CHALMERS

# Prosthetic Visualisation for Training and Rehabilitation using a Virtual Reality Environment

*Developing an interface of communication between a real-time pattern recognition software and a virtual reality environment.*

NICHLAS SANDER

Department of Biomedical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, 2012
Report No. EX015/2012

Prosthetic Visualisation for Training and Rehabilitation using Virtual Reality Environment
NICHLAS SANDER



Master of Science Thesis 2012
Department of Signals and Systems
Division of Biomedical Engineering

Chalmers University of Technology
SE-412 96 Göteborg
Sweden

Supervisor
Max J. Ortiz Catalan

Examinator
Yngve Hamnerius
Department of Signals and Systems

## Abstract

Integrum AB use a real-time pattern recognition system for accurately controlling a hand prosthesis using myoelectric signals. Recent work has been made at Integrum AB to create a Virtual Reality Environment which can be controlled with a sensor glove, aimed to be used during training of a pattern recognition algorithm. This Virtual Reality Environment is solely used along with the sensor glove, something which undermines its true potential. A new training and rehabilitation system can be created by using and modifying pre-existing developments at Integrum AB.

This thesis covers the implementation of a training and rehabilitation system by modifying and using a Virtual Reality Environment and a pattern recognition algorithm. Changes are made to the virtual reality system to support inputs, controlling the movements of the virtualisation, via a network socket interface. The interface is compatible with any software that supports socket communication, meaning it can be used together with any type of outputs. A Target Achievement Control Test was implemented into the system to test the patients control accuracy, efficacy and speed.

Implementations were made in C++ and MATLAB environments to create an interface between the pattern recognition system and the Virtual Reality Environment. Tests showed that using a well-trained pattern recognition algorithm it was possible to recreate adequate movements with good accuracy. The Target Achievement Control Test proved not too difficult to complete and provided the user with an efficacy percentage and completion time, giving the user a possibility to perform better and therefore improve.

**Keywords:** Virtual Reality Environment, Prosthesis, Prosthetic Training, Prosthetic Rehabilitation, TAC Test

## Acronyms

| | |
|---|---|
| ANN | Artificial Neural Network |
| AR | Augmented Reality |
| DoF | Degree of Freedom |
| EA | Error Augmentation |
| EMG | Electromyography |
| ENG | Electroneuronography |
| IP | Internet Protocol |
| IPC | Interprocess Communication |
| LDA | Linear Discriminant Analysis |
| MAV | Mean Absolute Value |
| PRA | Pattern Recognition Algorithm |
| SEMG | Surface EMG |
| SSC | Slope Sign Changes |
| STFT | Short Time Fourier Transform |
| TAC | Target Achievement Control |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |
| VR | Virtual Reality |
| VRE | Virtual Reality Environment |
| WL | Waveform Length |
| ZC | Zero Crossings |

# Contents

**1 Introduction** **1**

1.1 Background . . . . . 1

1.2 Aim . . . . . 3

1.3 Restrictions . . . . . 3

1.4 Problem . . . . . 3

1.5 Scope . . . . . 3

**2 Amputation** **5**

2.1 Statistics . . . . . 5

2.2 Causes . . . . . 5

**3 Prosthesis** **6**

3.1 Passive Prosthesis . . . . . 6

3.2 Body-Powered Prosthesis . . . . . 6

3.3 Externally-powered Prosthesis . . . . . 7

**4 Flow of Prosthetic Movements** **8**

4.1 Signal acquisition . . . . . 8

4.2 Signal processing and Feature Selection . . . . . 8

4.3 Pattern Recognition . . . . . 8

4.3.1 Offline Classification . . . . . 9

4.3.2 Online Classification . . . . . 9

4.4 Prosthetic Device . . . . . 9

**5 Training of a Pattern Recognition System** **10**

5.1 Using VRE as a Visual Aid during Training . . . . . 10

**6 Post-Training Utilities** **11**

6.1 TAC Test . . . . . 11

6.2 Rehabilitation Aid . . . . . 11

**7 Method** **13**

7.1 Means of Implementation . . . 13

7.2 Virtual Reality Environment . . . 13

7.2.1 Hand Model . . . 14

7.2.2 Degrees of Freedom . . . 14

7.2.3 Poses . . . 15

7.3 Communication . . . 17

7.3.1 Using Windows Socket Interface . . . 17

7.3.2 Socket Implementation . . . 19

7.4 Communication Protocol . . . 20

7.4.1 Movements . . . 21

7.4.2 Configuration Variables . . . 21

7.4.3 Resetting hand position . . . 22

7.5 TAC Test . . . 22

7.5.1 Virtual Reality Environment . . . 22

7.5.2 MATLAB . . . 22

**8 Results** **24**

8.1 Controlling the VRE . . . 24

8.1.1 Myoelectrical Control . . . 24

8.1.2 Control Without Pattern Recognition System . . . 26

8.2 Implementation of TAC Test . . . 27

**9 Discussion of Results** **29**

**10 Future Work and Improvements** **30**

**References** **30**

**A Commands for Implemented Protocol** **34**

A.1 Commands for movements . . . 34

A.2 Commands for configuration . . . 35

A.3 Commands for resetting the hand position . . . 35

# 1 Introduction

## 1.1 Background

During the year 2005 there was an estimated 1.6 million people living in the US with some form of limb loss. In the year 2050 this number is estimated to have risen to over 3.6 million [1].

Becoming an amputee can be very difficult and frustrating due to limitations in everyday tasks. When losing a leg, a sufficient replacements can be summarised as a device which enables the user to walk in a somewhat normal fashion, however, a more natural appearance and movement is preferred. In upper-limb amputation there are further complications due to the increase in the amount of degrees of freedom (DoF) associated with the limb loss. Not being able to use both arms result in substantial setbacks in everyday life. Prosthetic solutions have been presented to aid those with upper-limb amputations. In this report electrical prosthetics will be focused upon due to its relevance to the topic.

#### Electrical Prosthetics

Electrically powered prosthesis present an advantage for amputees, allowing for realistic movements and an increase in the amount of DoF along with the possibility to control the prosthesis via natural bioelectrical signals through muscles or nerves, rather than moving the hand purely mechanically. Control systems for the electrically powered prosthesis usually infer myoelectrical signals [2], powered by the intensity of the signals received from electromyography (EMG) signals retrieved on the surface of the skin above the remaining musculature [3]. In a more favourable case the patient has suffered from a short transradial amputation and still possess the musculature to control the prosthesis with muscles naturally associated with the movement. If, on the other hand, the patient has suffered a transhumeral or higher amputation, too much muscle may have been lost in order to fully control the prosthesis with the appropriate muscles. Kuiken et al. [4] have developed a method called Targeted Muscle Reinnervation where the neural ends are remapped into suitable muscle tissue which remain after the amputation, for instance the pectorals in the case of a shoulder disarticulation. Using this method, neural signals associated with the movements of the lost limb will trigger muscle contractions in the remapped muscle tissue. The contractions are then picked up using surface EMG (SEMG), something which is more easily performed compared to an electroneuronography (ENG) [4]. Limitations apply to the amount of space possible for retrieving data from the muscle and also in the amount of nerves which are possible to be remapped. If too many nerves are mapped to close to one another there might be misinterpretation of the data and the signals might get mixed up.

Even though ENG may prove difficult to acquire, since a lot of surrounding tissue and muscle cause huge amounts of noise, it does provide more accurate information about the wanted movements. In an article Ortiz-Catalan et al. explore the viability of using neural signal as a solution for neuroprosthetic control, retrieved using neural electrodes such as the cuff electrode [5]. The results look promising and in the article they discuss this solution as a viable option over SEMG.

Using a neural interface with prosthetics present additional features in the form of neural sensory feedback. In the case of Kuiken et al. the patient is able to feel sensations in the amputated limb via stimulation of the reinnervated skin [4]. Although the sensations are extremely random and unpredictable, it is still a considerable step towards achieving some form of sensory feedback. Other studies also

show that stimulating peripheral nerve endings can generate a sensation of stimulation to the non-existent limb [6].

### Training

In order to ensure accurate control of a prosthesis some substantial training of the system is needed, both in the case of myoelectrical prostheses and neuroprostheses. Training is achieved by feeding a pattern recognition algorithm with an input that has a known output, the system then trains to be able to match that input, and similar ones, to the corresponding output. A perfectly trained pattern recognition algorithm (PRA) would always be able to determine the output from any given input, this is most often not the case.

Training the PRA is an important phase in the case of prosthetic movements, to ensure that correct movements are performed the algorithm must receive accurate input data and be supplied with enough different cases to learn from a broad spectrum of cases. To assist the patient in producing the correct movement some demonstration might be effective, this can be achieved by the patient performing the same movement with both their hands at the same time, another alternative is to provide an instructional movement displayed at the moment when the user is asked to perform the movement.

### Rehabilitation

After an amputation, patients may suffer from muscular degeneration where the muscles that are no longer used are broken down. Apart from this, there may also be remapping, or reorganization, of the somatosensory cortical map [7]. Brain-functionality that was connected to no longer used movements will be remapped to smaller parts of the brain, where the possibility for finer movements may be lost. This functionality may be regained through careful rehabilitation, and at the same time rebuilding lost muscle strength.

In patients using either targeted reinnervation or myoelectrical control systems to control a prosthesis the motor control is a necessity. Mirelman et al. showed in a recent study, [8], that using a Virtual Reality Environment (VRE) coupled with a robotic prosthesis improved the patients walking ability more than training with a robotic prosthesis alone. Using this information one can see that it is very important to have a rehabilitation environment for the patients, which encourages training and makes training more approachable. Similar scenarios arise in stroke patients, where patients who have suffered a stroke must activate their muscles in order to rehabilitate. Using a VRE has been accepted as a tool for rehabilitation and also presents the physiotherapist with the possibility to easily upgrade the difficulty exerted to the patient, making personalisation possible during training [9].

### Integrum AB

Integrum AB is a company specialising in osseointegrated prosthetics, currently exploring the possibilities for prosthetic control through bioelectrical signals. A recent master's thesis done at Integrum AB provided the company with a VRE where a visualisation of a hand was made. The aim of the thesis was to create a VRE to assist the training of the pattern recognition algorithm, although no connection was made between the VRE and the pattern recognition [10]. Current training methods consist of the patient following directives given as text on a screen and a small picture which shows the final position

to be achieved. If the patient is successful in performing the correct movement, feedback is given, again in the form of a text, telling the patient which movements have been achieved.

To achieve better feedback and to create a more intuitive interface during training a mirroring of the movements of the healthy hand in unilateral amputees can be retrieved using a sensor glove. These can be used to make visualisations and to give patient proper feedback as to whether or not the proper action is performed.

## 1.2 Aim

The aim of this thesis is to rebuild large parts of the current VRE and to connect this to the biological pattern recognition software through an interface. Training and rehabilitation tests will be implemented and tested.

## 1.3 Restrictions

The following restrictions apply to this thesis work;

- No work being done to the pattern recognition software.
- No changes made to the rendering module of the VRE.
- No tests need to be made on patients.

## 1.4 Problem

These are the problems that will be faced in this thesis work;

- Creating a protocol for data transfer which ensures sufficient information is sent whilst limiting the network strain.
- Creating a modular interface allowing inputs from pattern recognition and other interfaces.
- Create a complete link between an existent pattern recognition system and a modified virtual reality environment to myoelectrically control a hand visualisation.
- Create a rehabilitation tool to use with a trained pattern recognition system.

## 1.5 Scope

At the beginning of the project a shorter study was carried out to see whether similar systems existed. In doing so a general overview of possible functionality, usability and implementation can be done. To ensure that sufficient speeds were obtained between the two applications, an investigation into different methods of communication between the systems was also carried out. Extensive testing of the existent system was also done, this to get familiar with the previous setup and structure of the program. Based on the knowledge of the existent system it was possible to implement a desired interface that met the standards of the system.

After implementations all necessary tests, to ensure sufficient communication speeds, were made ensuring that the interface met the requirement set by the existent system. After making sure sufficient communication speeds are met, a live test was conducted. Test subjects tried out the system, making sure that there was no delay between the two systems.

A rehabilitation tool for amputees was also created, this to provide a good training ground for signal training using the VRE.

# 2 Amputation

Amputation is the medical word for removing any part of the body, most commonly via surgical means. Surgical amputation is most commonly caused by injury or trauma, infections, tumour, diabetes and obstruction of blood flow.

Another form of amputation is congenital amputation, this is when someone is naturally born with a deformation of some part of the body.

Dysvascular amputation is caused by obstruction of blood flow in some part of the body, causing death of local tissue. When diabetic patients get what is called "diabetic foot" this is a form of a dysvascular disease, where one common treatment is amputation of the foot.

The most common types of amputations include; digit (i.e. fingers and toes), transtibial (shin), transfemoral (thigh), transradial (lower-arm) and transhumeral (upper-arm) amputation. In this thesis the cases of transradial and transhumeral amputation are most relevant due to the use of motor-prosthetics, something that is not yet considered in the other cases.

## 2.1 Statistics

In a study, Ziegler-Graham et al. [1] estimate the current prevalence of amputations over the next 40 years. In 2005 there were a total of almost 1.6 million people in the United States alone with some form of limb loss. 700.000 people had become amputated due to some form of trauma, where men accounted for roughly 80 percent of all trauma cases. Out of all the cases of limb loss more than half were caused by dysvascular diseases and almost three quarters of these had an underlying cause of diabetes [1].

During the next 45 years, until year 2050, it is estimated that the amount of persons suffering limb loss is to increase by 130% [1]. This greatly increases the need for further health services and prosthetics in order to ensure a high-quality of life.

## 2.2 Causes

As stated in the previous section there are two major causes for limb loss, trauma and dysvascular diseases. A small portion of people, who need some form of amputation, have some form of cancer as an underlying cause, but these only account for roughly 1% of all the cases. One major underlying cause for dysvascular amputations is diabetes, a disease which has become increasingly common during the last century, being the cause for 60% of all non-traumatic lower-limb amputations [11]. Due to the major increase in diabetes, and with an all increasing obesity in the world, a known cause of diabetes, the number of dysvascular cases is very unlikely to decrease or even stabilise.

# 3 Prosthesis

The earliest documentation of prostheses show that they were not made to replace functionality but to replace the feeling of a missing limb and to visually restore an amputees appearance. The prosthesis that was used for more than aesthetic reasons was found, in the year 2000, in an Egyptian tomb near the city of Thebas. It was an artificial toe that scientists believe was worn sometime between 1069 to 664 B.C. by a 50- to 60-year-old woman who had received it after complications due to diabetes [12].

There have been numerous records of prosthesis used in battle around the turn of the first millennium. These were prosthetic devices to assist soldiers in holding a shield or supporting them in a stirrup [13]. For almost 500 years there was little documented progress in the field of prosthetics, but in 1508 a German mercenary constructed a pair of iron hands which could be positioned manually and then moved through a series of pulleys and springs, a feature still found in many modern day prosthetic devices [13].

Motorised prosthetics were first introduced in the 20th century where electrically powered prosthesis were controlled by myoelectrical signals. The Central Prosthetic Research Institute of USSR developed the first commercial myoelectric arm in 1960 [14]. It was controlled by a single, strong contraction and had only one DoF, opening and closing the hand. Due to the simplicity of the prosthesis and that it is easy to use; it is still one of the most used techniques when using myoelectic prostheses.

In 1969, Fyson et al. [15] were discussing how to improve the existent prosthetics. Even today, more than 40 years later, many of the same requirements are still a hot topic in the development of prosthetics such as power consumption, ease of use and maintenance.

The different types of upper-extremity prostheses can be roughly divided into three different categories; passive, body-powered and externally-powered.

## 3.1 Passive Prosthesis

A passive prosthesis can either be considered functional or non-functional. A cosmetic prosthesis can usually be characterised as a non-functional prosthesis where the aim is to mimic and restore the appearance prior to the amputation, rather than provide the patient with any additional functional abilities. A functional prosthesis is in this case classified as a prosthetic device that provides the amputee with additional capabilities. In most cases a functional prosthesis, non-powered, provides the user with one or no DoF but rather provides a hook that can be used for carrying or holding. If the prosthesis does provide some DoF, they are usually positioned into the wanted pose by manually moving the prosthetic into place.

## 3.2 Body-Powered Prosthesis

A body-powered prosthesis receives the power to move from the movements of the amputee's body. A body-powered prosthesis is common in the case of transhumeral amputation, also known as above-elbow amputation, and is commonly used to restore flexion of the elbow and/or closing of the hand. Using shoulder motion, such as scapular abduction, a cable connected from the harness to the prosthesis will lift the forearm of the prosthetic or close the hand.

## 3.3 Externally-powered Prosthesis

An externally-powered prosthetic, also known as a motorised prosthesis, is a mechanical prosthesis which is powered by some form of external source, usually in the form of a battery and a motor. Advantages of a motorised prosthetic is that many more degrees of freedom are able to be restored since limiting factors such as existence of controllable muscles, as with the body powered prosthesis, are not a problem. In order to add more DoFs the prosthesis needs to be fitted with more motors, something which will cause problems with the weight.

The most common form of control for the motorised prosthesis is through the retrieval of Electromyogram (EMG) signals. A more detailed explanation on how the motorised prosthetic is controlled is discussed under various parts of section 4.

Apart from EMG, an externally-powered prosthesis can be controlled via neural signal; this is then referred to as a neuroprosthesis. A neuroprosthesis can be classified as a device which is used to substitute motor, sensory or cognitive modality in a person who may have lost this functionality to a disease or some form of injury. In this context a neuroprosthetics, or neural prosthesis, is a motorised prosthetic that the amputee patient is able to control via pure bioelectrical signals. The signals used to control the prosthesis can be retrieved in a number of different fashions, discussed under section 4.1, and are all interpreted and classified to control the movement of the prosthesis.

Warwick et al. [16] conducted a study on whether a micro electrode array was successfully implanted into the median nerve of a healthy patient and then used for sensory feedback and control of a prosthetic device. The array consisted of 100, 1.5 mm long, individual needle electrodes arranged in an array measuring 4x4 mm. Tests showed that accurate control of the prosthetic device's grip was possible with simple feedback on whether or not the grip on certain objects was slipping or not. The subject was also able to achieve real-time control of an electrical wheelchair via command signals, decoded in real-time, from the neural input of the median nerve. Reasonable navigation and a 90% accuracy of desired control was achieved within only one hour of practice [16].

In the case of a high-level upper-limb amputation there may be little muscle to retrieve signals from. Kuiken et al. [4] have been working on a method to bypass this problem, Targetted Muscle Reinnervation. When presented with very few myoelectric retrieval sites remaining motor nerves are transferred to nearby muscle tissue so that they can reinnervate this muscle. During normal neural activity, like closing the hand, the bioelectrical signals are transferred into the reinnervated muscle, causing a contraction, and EMG signals can then be picked up to control the prosthesis [4]. Although many advantages are presented with Targetted Muscle Reinnervation for high-level upper-limb amputations there is a very long period of recovery, stretching to more than 7 years [17]. There may also be trouble in retrieving the signals independently, this when reinnervating multiple nerves into the same muscle.

# 4 Flow of Prosthetic Movements

When using a neuroprosthesis there are many different parts, which must work seamlessly together in order to ensure correct and accurate movements. The process of going from a raw bioelectrical signal to moving a prosthetic hand can be split up into 4 stages; acquisition, processing, pattern recognition and prosthetic movement, each described briefly below.

## 4.1 Signal acquisition

Every movement performed by the human body can be oversimplified as muscular response to bioelectrical signals being sent out via the peripheral nervous system from the central nervous system. These electrical impulses, which travel along the nerves, are amplified when reaching the corresponding muscle tissue. In order to retrieve the bioelectrical signals an Electroneurogram (ENG) or an EMG has to be performed. Collecting any valuable data from an ENG can prove difficult due to surrounding noise, this since nerve signals have an amplitude in the order of $10\mu V$, whilst muscular tissue produces signals with an amplitude up to $2000\mu V$ [18]. In order to acquire these signals there are a few different types of electrodes which can be used, the two major types are surface and implantable electrodes.

## 4.2 Signal processing and Feature Selection

In prosthetic movement control systems it is vital for the system to be able to identify the intended movement from the generated bioelectrical signals. The signals which were collected, described in section 4.1, will be fed into a pattern recognition algorithm (PRA) in order to identify which movement was intended, see section 4.3. In order for the PRA to function correctly the input must be somewhat distinguishable from one another, and to ensure that the signals are as unique as possible all common artefacts such as noise and other disturbances must be removed.

Common techniques for removing the noise, which is accompanied with the raw signals retrieved from the electrodes, are to apply a band-pass filter with 20 Hz and 500 Hz as cut off frequencies. The idea of the filter is to remove motion artefacts and noise before amplifying the signal with a high gain. A notch filter is most often also applied to eliminate 50 Hz or 60 Hz, the noise created by the power line [19] [20].

Once filtering of the signal is complete some features of the signal must be retrieved in order for the pattern recognition to distinguish the difference of the different movements. To ensure an effective classification, features are to be extracted from both the time and frequency domain. According to Rivera and DeSouza the most common features to be used are: number of Zero Crossings(ZC), Mean Absolute Value (MAV), Slope Sign Changes (SSC), Waveform Length (WL), coefficients of the Short Time Fourier Transform (STFT), and a few more [20]. The amount of features which are available are extremely broad and the most effective combination is difficult to state since it may differ from case to case.

## 4.3 Pattern Recognition

In this thesis no particular pattern recognition will be covered but the general concepts will be described.

The key to a successful prosthetic movement system is to be able to correctly identify which movement is intended by the user. According to a literary study over prosthetic use and abandonment during the last 25 years, 23% of the electrically powered prosthetic users have rejected their device [21]. Common causes for abandonment of the device is displeasure with the functionality and appeal of the prosthesis. A mechanical prostheses functionality, controlled via SEMG signals, is directly affected by the classification accuracy of the pattern recognition algorithm used to identify the correct movement. When using a PRA for the prosthetic movement there are two important steps; training and classification.

Using the selected features of the recorded bioelectrical signals, a PRA will be trained to associate the values of the features to the desired outputting movement.

### 4.3.1 Offline Classification

In order to confirm that the training has been successful, testing is required. When there is no user inputting live data into the algorithm this is considered to be offline classification, recognition of pre-recorded signals. When recording the signals they are usually divided into training sets and validation sets, where the training sets are used for initial training of the algorithm and then validated using the remaining data. This is done to see the accuracy of the newly trained algorithm.

### 4.3.2 Online Classification

In order to use the prosthesis effectively it must be able to recognise the movements "on-the-fly", this is also known as online classification. It can prove more difficult to classify movements in the online case, this due to change in surroundings and the recorded data. It is also more vital for the online classification to have a low classification error to ensure correct control of the prosthesis and to reduce frustration for the end-time user.

## 4.4 Prosthetic Device

Once the correct movements have been classified these must be translated into prosthetic movement of an external device. Currently there are quite a few different products under development. One of the commercially available products is the i-limb ultra [22]; a motorised hand that can be controlled by various inputs. Another product is the DEKA arm [23] which is a research project in development by the U.S. Army Research Office and not yet a commercial product.

In this stage it is possible to have either a stand-alone device, such as the i-limb ultra, or to use the product of this thesis; a VRE for displaying the interpreted movements of the user.

# 5 Training of a Pattern Recognition System

Previously it was considered that little could be done to improve the classification accuracy of the Pattern Recognition Systems available, this was in consideration to the offline accuracy. Recent research has seen that the accuracy of the pattern recognition system had room for improvement when comparing the offline classification accuracy to the online classification accuracies [24]. Thus, by spending more time in trying to get the optimal pattern recognition system, it is possible to improve the offline and online classification accuracy.

In order to give an answer to the question of which method of training will give the best result, Bouwsema et al. compared three different types of myoelectric signal training. The three types of training were with either a fitted prosthesis, an independent prosthesis on the table or a virtual representation of a prosthesis displayed in a computer screen. The users first trained with the different systems and then were asked to perform some tests where the times and speeds were recorded and compared. Results showed that there was no evident difference between the three methods used, only subtle differences caused by users learning capability [25]. Using the results of this study, a virtual representation will give equally good training results, as using either a tabletop prosthesis or a fitted one, the pure financial advantage of being able to use a virtual prosthetic are substantial.

## 5.1 Using VRE as a Visual Aid during Training

In order for the patient to train the VRE as best they can, it is important to train using accurate signals. To ensure retrieval of accurate signals, a demonstration of the wanted movement is desired. To assist the patient something more than just an image of the final position is wanted. Using the VRE a short demonstration of the wanted movement can be displayed, thus assisting the patient to simply follow the movements displayed rather than having to perform them without any form of assistive guidance.

# 6 Post-Training Utilities

After training of the pattern recognition system, it can be used along with various utilities to improve the usability of the system for the patients.

## 6.1 TAC Test

Error Augmentation (EA) is the process of displaying the error which the patient performs during a movement, but in an exaggerated manner. For instance this might mean that the distance which the patient is off on a trajectory might be shown as double the distance. This is used so that the patient may better realise when they are performing a faulty movement. In a study, to see which type of EA is most effective, it is clear that when comparing those groups which use EA during training, or any form of visual error feedback, perform better, than those without any feedback [26].

Knowing that visual error feedback provides improved training, this is naturally a wanted feature to include in training, for using the prosthetic device. Wei et al. have conducted a study showing that there is little to no difference between using an actual fitted prosthesis and a virtual representation of one [26]. Using the information provided by this study, the use of a VRE can prove to be very cost-efficient in comparison to providing actual prosthetic devices.

In 2009 Simon et al. evaluated a new method for testing the real-time myoelectrical pattern recognition control of a prosthesis. The new test presented the user with a visual representation of the hand controlled by the user, displayed on a computer screen, and a desired position of a hand, displayed in the same screen, where the user uses the pattern recognition control to move the hand into the desired position. Similar previous tests had not taken the unintended movements into consideration, taking these into consideration the new virtual test was created, named Target Achievement Control Test (TAC Test) [27].

It is also shown that better results can be obtained if the patient uses visual feedback during training than if no feedback is given as to how the patient is performing [28].

## 6.2 Rehabilitation Aid

Using a VRE as rehabilitation aid has been proven effective for post-stroke patients [9], but for prosthetic rehabilitation there is little published on how effective this can be.

Motivating a child to perform a repetitive task in order to aid their rehabilitation may prove difficult, if implementing this task into an objective of a game this would most likely motivate the patient to persist with their rehabilitation. Again there is little published about this implementation for prosthetic patients, but there are similar articles published for post-stroke rehabilitation. Due to similar implementations these articles will be reviewed, but results may prove to be different.

A very simple game was developed by Cameirao et al. where the user sits in front of a computer screen and controls two hands and tries to catch a series of spheres which fly towards them. Success of intercepting the sphere accumulates the patients final score. The difficulty of the system is set dynamically depending on the performance of the patient, making each trial customised for the specific patient [9]. Even though

these results are not directly applicable for prosthetic users it is clear that simple virtual reality (VR) games can be implemented to provide an easy and accessible way of rehabilitation.

Using a VR-based rehabilitation game would also give the prosthetist the advantage of being able to monitor the progression of the use of the prosthesis. This since data on speed, accuracy and ability can be stored and displayed over the time of the rehabilitation period to track improvements.

# 7 Method

For this thesis implementations and modifications will be made in order to use a VRE for training and rehabilitation methods explained in prior sections.

The primary problem is to make the two existing software, one written in MATLAB and the other in Visual C++, to work together continuously. Using the existing VRE, mainly supported by inputs from an AcceleGlove, a new VRE was created supporting many different application such as normal movements, TAC Test and more.

## 7.1 Means of Implementation

This section will shortly describe the different environments which have been used in order to create the VRE.

Initial implementation of the VRE was made in Visual C++. Using Visual Studio 2010, tests were carried out of the current software. Visual Studio 2010 is an implementation tool developed by Microsoft that can be used to implement software in a variety of languages such as C#, VB, Visual C++ and many more. This to see the functionality that existed prior to changes. Quite quickly this environment was abandoned due to complications of distribution connected to the compiler used for Visual C++.

To implement pure C++ (not Visual C++), Code::Blocks is used, making distribution of software much easier. This program was then used throughout the entire project to implement the software.

The PRA, to be used as a controlling input for the VRE in this project, is implemented using MATLAB. In order to correctly control the VRE the same environment, MATLAB R2011b, is used to implement the inputting part of the communication interface.

The hand model which is used for this project was created using Blender, an application developed by Blender Foundation. Using Blender to make the adequate changes to the model it was later used to export these models to the correct file-format to be used with the software.

Blender is a free 3D modelling tool which can be used to create 3D models using a user-friendly interface.

## 7.2 Virtual Reality Environment

The existing VRE was configured to support the use of an AcceleGlove to control the movements of the hand.

Creating the visualisation of the hand from scratch was proven to take a lot of work and the result would most likely not be as good as if using an external framework for this. A ready-made 3D rendering engine framework, by the name of Ogre3D, was chosen as a basis to the project due to its simplicity and ease of implementation [10].

In order to successfully implement a communication protocol to control the movements of the VRE many changes needed to be made in both the rendering of the movements and also in the manner of which the movements were handled. Although the previous code layout, to display the movements of the AcceleGlove, was very well structured and somewhat object-oriented, the layout was not implemented to support the new modifications. Due to this there was need for modifications in the structure of the code.

### 7.2.1 Hand Model

The hand model used for the VRE, created using Blender, consists of the surface of the hand, covered by a pre-defined texture, and the bones of the hand. Creating movements is done by moving the bones of the hand. When a new position of the bone is set, the corresponding texture connected to it is also moved. Any other bone which is classified as a child of that bone is also moved, meaning that when changing the position of the palm all of the fingers will move along with it.

In order to create a more natural look for the user the setup of the bones in the hand needed to be changed from the existent extremely simple setup with one bone in each finger, leading to very unrealistic positions. To simulate the natural curving of a finger three bones were added to each DoF, replacing the previous one. The joint connecting the bones within the fingers were chosen at the natural position of the joints, all to simulate a movement as close to the real deal as possible. The actual movement of the limbs will be explained in section 7.2.2.

Along with adding additional bones to the hand, the appearance of the hand needed changing. With the aim of reproducing the look of a natural hand a new model was created, shown in figure 1 as displayed in blender. Along with more natural movements, the new hand model creates a more natural look and feel of the virtual hand.

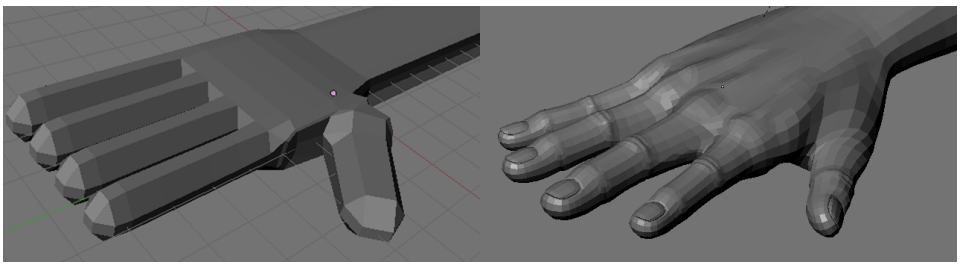

Figure 1: Changes in the hand model used in the VRE.

### 7.2.2 Degrees of Freedom

The initial implementation of the VRE supported only the DoF which would accompany the movements of the AcceleGlove, consisting of the individual movements of the fingers and the movement of the palm. In order to add more DoF substantial editing of the code was needed, this due to the direct connection between the DoF and the sensor inside AcceleGlove. To simplify a new way of handling the different DoF is configured where each DoF is created as a separate object and the bones, which are associated with that DoF, are added to the object along with the axis of movement for the bone. When the VRE then receives the input to move a specific DoF, all of the bones within that DoF are moved along their corresponding axis of movement. When creating a new DoF it is also needed to set the extension and flexion limits of the DoF. This is needed in order to create a realistic movement where no over-extension/flexion is allowed, the process of creating a DoF is shown in Listing 1.

Listing 1: Initialisation of DoF.

```
Create DOF
        set position = 0
    SetLimit
        set extensionLimit = value
        set flexionLimit = value
    SetSkeleton
        set skeleton = HandSkeleton
    ForEach NameOfBone in DOF
        add Bone(NameOfBone) to ListOfBones
        add DirectionOfMovement to BoneDirectionList
```

During movements, the position is of each bone in the DoF is updated to the new position. Before doing so it is controlled to see that the new position does not exceed any limitations set during the initialization. A short demonstration of the updating of a movement is shown in Listing 2.

Listing 2: Moving the DoF.

```
Moving DOF
    appendPosition
        if CurrentPosition + AppendingPositionValue < Limit
            add AppendingPositionValue to CurrentPosition
        else
            set CurrentPosition to Limit

    updatePosition
        ForEach (BoneName, BoneDirection) in ListOfBones
            get BoneInSkeleton from Skeleton(BoneName)
            set SkeletalBonePosition to CurrentPosition * BoneDirection
```

Reviewing the basic movements of the hand, and merely including the basic necessities, each finger has one DoF, the thumb has two and the wrist has three, making the total amount for the entire hand 9 DoF to be implemented. According to some researches there are 22 DoF in a normal hand and in order to be an adequate representation of this a virtual representation should contain about 16 DoF [29]. Even though not used in this implementation all possible DoF will be implemented into the software to ensure no limitations of future use is presented. For a full list of available movement and their corresponding activation signal, see the Appendix.

### 7.2.3 Poses

In order to simplify the control of the VRE it is possible to assign bones from different parts of the hand and group these together. Using this feature multiple DoFs can be combined to create a pose, for

instance, to control the opening and closing of the hand, pointing of a finger or other movements which require the movement of multiple DoF simultaneously. A short description of how a pose is created is given in Listing 3.

Listing 3: Initialising a Pose.

```
Creating Pose
    ForEach (DOFToInclude, DefaultPosition) in Pose
        add DOF to ListOfIncludedDOF
        add DefaultPosition to ListOfDefaultPositions
```

When creating a pose, a default position is also given for each DoF. This position is used when moving the pose and all DoF move towards this position during movement.

In table 1 some examples of poses which have been implemented into the VRE are given.

| Pose | Involved Bones | Movement |
|---|---|---|
| Hand | Thumb, Index, Middle, Ring, Pinky | Hand Closing and Opening |
| Wrist1 | Palm | Wrist Suponation and Pronation |
| Wrist2 | Palm | Wrist Flexion and Extension |
| Point | Thumb, Index, Middle, Ring, Pinky | Pointing Index, Closing rest of Hand |
| Pinch | Index, Thumb | Pinching of Index and Thumb |

Table 1: The different poses which are currently supported by the VRE.

In figure 2 a demonstration of one pose is given, Hand Closed. This movement uses all of the DoF in all fingers and the thumb in order to be performed. Again the figure displays the difference between the remodelling of the model discussed in section 7.2.1.

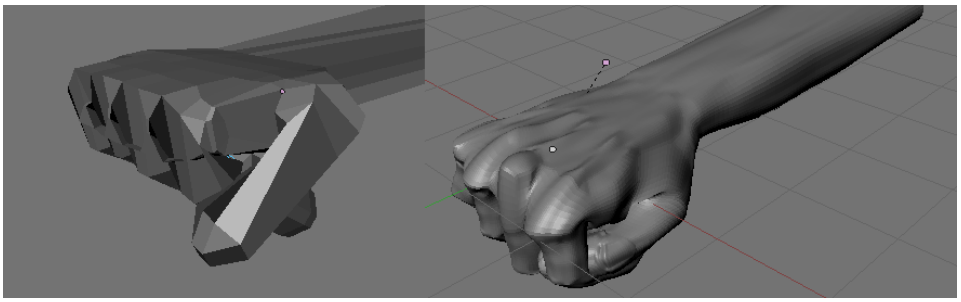

Figure 2: Closed Hand model before and after remodelling, displayed in Blender.

Poses also introduce another scenario that does not only simplify moving multiple DoF simultaneously, but will also reduce the amount of data needed to communicate the desired movement to the VRE. If the user wants to close the hand, without using poses, this will require a continuous flow of 5 signals being sent, in order to close each finger individually. Whilst when using poses the movement can be achieved with only one signal sent to the VRE, thus reducing the stress on the communication flow. The movement of a pose is explained briefly in Listing 4.

Listing 4: Moving the Pose.

```
Moving Pose
   ForEach DOF within ListOfIncludedDOF
      get CurrentPosition
      if DefaultPosition >= CurrentPosition
         if CurrentPosition + Value > DefaultPosition
            append DefaultPosition - CurrentPosition
         else
            append Value
      else
         if CurrentPosition - Value < DefaultPosition
            append DefaultPosition - CurrentPosition
         else
            append -1 * Value
```

## 7.3 Communication

Communication between two different programs, running on separate processes, uses an implementation of an interprocess communication (IPC) medium to transfer information. According to the Microsoft Windows documentation there are a number of implementations of IPC on the windows system. Out of the given alternatives there are a few which are interesting in this case; COM ports, Pipes and Windows Sockets. This due to their ability to be integrated into both MATLAB and C++. In the end Windows Sockets was chosen, due to its ease of implementation and its diversity in connections from different systems.

### 7.3.1 Using Windows Socket Interface

When talking about sockets there are two protocols which are discussed; Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). When it comes to the implementation there is little difference between the two transport layers on top of Internet Protocol (IP) and both are supported by the languages used for this software implementation.

A short list of features and benefits are summarised in table 2.

| Features | TCP | UDP |
|---|---|---|
| Connection | Three-Way Handshake | No-Connection |
| Speed | Slower | Fast |
| Reliability | Guaranteed Receival | No guarantee |
| Ordering | Packets are always in order | No ordering guarantee |
| Header Size | 20 bytes | 8 bytes |

Table 2: TCP and UDP feature list.

The difference of TCP and UDP is quite significant, but in order to know which is necessary for the implementation each part is assessed for the implications in this implementation.

**Connection** The TCP uses a so called three-way handshake to initialise its connection, this means extra time to get established and since UDP has no such features it is definitely faster in this step. The rendering of the environment needs also to be done before any transmission can be started, something which will take far longer than any connection, meaning this step is not crucial for the decision of which transport layer to choose.

**Reliability** Due to UDP not having any form of confirmation that the data has been received properly, it has very low reliability when transferring data.

**Ordering** In the header of the TCP there are 4 bytes which are dedicated to a *Sequence number*, this ensures that the packets, even though one is delayed and arrives to the client at a later stage, are ensured to be processed in the correct order. Since the UDP header is much lighter and contains less information, this is not the case for the UDP where there is no way of telling in what order the packets arrive in.

**Header Size** Considering only the header of the packet, since the other data is equivalent independent of the type of protocol used, the UDP header is 8 bytes whilst the TCP header is 20 bytes.

**Speed** One of the larger differences between the two protocol is the speed at which they transfer data. Since the header size for TCP of 20 bytes, this will be more than twice as much as UDP that only has a header size of 8 bytes, something which will naturally affect the speed of the transfer.

To ensure that enough data is received in the VRE to have a natural flow of the movements, the amount of packets per second must exceed the desired frames per second of the rendering. This requirement ensures that the speed of the rendering is in no way hindered by the receiving of packets. A reasonable update speed for the VRE is to be greater than 25-30 frames per second, a small test was made to ensure that the transport protocol did in no way hinder the required frames-per-second limit. The test matched the implementation's environment exactly, having a client in C++ connect to a server hosted in a MATLAB application. Since we know that UDP is a faster protocol the TCP was tested first to see whether this met the required transfer speed.

| Number of Bytes | Transfer time of 1000 packets [ms] |
|---|---|
| 2 | 31 ms |
| 4 | 94 ms |
| 8 | 109 ms |
| 16 | 109 ms |
| 32 | 141 ms |
| 64 | 156 ms |
| 128 | 250 ms |
| 256 | 2621 ms |

Table 3: Display of time taken to transfer 1000 data packets in different sizes between MATLAB and C++ over TCP.

Taking into account that the speeds may differ slightly depending on the hardware of the machine which they are performed on, and the amount of time it took for the transfer, it is safe to say that any size of the packet which is below 128 bytes results in sufficiently high transfer speeds.

**Conclusion** When building an interface which has a heavy data load, and a lot of information needs to be passed in a short amount of time, UDP is a very good choice. On the other hand, if speed is not of the essence and a slightly slower but a lot more secure data connection is desired, TCP is the best choice. Since the speed test displayed in table 3 shows that the speed of the TCP transfer will not hinder the rendering, and it is the most secure connection, this will be the choice for this implementation.

### 7.3.2 Socket Implementation

Due to working with two different environments, two different implementations needed to be done. One integrated into the existent biological pattern recognition software, BioPatRec, based in MATLAB, and the other implemented into the VRE, developed in Visual C++.

**Matlab** The implementation was done using the integrated support for TCP-IP sockets in the Instrument Control Toolbox [30]. Using an object-oriented approach, a *Server* class was written, containing three functions; *Connect*, *Read* and *Write*.

Upon initialisation the *Connect* function will be called and create the connection between the server and client. This function will take a port as a parameter and will connect to *127.0.0.1* or *localhost* on the given port, thereby assuming that the server and client are hosted on the same computer. Once the connection is established the *Write* function can be used to send any data to the client. Once data is sent the *Read* function should be used, to see the returning value from the client. The returning value from *Read* will also tell the server how the data was interpreted at the client side and if it has arrived correctly. It may be more intuitive to call the *Read* function after sending the data in the *Write* function, but since it is not needed for all uses of a TCP-connection it is not implemented.

**C++** Using the information provided by Hall [31], in his book on network programming using Internet sockets, the implementations in C++ were done using *winsock*, also known as *Windows Socket*.

As was done in MATLAB, an object-oriented approach was taken, creating a *TCPSocket* class which connects, sends and receives all data from the interface.

During initialisation a winsock object is created and stored in the *TCPSocket* class stating the type of socket connection, TCP in this case. Once initialised, the socket can then connect to a specified address and port. In order to make sure that there is a server waiting for the connecting socket a minor loop is constructed to try to to connect to the server at the specified address, at the specified port every second until a connection is established. This is a minor precaution to make sure that the program simply does not exit due to not finding a server to connect to. After the connection is established the socket is configured to act as a non-blocking socket, this to ensure it does not block all computations whilst waiting for data to be received.

Receiving data from the socket interface is quite simple after initialisation is completed, a *Receive* function is comprised to handle this. The function calls the appropriate receive function for the sockets, with a memory buffer as a parameter. This buffer will then be filled with the appropriate data, and an integer will be returned signifying how many bytes of data have been written to the buffer.

Since a non-blocking socket is created this means that calling the receive function will not lock the program and wait until data is sent but will return, with data if there exists any. If no data is present

the function will return $-1$ instead of number of bytes written to the buffer, an error of type 10035 will also be generated, which can be retrieved using the *WSAGetLastError* function. Thus, when the receiving function returns a control to see whether or not the last error was equal to 10035 will indicate that there was no data sent. The connection is still wanted to be open, only performing no movements in the VRE, and this is signified by a returning value of 2. However, if the receiving function returns 0 this indicates that the connection has been closed by the server, once this happens the *Receive* function will return 0, signifying that the application can be closed.

## 7.4 Communication Protocol

In order for the two programs to correctly communicate with one another a protocol had to be established. Before establishing the layout of the protocol it was important to realise some requirements and restrictions. The protocol had to provide enough information for the VRE to perform the corresponding movements.

When wanting to move the virtual hand there are only a few things which the VRE needs to know; which DoF, which direction and how far. Considering that there will most likely not be more than 255 movements, directions or distances, each one of these can be represented by one byte each. In the protocol there must also be some way of distinguishing whether to move the normal hand or the TAC hand, to set this into an appropriate position, to configure the environment (explained further in section 7.4.2) and additional features, such as resetting the hand position.

Considering the required information the decided protocol is setup using 4 bytes of data being sent from the server to the client, and then 1 byte of data being sent back to the server after completing the task at the client. In the data sent to the client it is important to state, first and foremost, what procedure wants to be done. Therefore the first byte represents the action which is to take place, shown in table 4.

| Character Value of Byte [Hex-Code] | Significance |
|---|---|
| 1 [0x31] | Choosing the normal hand for movements |
| 2 [0x32] | Choosing the TAC hand for movements |
| c [0x63] | Configure environment (see section 7.4.2) |
| r [0x72] | Reset the position of the hand (see section 7.4.3) |

Table 4: Protocol for Socket Communication, First Byte

Based upon the first byte, the second, third and fourth bytes will have very different significance. If the first byte signifies the movement of a hand entity this is described in section 7.4.1, configuration of the environment is described in section 7.4.2, and if it signifies to reset a hand position, this is described in section 7.4.3.

In the current implementation there is not much information which is needed to be passed from the client to the server after the movement has been completed. In order for the server to acknowledge that the movement has been successful one simply byte will suffice. This acknowledgement, also known as a ACK, can also inform the server of whether or not the TAC test has been successfully completed, if not completed, this is then referred to as a negative acknowledgement, also known as a NACK. The values which are currently implemented are shown in table 5.

| Character Value of Byte [Hex] | Significance |
|---|---|
| 0 [0x30] | A fault has occurred and movement has not been performed. |
| 1 [0x31] | Movement has been performed. |
| t [0x74] | Movement has been performed and TAC Test is completed. |

Table 5: ACK or NACK from client to server after computations at client.

A full list of all byte combinations is given in the Appendix.

### 7.4.1 Movements

When the VRE realises that the first byte of the incoming signal corresponds to a movement, whether it corresponds to the user-controlled hand or the TAC hand, it uses the remaining bytes to define which DoF to move, in what direction and how far to move it, see table 6. The distance which the hand is moved is set to a value of 1/100 of the specified value, this due to the lack of possibilities of sending fractions as a byte.

| Which byte | Significance during Movements |
|---|---|
| Second Byte | Indicates which DoF (section 7.2.2) to move. |
| Third Byte | Indicates in which direction the chosen DoF is to be moved in. |
| Fourth Byte | Specifies the distance to move the DoF |

Table 6: Protocol for Socket Communication, during Movements

Both the TAC hand and the user-controlled hand are setup in identical manners, meaning there is no difference in how their movements are handled. Therefore one function can manage the movement of any hand entity supplied as a parameter to the function.

The distance which the DoF is moved is not only affected by the variable that is sent with the movement signal. It is also affected by a variable which is set globally within the environment, and can be changed using the configuration variables, section 7.4.2. The distance which the DoF is moved then corresponds to the distance given, multiplied by the set global variable.

### 7.4.2 Configuration Variables

To make sure no alterations are needed of the end-product after publication the ability to configure the VRE is implemented. The end user is assumed to have little to no experience with programming or texture editing, this means that all values which are set inside the software need to be changeable during operation. Such variables include the set distance of movement of hands, meaning how far the distance of for instance 1 is interpreted, changing the viewing angle of the camera, general settings concerning the TAC Test, and more.

| Configuration Character | Significance |
|---|---|
| 1 | Interpretation distance (value / 100) |
| 2 | Polls between performing TAC Test or not (displaying TAC hand) |
| 3 | Allowance of TAC Test (see section 7.5) |
| 4 | Changing the angle and position of the Camera |

Table 7: Configuration Variables

### 7.4.3 Resetting hand position

Once completing a movement, or a TAC Test, it might be desirable to reset the position of the hand. The most simple method for this, with the implementation for this thesis, is done by setting the position value of all the DoF to zero and then updating the rendered image. Knowing that the incoming signals translate to resetting the hand position, the VRE checks the second incoming byte to see whether the user-controlled hand is to be reset or if it is the TAC hand.

## 7.5 TAC Test

A TAC Test, procedure covered in 6.1, consists of many different parts within the system in order to function properly and give the patient and/or prosthetist the required information. The first thing is that it must be easy to initialise and it should not require any changes to the software in order for this to run. The user should quickly be able to start the test, and just as quickly go back to simply controlling the hand-visualisation. In order to grasp what is required a simple flow of procedures was conducted to see which parts needed to be supported for the TAC Test to function correctly. Figure 3 shows the full procedure of performing a TAC Test, where the times and efficacy are then recorded and compared to previous tests.

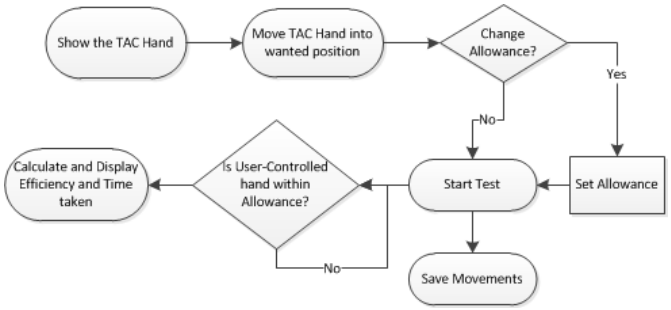


Figure 3: TAC Test Procedure

### 7.5.1 Virtual Reality Environment

When initialising the system it renders and loads all objects, including the TAC hand, and sets its visible parameter to false. Due to this it is quick to start the TAC Test, since no modules have to be loaded. When running a TAC Test, after completing each movement, a comparison between the user-controlled hand and the TAC hand is made, if these are within the set allowance the returning ACK signal will be *t*, indicating that the TAC Test has been performed successfully.

Comparison of the two hand entities are done in terms of their added DoF. The current position of each DoF on the user-controlled hand is checked and compared to the corresponding DoF on the TAC hand. Only if all the DoF are within the set allowance will an accomplished TAC Test be achieved and signalled.

### 7.5.2 MATLAB

In this setup of the TAC Test, the VRE is solely used for moving the objects around the space and checking whether or not they are in the same position. This means that all other data handling is taken

care of by the interfacing software, in this case the BioPatRec. In order to meet the requirements of the TAC Test, the software must track the amount of time it takes for the user, from starting the test, until they have reached the final position. This is quite easily achieved since the receiving byte, after the movements have been made, indicate whether or not the TAC test was successful. Thus, a simple timer is constructed and started when the user initialises the test. This runs until the system receives an indication that the TAC test was completed, and then stops the timer, recording and displaying the final time.

In order to track the efficacy of the test, this requires a little more computation. The system must keep track of the current position of the hand, and use this value to calculate whether or not the user is moving towards the desired position. When the system first runs a position of zero is set to all the DoF for both the hand entities. Knowing this, all changes in the position must be stored once a movement is operated, allowing the system to keep track of the current position at all times. The efficacy of the TAC Test is computed in the percentage of movements which move the user-controlled hand towards the position of the TAC hand. Thus all movements which move in any direction other than that of the TAC hand can be considered inefficient.

# 8 Results

In this section all results from the performed tests will be presented, it will cover the different uses for the VRE and present the results. Tests with patients were planned at the beginning of the project but were dismissed due to a limiting time-frame.

## 8.1 Controlling the VRE

Controlling the VRE was done in two manners; myoelectrically and manually.

### 8.1.1 Myoelectrical Control

In order to control the VRE, using myoelectric signals, all procedures explained in section 4 must be followed, except for the last stage where the final step is substituted for the VRE.

Two different tests were performed, individual finger control and basic hand movements.

First an array of electrodes were setup on the forearm to recognise the muscle activity during movements of the hand. The electrodes must be placed at specific positions, on top of active muscles, and covering all the vital muscles which control hand movements. The placement of the electrodes in this test can be seen in figure 4.

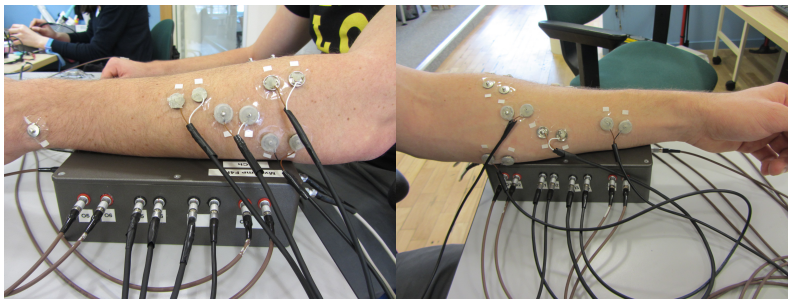

Figure 4: Electrode placement during test of VRE control.

**Assisting Training** In order to assist in training, the VRE was used to display the movements as the patient is asked to perform them. Due to extremely different circumstances in each training case it is difficult to assess whether or not the system really improved the training, but from gathered test data it is clear that it gave a more clear view of which movement was to be performed during the training.

For both the tests a series of different pattern recognition algorithms were setup in the biological pattern recognition system to be used as a controlling environment. These were compared against one another to see which gives the best offline training percentage. For the finger control test an Linear Discriminant Analysis (LDA) was trained using Zero-Crossing (ZC) and Waveform Length (WL) features, giving an offline training percentage of 93.2%. For the control of hand movements both a Linear Discriminant Analysis (LDA) and an Artificial Neural Network (ANN) were tested using the same features as in previous test, but this time with the offline training percentage of 95%, using the ANN.

Running the first test, individual finger control, it was difficult to accurately control the fingers individually and many movements were misinterpreted. This was most likely caused by the PRA, the placement

of the electrode or simply inexperience of usage. The interface between MATLAB and the VRE worked as expected and moved all the interpreted movements, seemingly, instantly. A screenshot of the system during the test is displayed in figure 5.

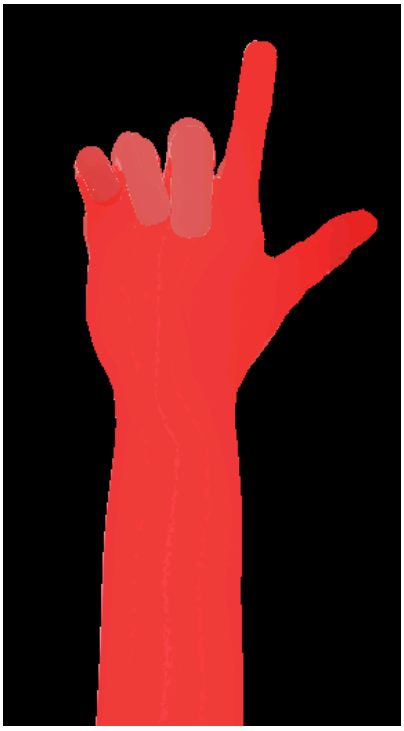

Figure 5: Screenshot of the VRE performing individual finger control using myoelectric signals.

Testing the control of hand movements proved to be much simpler. As stated previously there was an improvement, although minor, during offline training between the two tests. Since the user had no previous experience in using the PRA, this might have caused some minor difficulties in achieving the correct movement, but apart from some minor faults in the movements most of the interpreted signals corresponded to the correct movement. Again the interface between MATLAB and the VRE worked without fault and produced the correct movements in relation to the interpretation. A photo taken during the testing session can be seen in figure 6.

Figure 6: Photo taken during testing session of controlling hand movements using myoelectrical signals.

Even though some movements were not the correct ones this is caused by external factors rather than the actual interface. Thus after testing the system with the PRA it is clear that it fulfils its function as a substitute for a motorised prosthetic hand and can therefore be used during training and testing of a PRA.

### 8.1.2 Control Without Pattern Recognition System

Moving the hand without having a trained pattern recognition system as a control is quite an important aspect of this project, this since placement of the TAC hand must be made manually. Using the same socket interface as the system, and the same protocol, signals are sent to move the hand or TAC hand.

In order to test the VRE, without the need of setting up a pattern recognition algorithm, a small graphical user interface (GUI) application was built using MATLAB. This interface was solely setup to test the functionality of the VRE, but turned out to be far more useful, providing a lot of information and feedback. The interface is shown in figure 7.

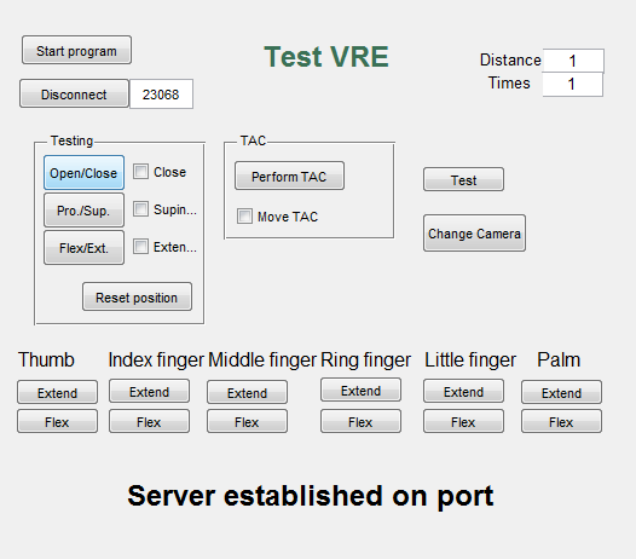


Figure 7: GUI Application in MATLAB for testing the VRE.

Using this interface, all DoF can be moved individually to mimic any hand movement. It is also possible to test additional functionality, such as TAC Test, configuration variables, etc. This means that it can also be used to test the system, making sure the socket interface is functioning properly.

## 8.2 Implementation of TAC Test

In order to test the TAC Test the same setup of the PRA was conducted as in section 8.1.1.

To conduct a TAC Test from the pattern recognition system a small addition to the existing system was added. Using this small addition showing the TAC hand, moving the TAC hand and calculating the amount of time it takes to reach the TAC hands position is possible. This small addition is displayed in figure 9.

To start the test the TAC hand is first displayed and then moved into the wanted position. This is done by checking both the checkboxes displayed in figure 9, *Showing* and *Move*. The movements are then conducted by sending manual movement signals across the interface. Once a desired position is achieved the test can be started, this is done by clicking the *Start* button, upon which a timer is started to calculate the time for completion.

The entire TAC process is displayed in figure 8; when the initial position of the TAC hand has been set, figure 8a, moving the user-controlled hand towards the TAC hand, figure 8b, and upon completion, when the user-controlled hand has reached the same position as the TAC hand, figure 8c.

(a) Moving TAC hand into wanted position.

(b) Moving the user-controlled hand towards wanted position.

(c) Completed TAC Test.

Figure 8: The procedure of a TAC Test.

Once the test is completed the timer is stopped and the time taken to complete the test is displayed, seen in figure 9.

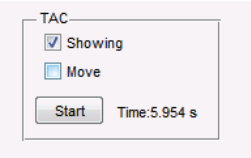


Figure 9: TAC Test functionality in the Pattern Recognition System.

# 9 Discussion of Results

Using the socket connection as the communication interface between the VRE and the Pattern Recognition System proved to be sufficiently fast and secure to transfer the information needed between the two systems. In order for the system to work as intended it is important that control from myoelectrical signals works well and that no delay is present in the communication.

As is shown in the results it is proven that the system works without any clear delay in the communication between the two systems and that control is possible through the use of myoelectrical signals. Even though our test subjects were non-amputees it was still evident that the system work as intended and control was achieved. Amputee patients may not be able to produce as strong muscle signals as a non-amputee patient and therefore may have more difficulty in controlling the hand.

During the results it is shown that the system works well using myoelectrical signals as a controlling input.

During control of the system it seemed that the movement of the virtual hand was quite slow and unresponsive. To overcome this the distance moved for each identified movement was increased, this meant increased movement speeds and quicker response times, overcoming the initial issue.

The main deviation from the intended result was with the TAC test where the implementation only included a test consisting of a single movement to one position. This may seem as an adequate challenge at the beginning, but once the patient gains more experience in controlling the hand a harder challenge may be needed. To implement multiple, consecutive targets within the test would prove more difficult, ensuring that the patient can quickly switch to different positions without the need to pause. This may also prove efficient to train for real-life scenarios where the patient may need to perform multiple consecutive movements to different positions.

# 10 Future Work and Improvements

The VRE in this setup is used to improve training and rehabilitation for patients which are to use prosthetic devices. Many more implementations can be made, to support various different setups and systems. To improve the acceptance and realisticity of the system Augmented Reality could be implemented. This implies that the user marks a spot on the amputated arm and stands in front of a camera which records his actions and displays these in realtime on a computer screen. At the marked position the visualisation of the hand is drawn, making it seem as though the virtual hand is connected to the patient.

In order to improve the rate of improvement for prosthetic control it is important for the user to be doing rehabilitation persistently. It might be difficult to be motivated in doing rehabilitation, but if you are presented with the advancements which you make on the spot, this might cause users to persist with their rehabilitation. Combining the elements of a game with the Virtual Reality setup will provide the patient with means to control a component inside the game, creating high scores or other means of progress tracking. This could cause the patient to continue with rehabilitation, since it gives a sense of improvement. It may also prove important when motivating children to rehabilitate amputation injuries, since they may not be able to see the significance of continuing on with their rehabilitation.

Right now the VRE is only taking whatever input it receives from the PRA and displaying it. In order to demonstrate the movement more intuitively it may be good to implement some form of smoothing algorithm, this to ensure that the movements of the user is cleaned up and all movements which do not match to the current pattern of movements are discarded. This will ensure that movements which are most likely to be unintended are discarded, creating a smoother movement.

# References


[1] Kathryn Ziegler-Graham, Ellen J. MacKenzie, Patti L. Ephraim, Thomas G. Travison, and Ron Brookmeyer. Estimating the prevalence of limb loss in the united states: 2005 to 2050. *Archives of Physical Medicine and Rehabilitation*, 89(3):422 – 429, 2008.

[2] Englehart K. Scheme E. Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use. *Journal of Rehabilitation Research & Development*, 48(6):643–660, 2011.

[3] C. Ishii, A. Harada, T. Nakakuki, and H. Hashimoto. Control of myoelectric prosthetic hand based on surface emg. In *Mechatronics and Automation (ICMA), 2011 International Conference on*, pages 761 –766, aug. 2011.

[4] T.A. Kuiken, L.A. Miller, K.A. Stubblefield, R.D. Lipschutz, and B.A. Lock. Improved myoelectric prosthesis control using targeted reinnervation surgery: A case series. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 16(1):46 –50, feb. 2008.

[5] M.J. Ortiz-Catalan, R. Brånemark, B. Håkansson, and J. Delbeke. On the viability of implantable electrodes for prosthetic control based on pattern recognition: Review and discussion.

[6] G.S. Dhillon and K.W. Horch. Direct neural sensory feedback and control of a prosthetic arm. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 13(4):468 –472, dec. 2005.

[7] Michael M. Merzenich, Randall J. Nelson, Michael P. Stryker, Max S. Cynader, Axel Schoppmann, and John M. Zook. Somatosensory cortical map changes following digit amputation in adult monkeys. *The Journal of Comparative Neurology*, 224(4):591–605, 1984. URL `http://dx.doi.org/10.1002/cne.902240408`.

[8] Anat Mirelman, Paolo Bonato, and Judith E. Deutsch. Effects of training with a robot-virtual reality system compared with a robot alone on the gait of individuals after stroke. *Stroke*, 40(1): 169–174, 2009. URL `http://stroke.ahajournals.org/content/40/1/169.abstract`.

[9] Monica Cameirao, Sergi Badia, Esther Oller, and Paul Verschure. Neurorehabilitation using the virtual reality based rehabilitation gaming system: methodology, design, psychometrics, usability and validation. *Journal of NeuroEngineering and Rehabilitation*, 7(1):48, 2010. URL `http://www.jneuroengrehab.com/content/7/1/48`.

[10] Joakim Arver. A virtual hand for prosthetic training. Master's thesis, Chalmers University of Technology, 2011.

[11] National diabetes fact sheet: national estimates and general information on diabetes and prediabetes in the united states,2011. *Atlanta, GA: U.S. Department of Health and Human Services, Centers for Disease Control and Prevention*, 2011.

[12] Charles Choi Q. World's first prosthetic: Egyptian mummy's fake toe. July 2007. URL `http://www.livescience.com/4555-world-prosthetic-egyptian-mummy-fake-toe.html`.

[13] Kim M. Norton. A brief history of prosthetics. *inMotion*, 17, November 2007.

[14] David E. Sherman. A russian bioelectric-controlled prosthesis. *Canad. Med. Ass. J.*, 91:1268–1270, December 1964.

[15] J. Fyson, P.H. Jilbert, and J.R. Truscott. Design considerations in a myoelectric hand prosthesis. *Electrical Engineers, Proceedings of the Institution of*, 116(2):281 –290, february 1969.

[16] Kevin Warwick, Mark Gasson, Benjamin Hutt, Iain Goodhew, Peter Kyberd, Brian Andrews, Peter Teddy, and Amjad Shad. The application of implant technology for cybernetic systems. *Arch Neurol*, 60(10):1369–1373, 2003. URL `http://archneur.ama-assn.org/cgi/content/abstract/60/10/1369`.

[17] Stubblefield Kathy A, Miller Laura A, Lipschultz Robert D, and Todd A. Kuiken. Occupational therapy protocol for amputees with targeted muscle reinnervation. *Journal of Rehabilitation Research & Development*, 46(4):481–488, 2009.

[18] John G. Webster. *Medical Instrumentation: Application and Design*. John Wiley & Sons, Inc., , third edition edition, 1998.

[19] Mahdi Khezri and Mehran Jahed. Real-time intelligent pattern recognition algorithm for surface emg signals. *Biomed Eng Online.*, 6(45), 2007.

[20] L.A. Rivera and G.N. DeSouza. Recognizing hand movements from a single semg sensor using guided under-determined source signal separation. In *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pages 1 –6, 29 2011-july 1 2011.

[21] Biddis EA and Chau TT. Upper limb prosthesis use and abandonment: a survey of the last 25 years. *Prosthet Orhtot Int.*, 31(3):236–257, September 2007.

[22] Touch Bionics Inc. and Touch EMAS Ltd. i-limb ultra, February 2012. URL `http://www.touchbionics.com/products/active-prostheses/i-limb-ultra/`.

[23] DEKA Research and Development Corporation. The deka arm, February 2012. URL `http://www.dekaresearch.com/deka_arm.shtml`.

[24] E. Scheme, A. Fougner, O and. Stavdahl, A.D.C. Chan, and K. Englehart. Examining the adverse effects of limb position on pattern recognition based myoelectric control. In *2010 Annual International Conference of the IEEE, Engineering in Medicine and Biology Society (EMBC)*, pages 6337 –6340, 31 2010-sept. 4 2010.

[25] Hanneke Bouwsema, Corry K. van der Sluis, and Raoul M. Bongers. Learning to control opening and closing a myoelectric hand. *Archives of Physical Medicine and Rehabilitation*, 91(9):1442 – 1446, 2010.

[26] Yejun Wei, P. Bajaj, R. Scheidt, and J. Patton. Visual error augmentation for enhancing motor learning and rehabilitative relearning. In *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005*, pages 505 – 510, june-1 july 2005.

[27] A.M. Simon, L.J. Hargrove, B.A. Lock, and T.A. Kuiken. Target achievement control test: Evaluating real-time myoelectric pattern recognition control of multifunctional upper-limb prostheses. *J Rehabil Res Dev.*, 48(6):619–628, 2011.

[28] Ian Sharp, Felix Huang, and James Patton. Visual error augmentation enhances learning in three dimensions. *Journal of NeuroEngineering and Rehabilitation*, 8(1):52, 2011.

[29] H. Hashimoto, A. Sasaki, S. Yokota, Y. Ohyama, and C. Ishii. A study on degree of freedom in hand modeling. In *2011 Proceedings of SICE Annual Conference (SICE)*, pages 2492 –2493, sept. 2011.

[30] The MathWorks Inc. Instrument control toolbox, February 2012. URL `http://www.mathworks.com/products/instrument/`.

[31] Brian Hall. Beej's guide to network programming, February 2012. URL `http://beej.us/guide/bgnet/`.

# Appendices

# A Commands for Implemented Protocol

## A.1 Commands for movements

This is a list of all possible commands to control the VRE. All values that can be substituted for an arbitrary value, such as the distance to move a given degree of freedom, are given by a $*$.

| Byte Combination | Responding movement in the VRE |
|---|---|
| '1 1 0 *' | Flexion of the pinky of the user-controlled hand, moving * spaces. |
| '1 1 1 *' | Extension of the pinky of the user-controlled hand, moving * spaces. |
| '1 2 0 *' | Flexion of the ring finger of the user-controlled hand, moving * spaces. |
| '1 2 1 *' | Extension of the ring finger of the user-controlled hand, moving * spaces. |
| '1 3 0 *' | Flexion of the middle finger of the user-controlled hand, moving * spaces. |
| '1 3 1 *' | Extension of the middle finger of the user-controlled hand, moving * spaces. |
| '1 4 0 *' | Flexion of the index finger of the user-controlled hand, moving * spaces. |
| '1 4 1 *' | Extension of the index finger of the user-controlled hand, moving * spaces. |
| '1 5 0 *' | Flexion of the thumb of the user-controlled hand, moving * spaces. |
| '1 5 1 *' | Extension of the thumb of the user-controlled hand, moving * spaces. |
| '2 1 0 *' | Flexion of the pinky of the TAC hand, moving * spaces. |
| '2 1 1 *' | Extension of the pinky of the TAC hand, moving * spaces. |
| '2 2 0 *' | Flexion of the ring finger of the TAC hand, moving * spaces. |
| '2 2 1 *' | Extension of the ring finger of the TAC hand, moving * spaces. |
| '2 3 0 *' | Flexion of the middle finger of the TAC hand, moving * spaces. |
| '2 3 1 *' | Extension of the middle finger of the TAC hand, moving * spaces. |
| '2 4 0 *' | Flexion of the index finger of the TAC hand, moving * spaces. |
| '2 4 1 *' | Extension of the index finger of the TAC hand, moving * spaces. |
| '2 5 0 *' | Flexion of the thumb of the TAC hand, moving * spaces. |
| '2 5 1 *' | Extension of the thumb of the TAC hand, moving * spaces. |
| '1 7 0 *' | Flexion of the wrist of the user-controlled hand, moving * spaces. |
| '1 7 1 *' | Extension of the wrist of the user-controlled hand, moving * spaces. |
| '2 7 0 *' | Flexion of the wrist of the TAC hand, moving * spaces. |
| '2 7 1 *' | Extension of the wrist of the TAC hand, moving * spaces. |
| '1 8 0 *' | Pronation of the user-controlled hand, moving * spaces. |
| '1 8 1 *' | Suponation of the user-controlled hand, moving * spaces. |
| '2 8 0 *' | Pronation of the TAC hand, moving * spaces. |
| '2 8 1 *' | Suponation of the TAC hand, moving * spaces. |
| '1 9 0 *' | Closing the user-controlled hand with * spaces. |
| '1 10 0 *' | Opening the user-controlled hand with * spaces. |
| '2 9 0 *' | Closing the TAC hand with * spaces. |
| '2 10 0 *' | Opening the TAC hand with * spaces. |

## A.2 Commands for configuration

The values used for the configuration of the environment are specified as $*1$ and $*2$.

| Byte Combination | Configuration done in the VRE |
|---|---|
| 'c 1 *1 *2' | Setting the relative movement to $\frac{*1}{100}$ |
| 'c 2 *1 *2' | Toggling whether or not to perform the TAC test. |
| 'c 3 *1 *2' | Setting the distance of allowance in the TAC test to $\frac{*1}{100}$ |
| 'c 4 *1 *2' | Specifying to use the $(*1)$th camera |

## A.3 Commands for resetting the hand position

| Byte Combination | Response by the VRE |
|---|---|
| 'r 1 - -' | Resetting the position of the user-controlled hand. |
| 'r t - -' | Resetting the position of the TAC hand. |