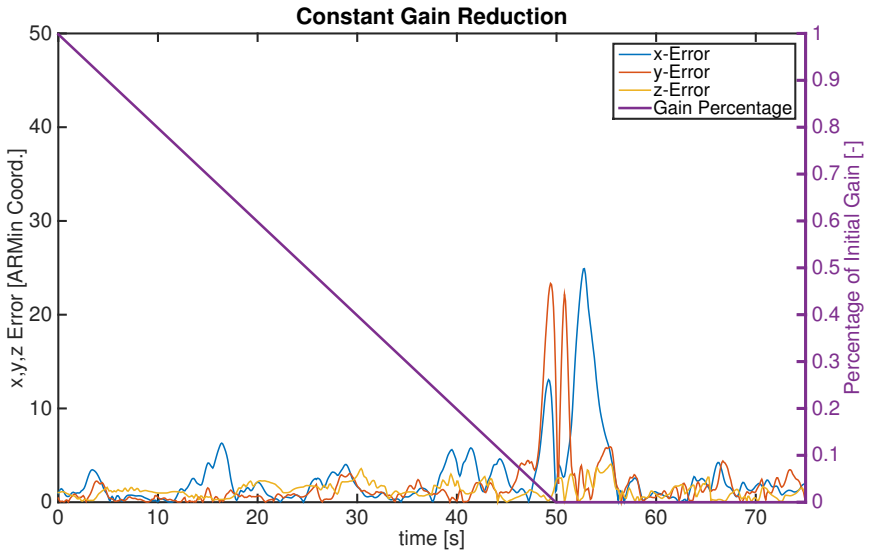



Figure 3.3: Constant Gain Reduction

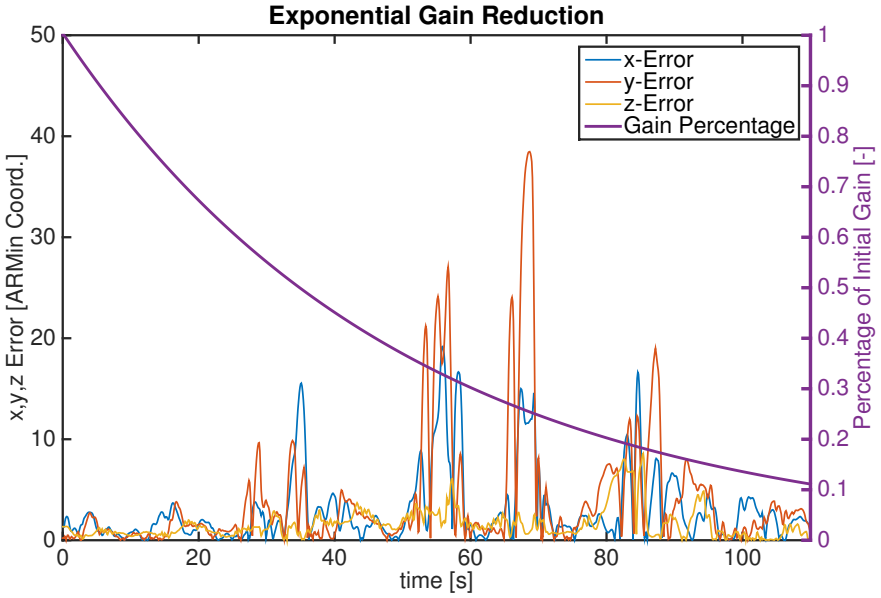



Figure 3.4: Exponential Gain Reduction

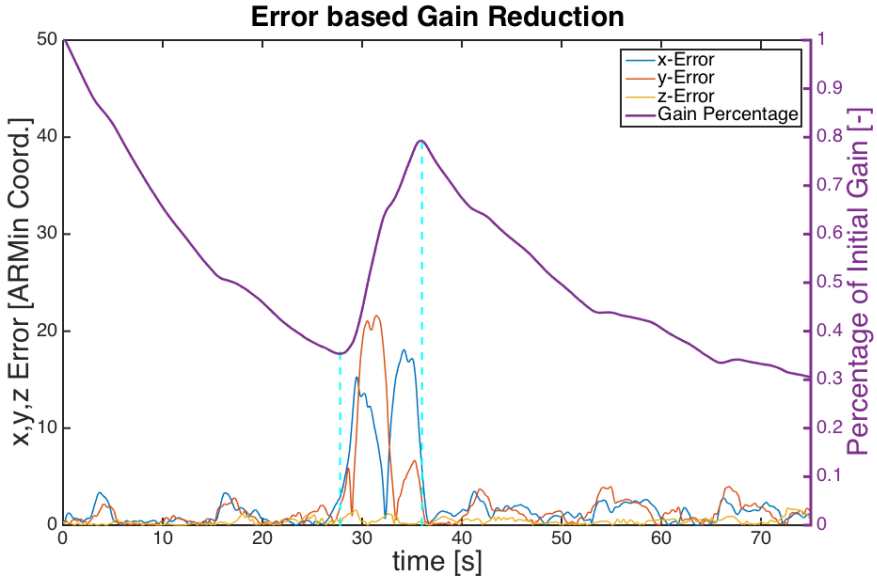



Figure 3.5: Error based Gain Reduction

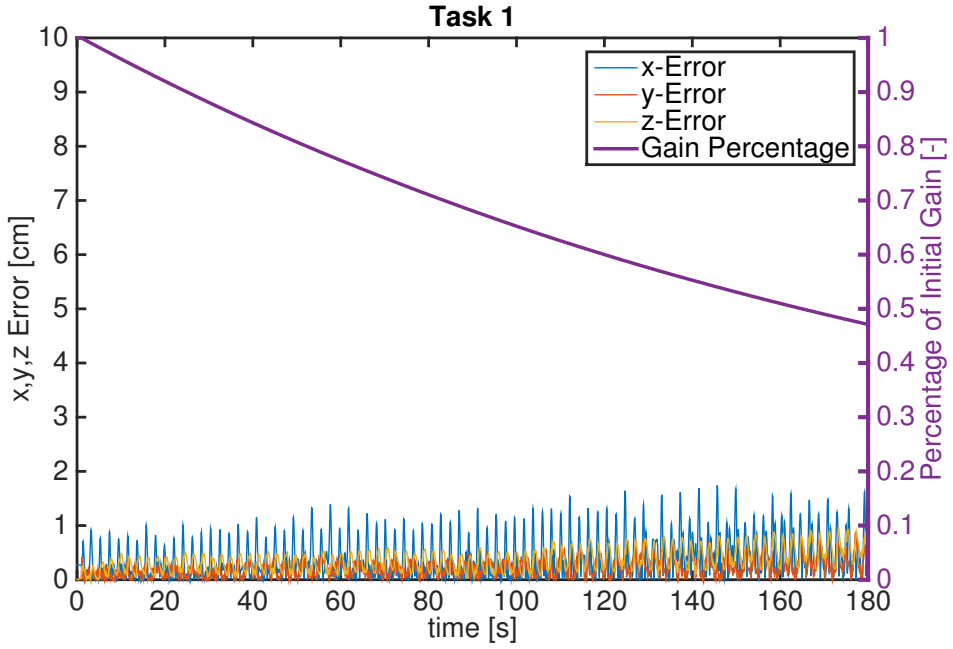



Figure 3.6: Second Pilot Study, Task 1

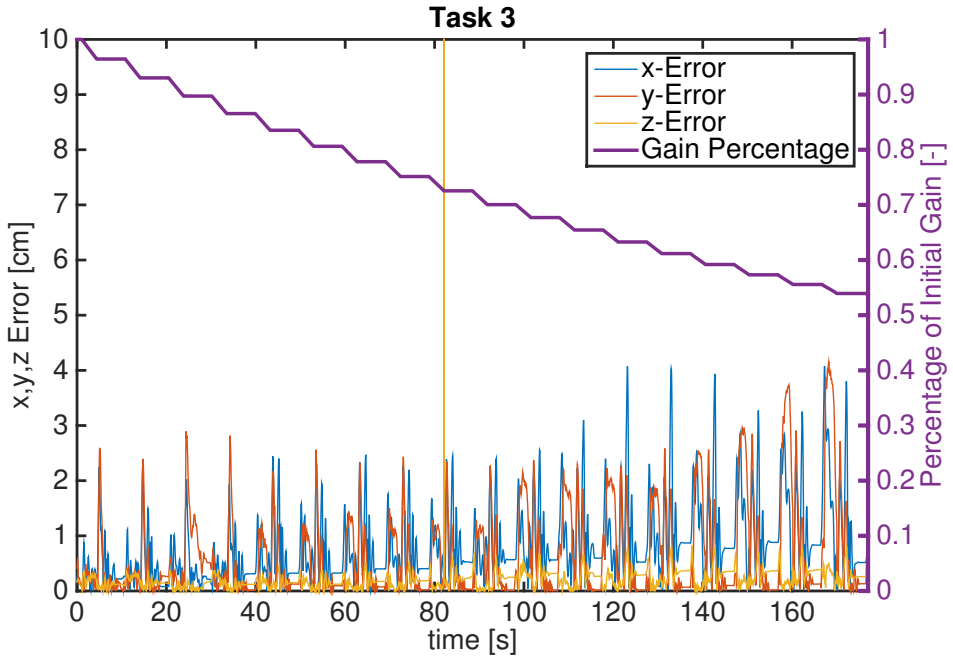



Figure 3.7: Second Pilot Study, Task 3

# Chapter 4

# Discussion, Conclusion and Outlook

## 4.1 Discussion

Figures 3.1 and 3.2 as well as figures 3.3 and 3.4 show that proper guidance prevents one from making too many errors in the early stage of a task and the velocity profile as well as the trajectory are followed correctly. After the gains faded to less than a third, it was easier to make errors, which is positive because, as pointed out in chapter 1, errors may be beneficial in motor learning.

Furthermore, in the case of the error-based decay or assisted-as-needed controller, figure 3.5 shows that the trend of the gain percentage is similar to an exponential curve while the error is small. The reaction of the controller when the errors increase are quite immediate, which can be underlined by the vertical doted lines in figure 3.5, which are limiting the area with a bigger error.The exponential trend kicks back in once the errors are minor again.

Another characteristic of the results in every case is the periodicity of the error patterns. This can be seen best in the second pilot study by my supervisor, due to the longer task time. The peaks in the $x$-direction clearly show that some parts in the trajectory are harder to track then others. Moreover, the fact that the error in $x$-direction is more pronounced than those in the other dimensions seems logic because the most movement for task 1 and 3 is in the $x$-direction. This would probably change if you logged data from fulfilling task 2.

Concerning the results of task 3 in figure 3.7, it is interesting to see the constant parts of the gain percentage curve. This is due to the fact that the gains are held constant during the transition from one pass of the discrete movement to the next one. This was done in order to take into consideration only the performance during the task itself.

## 4.2 Summary

In my studies on mechatronics, I learned that a lot of different methods for controlling rehabilitation devices can be used in different cases. I looked into some resistive methods that keep the subject's attention due to repeated disturbance of the desired movements. Moreover, I looked int HG feedback methods, which can be very useful in early stages of learning as well as for complicated

tasks. Further, HG can be useful to improve timing aspects more than spatial aspects. This could be the exact opposite in the case of EA. To prove the hypothesis, a testing environment was implemented that would keep the motivation of the subject up, by distancing him from the idea of fulfilling a boring tracking task. This was done by using the game engine and game developer environment called Unity 5.0. Other programs used in the process of development were Blender, NX-Siemens and Photoshop. In a later step, I implemented an assisted-as-needed controller in Simulink, which is based on a PD-controller in end-effector space combined with a force controller in joint-space. This resulted in a transparent impedance controller for the 7 DoF upper limb rehabilitation robot ARMin. The communication between the robot, namely the xpc-target, and the game running on the Host PC, was established with a UDP protocol. Some pilot studies were done to check the functioning of the game, controller and communication, especially focusing on the gain decaying rules. An EA-based controller was implemented from Tanja Baumann and was adapted to function with the game. After testing, everything is set up for upcoming studies to change the parameters in the game, based on the criteria of a future study.

## 4.3 Conclusion

Everything is set up to test the hypothesis, cited in chapter 1, and to do basic research on motor learning. Optimizing changes could always be made on the side of the game as well as the controller but all necessary functions are implemented and should be sufficient to form an adequate testing environment. Having tested all the functioning, the only thing that is left is to fine tune the parameters of the task (e.g. task time, robot forgetting factor, robot learning gain etc...)

## 4.4 Outlook

If the hypothesis holds, the results could be used to design novel rehabilitation set ups for the ARMin, which could be used in different partner hospitals. The physiotherapists could treat a multiple number of patients without too much physical effort.

Moreover, the game concepts described in section 2.1.1 that were not used, could be implemented by other students to get a variety of games that would make rehabilitation more interesting for patients.

# Appendix A

# Appendix: The Ice Cream Maker

## A.1 Scenes

Scenes are the places where all the game objects are contained. If you load a new scene, every game object in the scene before will get erased. The game "The Ice Cream Maker" has two scenes:

- Main Menu Scene (containing the Main Menu and the advanced Settings)
- Main Scene (containing all the game content)

Pressing the Start Button in the main Menu calls a function Application.LoadLevel(); that loads the Main Scene and destroys the MainMenu Scene. Now all game objects in the Main Scene and their scripts are initialized and those in the Main Menu Scene are destroyed.

## A.2 Scripts

### A.2.1 Introduction

Scripts are the Main driving engine of the game. All in-game-movements, game switches and events are handled by them and mostly updated every newly calculated frame. The basic structure of a C Sharp MonoBehaviour Script (nearly all of the scripts in Unity are of that type) is as follows:

- Start(); is called before the first frame of a new Scene.
- Update(); is called every new displayed frame.
- LateUpdate() is used mostly for Camera Movements to reduce Lag.

Other class methods can, obviously, be implemented by the programmer and can be called from the other functions. Most of the time, variables are defined before the Start() function in the class scope. Have a look at the example below that initializes a variable i at the start, sets it to zero, displays the value in the developer console every update and increments it.

```
using UnityEngine;
using System.Collections;

public class NewBehaviourScript : MonoBehaviour {

    int i;


    // Use this for initialization
    void Start () {

        i=0;

    }

    // Update is called once per frame
    void Update () {

        Debug.Log(i);
        i++;

    }
}
```

### A.2.2 Script Listing

| **Script Name** | **Gaemobject** | **Purpose** |
|---|---|---|
| "MainMen" Scene | | |
| GameMaster.cs | GameMasterObject | – Handles the gameplay flow<br>– Sets up all the difficulties and modes once the player has pressed the "Start" button |
| MainMenuButtons.cs | Main | – Contains all the functions called by buttons<br>– Reads all the preferences chosen by the player, saves them and sets them to be sent by UDP Protocol. |
| "MainScene" Scene | | |
| CameraController.cs | Main Camera | – Takes the Camera positions and rotations as input in the editor<br>– Triggers the Camera movements |

| | | |
|---|---|---|
| Completion.cs | Toppic Trigger<br>Sauce Trigger<br>IC Trigger | – Increases the filling state if all conditions are met<br>– Turns the cup green if particle was caught<br>– Turns the sparkling green if in the right zone |
| EmitterControl.cs | Distributor_3D | – Controls the emitting of the topics as a function of the current percentage of the total distance |
| ICTrigger.cs | ReceiveTrigger | – Handles the finishing sparkling spot<br>– Hands the cup over to the kid |
| KidController.cs | kid | – Controls the animations of the kid<br>– Moves it around accordingly |
| MenuNavigation.cs | EventSystem | – Contains all the functions called by the buttons |
| OrderActivation.cs | order canvas | – Activates/Deactivates the order message when needed |
| PlayerController.cs | fullarm_Joint | – Checks WSAD to control the arm in UDP-disabled mode<br>– Checks WSAD to fine-tune the arm offsets in UDP-enabled mode |
| ProgressBar.cs | Filling | – Visualises the current filling amount |

| | | |
|---|---|---|
| TextMessage.cs | UI | – Changes the text on the upper right<br>– Triggers the timer<br>– Triggers the current Distributor to get activated |
| UDPInput.cs | UdpMaster | – Handles everything that has to do with packing, unpacking, receiving and sending via UDP Protocol<br>– Logs the needed data<br>– Sets the In-game positions of Distributors and arms<br>– Sets everything according to the right side |

Table A.1: Script Description

### A.2.3 Gameobject Listing

| Gameobject | Description |
|---|---|
| "MainMenu" Scene | |
| GameMasterObject | The object that contains the script which controls most of the game flow |
| MainCamera | The principal Camera in the Main Menu Scene |
| MainMenu: | Parent gameobject containing all the main menu features (e.g. buttons, text, etc...) |
| -Difficulty | Gameobject that hosts a toggle group |
| -Start Button | Button containing the start trigger and the start button text |
| -Select Text | Text beneath the Title |
| -Hard | Toggle for the Hard Difficulty |
| -Easy | Toggle for the Easy Difficulty |
| -Medium | Toggle for the Medium Difficulty |
| -Title | Title text |
| -Cup Left/Right | Cup images |
| -Visual Feedback | Toggle for the visual Feedback |
| -AdvSettBtn | Button to go to the advanced settings |
| -ExitButton | Button to exit the game |
| -Left/Right | Toggles for the side selection |
| -Side Text | Text for the side selection |
| -Side | Gameobject hosting the side toggle group |
| -EventSystem | Automatically created gameobject |
| _Main | Gameobject hosting the script containing all the button actions |
| AdvMenu: | Parent containing all the advanced Menu features |

| | |
|---|---|
| -ControlModeToggleGroup | Gamobject containing the toggle group for the control selection |
| -Title | Title Text |
| -ICSett | Speed setting for the Icecream |
| -SauceSett | Speed setting for the Sauce |
| -TopicSett | Speed setting for the Topic |
| -Submit | Submit Button |
| -ControllerModeText | Text for control selection |
| -Haptic Guidance | Toggle for HG |
| -E.A.Hybrid | Toggle for EA Hybrid |
| -Error Amplification | Toggle for EA |
| -Force Control | Contains Toggles for enable/disable Force Control |
| -ZeroForceControl | Toggle for Zero Force Controller |
| -Task Selection | Contains toggles for task selection |
| ‘MainMenu” Scene | |
| cups | Parent containing all the decoration cup samples in the game |
| Distributors: | Parent containing all the distributors |
| -Ditributor_3D | Containing all 3 particle emitters (3 diff.) and the trigger beneath for task 3 |
| -Ditributor_Hor | Containing all 3 particle emitters (3 diff.) and the trigger beneath for task1 |
| -Ditributor_Ver | Containing all 3 particle emitters (3 diff.) and the trigger beneath for task2 |
| Environment | Contains all decoration objects (e.g. Tree, Road, Gras, House etc...) |
| FinUI | Contains all features of the finishing UI |
| keypoints | Contains all keypoint colliders for the kid movement |

| | |
|---|---|
| kid | Contains the kid armature and the ordering bubble |
| Lightning | Contains a main, fill and point light |
| Main Camera | The Main Camera in the Main Scene |
| ReceiveTrigger | Trigger collider to hand back the cup to the kid |
| UdpMaster | Hosts the script that enables Udp communication |
| UI | Contains all the UI features (e.g. progressbar, main menu button, information window |
| Van | Contains all the models of the van |
| WindZone | Wind generator gameobject |

Table A.2: Gameobject Description

# Bibliography


[1] J. Bluteau, S. Coquillart, Y. Payan, and E. Gentaz. Haptic guidance improves the visuo-manual tracking of trajectories. *PLoS ONE*, 3(3):e1775, 03 2008.

[2] L. L. Cai, A. J. Fong, C. K. Otoshi, Y. Liang, J. W. Burdick, R. R. Roy, and V. R. Edgerton. Implications of assist-as-needed robotic step training after a complete spinal cord injury on intrinsic strategies of motor learning. *The Journal of Neuroscience*, 26(41):10564–10568, 2006.

[3] P. Dane and O. Marcia, K. The task-dependent efficacy of shared-control haptic guidance paradigms. *Haptics, IEEE Transactions*, 5(3), 2012.

[4] J. E. Duarte and D. J. Reinkensmeyer. Effects of robotically modulating kinematic variability on motor skill learning and motivation. *J Neurophysiol*, 113(7):2682–91, Apr 2015.

[5] J. E. Duarte and D. J. Reinkensmeyer. Effects of robotically modulating kinematic variability on motor skill learning and motivation. *Journal of Neurophysiology*, 113(7):2682–3691, April 2015.

[6] J. L. Emken, R. Benitez, and D. J. Reinkensmeyer. Human robot cooperative movement training: Learning a novel sensory motor transformation during walking with robotic assistance-as-needed. *Journal of Neuroengineering and Rehabilitation*, 4(8), 2007.

[7] M. A. Guadagnoli and T. D. Lee. Challenge point: A framework for conceptualizing the effects of various practice conditions in motor learning. *Journal of Motor Behavior*, 36(2):212–224, 2004. PMID: 15130871.

[8] K. Kornysheva and J. Diedrichsen. Human premotor areas parse sequences into their spatial and temporal features. *eLife*, 3, 2014.

[9] H. Lee and S. Choi. Combining haptic guidance and haptic disturbance: an initial study of hybrid haptic assistance for virtual steering task. In *Haptics Symposium (HAPTICS), 2014 IEEE*, pages 159–165, Feb 2014.

[10] J. Lee and S. Choi. Effects of haptic guidance and disturbance on motor learning: Potential advantage of haptic disturbance. pages 335–342, 2010. cited By 32.

[11] P.-J. Maes, M. Wanderley, and C. Palmer. The role of working memory in the temporal control of discrete and continuous movements. *Experimental Brain Research*, 233(1):263–273, 2015.

[12] L. Marchal-Crespo, S. McHughen, S. C. Cramer, and D. J. Reinkensmeyer. The effect of haptic guidance, aging, and initial skill level on motor learning of a steering task. *Experimental Brain Research*, 210:209–220, 2010.

[13] L. Marchal-Crespo and D. J. Reinkensmeyer. Haptic guidance can enhance motor learning of a steering task. *Journal of Motor Behavior*, 40(6):545–557, 2008. PMID: 18980907.

[14] L. Marchal-Crespo, P. Wolf, N. Gerig, G. Rauter, L. Jaeger, H. Vallery, and R. Riener. The role of skill level and motor task characteristics on the effectiveness of robotic training: first results. 2015.

[15] M.-H. Milot, L. Marchal-Crespo, C. Green, S. Cramer, and D. Reinkensmeyer. Comparison of error-amplification and haptic-guidance training techniques for learning of a timing-based motor task by healthy individuals. *Experimental Brain Research*, 201(2):119–131, 2010.

[16] P. Morasso, M. Casadio, and V. Squeri. Haptic training for a visuomotor fetch amp; pursue task. In *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, pages 4133–4136, July 2013.

[17] T. Nef, M. Mihelj, G. Colombo, and R. Riener. Armin - robot for rehabilitation of the upper extremities. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3152–3157, May 2006.

[18] G. Rauter, R. Sigrist, L. Marchal-Crespo, H. Vallery, R. Rober, and P. Wolf. Assistance or challeng? filling a gap in user-cooperative control. *IEEE*, September 2011.

[19] G. Rauter, R. Sigrist, R. Riener, and P. Wolf. Learning of temporal and spatial movement aspects: A comparison of four types of haptic control and concurrent visual feedback. *Haptics, IEEE Transactions on*, PP(99):1–1, 2015.

[20] R. A. Scheidt, D. J. Reinkensmeyer, M. A. Conditt, W. Z. Rymer, and F. A. Mussa-Ivaldi. Persistence of motor adaptation during constrained, multi-joint, arm movements. *Journal of Neurophysiology*, 84(2):853–862, 2000.

[21] R. Schmidt. Frequent augmented feedback can degrade learning: Evidence and interpretations. In J. Requin and G. Stelmach, editors, *Tutorials in Motor Neuroscience*, volume 62 of *NATO ASI Series*, pages 59–75. Springer Netherlands, 1991.

[22] R. Sigrist, G. Rauter, R. Riener, and P. Wolf. Augmented visual, auditory, haptic, and multimodal feedback in motor learning: a review. *Psychon Bull Rev*, 20(1):21–53, Feb 2013.

[23] F. Tamar and H. Neville. The coordination of arm movements: An experimentally confirmed mathematical model. *The Journal of Neuroscience*, 5(7):1688–1703, July 1985.

[24] K. A. Thoroughman and R. Shadmehr. Learning of action through adaptive combination of motor primitives. *Nature*, 407:742–747, October 2000.

[25] H. Vallery, M. Guidali, A. Duschau-Wicke, and R. Riener. Patient-cooperative control: Providing safe support without restricting movement. *IFMBE Proceedings*, 25(9):166–169, September 2009.

[26] Y. Wei, J. Patton, P. Bajaj, and R. Scheidt. A real-time haptic/graphic demonstration of how error augmentation can enhance learning. *IEEE*, April 2005.

[27] C. Zilles and J. Salisbury. A constraint-based god-object method for haptic display. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 146–151 vol.3, Aug 1995.