CHALMERS

A virtual hand for prosthetic training

Development of a environment for exercising and training patients with hand prosthesis for Integrum AB.

Joakim Arver

Department of Signals & Systems
Chalmers University of Technology
Gothenburg, Sweden 2011
Master's Thesis EX059/2011

Abstract

Integrum AB has developed a control system for prosthetic devices based on pattern recognition algorithms (PRAs). Currently the training of the PRAs is done by demanding the patient to perform given movements without any feedback to assure a trustable correlation between the demanded movement and the recorded bioelectric signals. An alternative to this procedure is to use an instrumented glove in the healthy hand of a unilateral amputee in order to facilitate the patient to produce the intended movements by commanding both limbs in parallel.

A literature study of similar projects is performed. A comparison of the different methods of developing a Virtual Reality Environment (VRE) is done and the method chosen is to develop a new system using C++ and the Ogre3D- and Openframeworks frameworks to show the pose of a sensor glove of the brand AcceleGlove.

The model of a virtual hand is created using Blender and an iterative process improves the model to a more realistic looking model. The hand is implemented in the VRE and the result is a VRE capable of reading the accelerometers of the glove and representing the sampled values as angles of the hand and fingers. The signals from the glove is filtered by a Kalman- or running average filter to suppress the noise from the sensors.

Acronyms

- ANN - Artificial neural network
- AR - Augmented Reality
- DOF - Degrees of freedom
- EMG - Electromyography, electrical signals made by skeletal muscles.
- OHMG - Osseointegrated Human-Machine Gateway
- PRA - Pattern recognition algorithm
- VRE - Virtual Reality Environment

Acknowledgements

First I want to thank Integrum AB for letting me write my thesis for them and letting me have an own place at their office. I also want to thank my supervisor Max Ortiz for all help and the possibility to always ask questions and for all support throughout the project. I thank my examiner Tomas McKelvey for all help I have received.

Joakim Arver, Gothenburg, June 3, 2011

Contents

1 Introduction

This is the report for the master thesis A virtual hand for prosthetic training performed at Integrum AB in Gothenburg.

1.1 Background

Integrum AB has developed a control system for prosthetic devices based on pattern recognition algorithms (PRAs). Currently the training of the PRAs is done by demanding the patient to perform given movements without any feedback to assure a trustable correlation between the demanded movement and the recorded bioelectric signals. An alternative to this procedure is to use an instrumented glove in the healthy hand of a unilateral amputee in order to facilitate the patient to produce the intended movements by commanding both limbs in parallel. These movements then could be followed by a virtual hand as an example of how a virtual environment could be used for a more intuitive training for prosthetic components. Similar work has been done by Sebelius et al. [17] where a data glove on one hand is used to sample poses at the same time as signals of various Electromyographs (EMGs) from the amputated limb are sampled. This resulted in that the patients could control a virtual hand after having done the same motions with both "hands" as seen in Figure 1 [17]. In this thesis a similar method is developed which in the future will be used to train the Artificial Neural Network (ANN) for a robotic hand prosthesis.

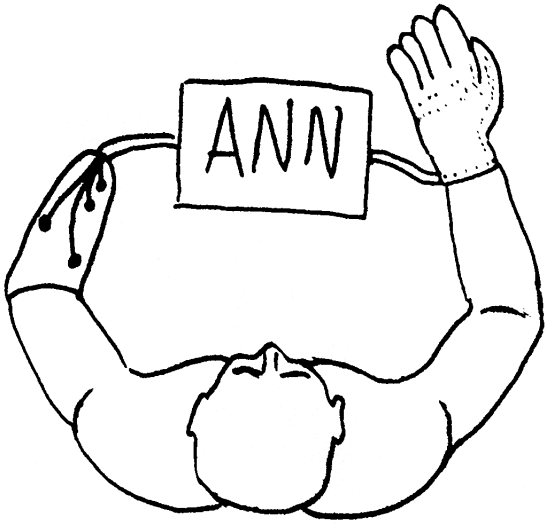

Figure 1: Sketch of the training where both hands are to do the same motion to train the ANN.

1.2 Purpose

The purpose of this master thesis is to develop a VRE that is able to show a virtual prosthesis controlled by various signals such as from an instrumented glove or pre recorded poses. Such a system open up possibilities when developing, testing and evaluating robotic prostheses. A test environment in Matlab is also to be developed to test and evaluate the code implemented in the VRE.

1.3 Project goals

The goal for this thesis work is to integrate an instrumented glove and a VRE to the current computer system developed by Integrum AB [14]. A simple GUI should be developed in Matlab. The virtual prosthetics could be controlled by the instrumented glove, computer commands or EMGs.

1.3.1 Sub goals

- Literature study on the current state-of-the-art and promising work.
- Create a virtual reality environment.
- Interface an instrumented glove with both Matlab and the VRE.

- Develop a graphical environment where the movements of a virtual hand can be controlled by an instrumented glove or computer commands.

1.4 Method

At first a theoretical study was performed to learn about similar projects and the methods chosen. Then a comparison between different solutions was performed and one method was chosen. The VRE was written in C++ with the aid of graphics with Ogre3D and the serial communication protocol from Openframeworks. The sensor values was sampled from an AcceleGlove provided by Integrum AB which gives six three-dimensional accelerometer readings, one for each finger and one on the palm. The test environment for calibration and evaluating new code was written in Matlab.

1.5 Outline

The introduction chapter describes the background of the project and specifies purpose, goals and method. In Chapter 2 the Integrum project is described and in Chapter 3, different methods of developing a Virtual reality environment is described and evaluated. In Chapter 4, the modeling of the system is shown and the different tools used are described and in Chapter 5 the orientation calculations and methods are shown. Last is the results, discussion, conclusion and future work.

2 Integrum project

This Chapter explains the Integrum project and gives background knowledge of why the thesis is made.

Integrum AB has developed an Osseointegrated Human-Machine Gateway (OHMG) that allows recordings from implanted electrodes in muscles and nerves by transferring them through the osseointegrated titanium implant. This method is out of the scope for this thesis and will not be further described. Alternatives to the OHMG are to use skin attached sensors or to use sensors attached to muscles or nerves inside the body through the skin. By not using the OHMG method there is a high risk of unstable readings (skin sensors) or infections (through the skin) and an alternative may be to reposition nerves from one part of the body more close to the skin to give better readings as described in [12]. The problem with bad readings is still present in such a system.

2.1 Training the ANN using a sensor glove

The training will be assisted with the use of a sensor glove named AcceleGlove described in Section 5.1, if the patient has one healthy hand. The healthy hand and the amputated hand are to do the same movements and serve as training information for the Pattern Recognition Algorithms (PRA).

The glove will be used in this thesis to read poses and motions from a healthy hand to help the robotic prosthesis to learn its motions. Other applications are to read poses for sign language and being able to both read what the person is saying such as in Hernandez-Rebollar et al. [11] or to help the person to learn the different signs. It is also possible to monitor motion patterns by people with injuries or disorders. One application might be to monitor the tremors for a Parkinson's or stroke patient and give medicine or other treatment and read if any change in movement and stability is achieved.

The present training for the patients consists of pictures of various poses of the hand that the patient should try to copy during a time window. This information in sampled and then processed by the PRA. This method may create problems while training. One problem is if the patient misunderstands the pose he or she should copy and the sampled values are used to train the PRA. Then this will make the neural net to behave in an unsatisfying way. The patient can also have problems with the time he or she is supposed to do the motion or pose during. Then the neural net is also trained with non optimal signals. The method of making the same movements with both the real hand with the sensor glove and the amputated limb and tracking both at the same time will hopefully help the PRA and thus giving a better result for the robotic prosthesis. The risk that the patient does not perform the same motions with both "hands" is of course still possible.

Since the VRE also should be able to take readings or poses from a computer instead of the sensor glove and illustrate them it is also possible to use it as a tool to see what the PRA has calculated and see if it matches the desired result. It is also possible to animate a motion of the hand to help the patient to understand how a specific motion is done.

2.2 Virtual Reality Environments

Virtual Reality Environments (VREs) are used in various ways. In games, industry, commercial or medical projects, it is often used as an aid for people to easier control

or observe a task. Some tasks are too abstract to be seen in numbers or even in an image, and therefore a VRE in three dimensions may be more intuitive and suitable.

In medicine there are many projects that use a virtual environment for rehabilitation. Many projects are directed to children that need stimulation to be able to understand and continue the rehabilitation. They might otherwise not understand the task or they may think it is too boring to perform it. A game where the child solves different tasks is a good way to keep their attention high and make them to be well motivated towards future sessions. A project suited for rehabilitation for children is the Paediatric Interactive Therapy System (PITS) where different sensors track the movements of a child's arms and integrating it in a VRE with different games [15]. The positive gain in rehabilitation is of course not only for children, but for all people in need. The game does not need to be very graphically advanced or have complex tasks. It can also be just a simple GUI to let the user have visual feedback and track the behavior of the patient. According to Sebelius [16, 17], their results on hand prosthesis training would not have been as good as it was without a VRE.

While doing the rehabilitation with a computer, the statistical information and the measure of improvement is much easier to calculate and analyze than if doing a for example an off-line test by hand with subjective observations. A shaky hand movement might be more or less shaky depending on how stable the previous patient is, or if the rest of the body is moving or not.

In Merians et al. [13] a VRE was used to exercise limbs for upper body recovery and the results were improved proximal stability, smoothness and efficiency of the movement path of the patients.

In most projects, the researching teams are developing their own GUI and detailed documentation is normally not available. The environment might be bought from an external company since the research is the rehabilitation method and not the graphical environment. They sometimes use a Software Development Kit (SDK) that includes a graphical environment for the hardware that the patients are using. Other projects use Matlab which is able to show three-dimensional graphics via the Simulink 3D Toolbox. Since the GUI most often is not documented, examples of how other have done is not possible.

According to Churko et al. [10]

> "there are nine phases of amputee rehabilitation: preoperative, amputation surgery, acute post surgical, pre-prosthetic, prosthetic prescription and fabrication, prosthetic training, community integration, vocational rehabilitation, and follow-up".

By simplifying or improving any of these phases, the life of someone who needs to go through them would greatly improve.

Other applications for a VRE has been tried and many projects where stroke patients have been using such programs have shown satisfactory results [9, 18]. Also here are simple games or similar used to simplify the task for the patient.

3 Method evaluation

In order to create a Virtual environment for the project, three methods are compared and evaluated. The data acquisition will be done by serial connection to the data glove that is connected by USB cable. The data sampling method is similar for most kinds of environment and the following discuss methods to create a graphical environment. This chapter describes the different methods, compares them and in the end a method is selected.

3.1 A new system

A possible method is to develop a new system from scratch in for example C++, C# and Ogre3D. Benefits by developing a new system are the possibility to tailor it to the project in the best possible way. A new system enables the possibility to optimize the methods for the project in the best way. Drawbacks are development time and the system might not be as optimal as projects developed for a long time by various people. In comparison with an open source project where many people is involved and the possibility to gain knowledge and feedback from others, the system might require more effort to become robust.

Ogre3D is a free 3D engine written in C++ that is designed to help creating hardware accelerated applications in an easier and more intuitive way. The engine is scene-oriented and has an interface based on world objects and focuses on the result rather than the underlying system libraries like Direct3D and OpenGL. The code is intuitive and is easy to extend further.[3]

Openframeworks is an open source toolkit for C++ functioning as a set of useful tools for programmers which simplifies making a prototype with graphics, sound, input and output. It is a package of useful libraries available for several platforms [4]. The Openframeworks toolkit is an state-of-the-art programming utility used in many projects dealing with vision systems or graphical applications. It is very easy to connect to various image sources or sensors, simple to analyze the information and present the results in an nice interface. Various projects can be found at Openframeworks website.[4]

Both Ogre3D and the Openframeworks toolkit are using C++ as language which enables a fast program able to handle sensor readings and graphical output with the possibility to implement it in a real time system. The useful visual libraries can also be used to render the virtual hand in an environment captured by a webcam. The patient then will see a hand where the prosthesis should be. This method is called Augmented Reality (AR) and is currently gaining popularity among various developers in different fields. One popular application is to use the AR with smart

phones and render information on the screen while filming the environment with the built in camera. This information can for instance be distance to the nearest cinema or similar[3, 4].

3.2 Matlab and Simulink

An alternative way is to use Matlab and Simulink in collaboration with the Simulink 3D Animation Toolbox (S3A) which is able to render graphics in realtime from Matlab and connect it using simulink. It is based on the Virtual Reality Modeling Language (VRML) which is mainly constructed with web development in mind in the same way as it's successor X3D [22, 5, 6]. This option simplifies debugging and evaluation due to Matlab's error handling methods and Simulink's intuitive programming but may be computational intensive. Due to the fact that Matlab works on OSX, Linux and Windows it is easy to transfer files and there is no need to compile for different architectures. S3A can import CAD models and uses textures to get a more natural look and has compatibility for standard peripherals such as mouse and keyboard to easy be able to pan and zoom in the environment. It is included in Simulink. This method is chosen by Tsepkovskiy et al. [19] where a CAD model of a mechanical hand is implemented in Simulink and animated with Simulink 3D Animation. It can therefore provide a quite good VRE for the thesis. As a start, Matlab can be used to sample and analyze data to simplify further development regardless of future language of the code.

Together with the Matlab real time toolbox [2] it is possible to convert Matlab and Simulink code to C code and use it in a real time environment. This may be critical when mapping a motion to the pattern recognition neural net.

3.3 Open source software

The third way is to use an open source environment such as OpenSIM, also known as Open Simulator which is widely used among researcher groups to build an environment for their specific needs. As mentioned earlier the benefit of using an open source program is that a community is developing the system and if the project is very active, new features could be added fast and bugs removed in an early stage. If the project is not so popular you are left on your own and to fully understand someone else's code fully without being able to discuss it can be quite cumbersome. On the other hand if the project is widely used, the possibility to get help and feedback could benefit the project. OpenGrasp is also an open source program that is possible to use.

As an alternative to VRML and X3D mentioned in Chapter 3.2 is COLLADA

which stands for COLLAborative Design Activity and is very similar to the previous mentioned software. Is also uses XML which is a standard in websites. Users can for instance drag and drop COLLADA-files directly to google earth or sketchup. In later versions there is a support for physics and the user can define for example friction and weight.

Another system that handles COLLADA-files is OpenGRASP which is based on OpenRAVE. OpenRAVE is a robot planning architecture which handles collision detection, kinematics and controls. Its extension OpenGRASP, is an adaptation for grasping robots. They are both open source and free to download.

3.4 Inverse kinematics

In order to make the model behave as realistic as possible it might not be possible to take raw readings from the sensor glove and map them to a specific pose. One way to get better values is to use inverse kinematics where constraints of the human hand are taken into count. Wang and Popović [21] used a glove with a color pattern on which was mapped by a webcam. They then used inverse kinematics to get a better pose of the mapped hand. One method is FABRIK (Forward And Backward Reaching Inverse Kinematics) algorithm described by Aristidou and Lasenby [8].

The AcceleGlove values are directly converted to an angle and no more manipulations need to be done if the graphic environment can assign angles directly to each joint of the virtual hand. The graphical model editor Blender uses a bone structure called armatures which can be described as a customizable skeleton. Each bone can have a connection to another bone and will deform the surrounding skin in a realistic way. This in cooperation with the Ogre3D system gives an easy way to modify an object according to angles.

3.5 Choice of method

The method chosen should be structured in modules so it can be easier improved in the future and have the possibility to be separated and let other modules communicate with those developed in this thesis. The proposed structure is illustrated in Figure 2.

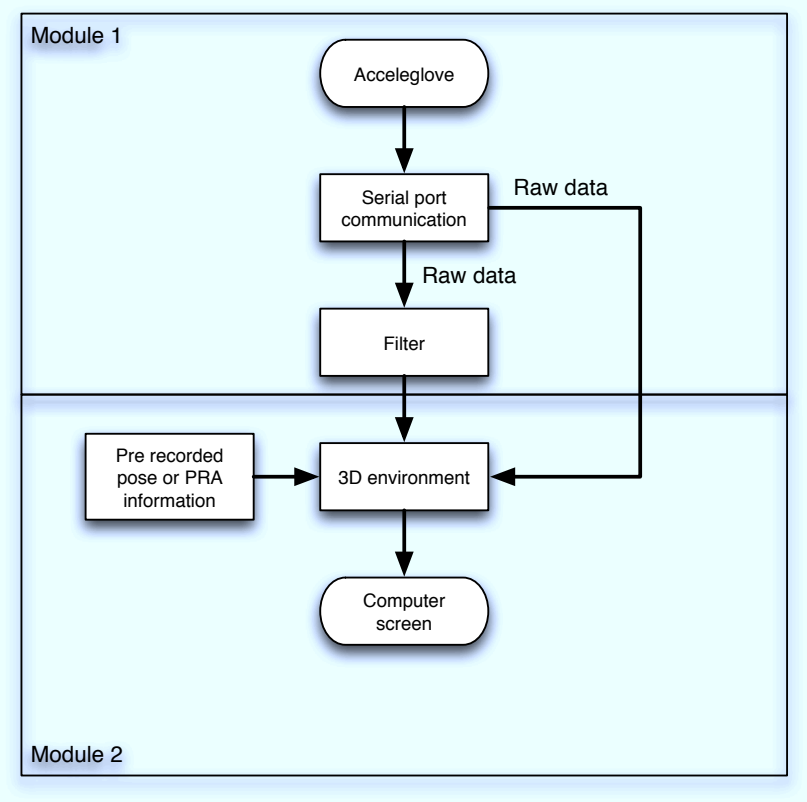

Figure 2: **Flow chart of the system**

The usage of any of the open source platforms stated in Section 3.3 is not suitable for the project since they best fit for multiplayer game development (OpenSIM) or industrial robotics (OpenGRASP). It is possible to have a good structure due to the frameworks and the possibility to get help by the open source community is an advantage. The project only needs small parts of the large open source frameworks such as the parts for rendering graphics. There is specific frameworks that manages this in a good way described later in the report.

Matlab with Simulink 3D Animation described in Section 3.2 can be a good and useful tool with easy debugging and rather fast prototype building. Matlab is capable of solving very advanced problems but due to its advanced toolbox also very computationally heavy and is more for analyzing than just representing data. This might complicate the possibility to get the program to run in real time. The real time toolbox should be of large help when running the program in real time, but it is still not as fast as using pure C/C++ code. Matlab might still be used as a prototype and test tool where new algorithms and methods may be tested.

The connection with Simulink gives a good overview of the project and often sim-

plifies debugging. This is a good alternative when deciding which foundation the project will have. One great advantage with Simulink is the ease of plotting and analyzing the different parts of a project due to the embedded tools and the flow chart layout of the functions.

To create a new environment using C++ with Ogre3D seen in Section 3.1 gives the advantage of being computationally very fast and effective, and the time to get a first prototype will be rather short. The endless possibilities to develop the project is a large advantage of using an own basis. The community using Ogre3D also seems to be active and the framework itself is continuously evolving. Further development of the project will probably be easier with the usage of this method and the usage of Ogre3D in comparison to use native OpenGL or Direct3D will save very much time due to good tutorials and the simplicity of the environment.

All methods described are possible to use for this thesis. The usage of C++ gives the advantages of fast computations and good possibilities for further development at the cost of starting from scratch. Matlab has easy prototyping and gives a intuitive view of the structure due to the graphical environment. The further development using Matlab might be limited by the computational speed. Both these languages give possibilities to communicate with the other parts of the whole project. The Ogre3D seems to be very easy to get started with and to create a first prototype. People using it are recommending it due to the straightness in the environment compared to use native OpenGL or Direct3D.

An inverted cost matrix seen in Table 1 is done to aid in the selection of the method. The different categories of the matrix are explained below. The numbers refer to the methods placement compared to the other alternatives. The best candidate will get 1 and the last 3. If two are equally good they get the same point and the remaining will get one placement behind.

Speed How fast the system can be and if it is possible to use in real-time.

Potential development What potential the project may have for further development. Simpleness of development and how advanced the result can be is considered mainly.

Time to prototype How early a first prototype can be created.

Support and accessible information How easy it is to find information and to get help using the method.

Each task is given a placement relative to all other methods. The three methods

are generalized even though for example the open source method is many methods. When a candidate is chosen, a more thorough decision might be done. The candidate with the least amount of points should be the best option to choose. Note that Matlab with S3DA assumed not to have the real time toolbox. Note also that some Open Source systems might be associated with a cost and some are free.

	Speed	Development potential	Time to first prototype	Support and information	Total
New system	1	1	1	1	4
Matlab w. S3DA	3	2	1	1	7
Open Source	2	3	2	2	9

Table 1: Comparison matrix between the different methods.

3.6 Conclusion of methods

The new system alternative gets the lowest amount of points and should therefore be the best method to use. A new system has the most promising conditions. The computations should go much faster with C++ than with Matlab and the potential for development and large amount of information on the Internet should be an asset for the project. The thesis will therefore be created using a new system from the ground up using Ogre3D for graphical rendering and Openframeworks for the serial communication.

4 Mapping and modeling

4.1 Ogre3D

Ogre3D helps the programmer to render 3D graphics and lets the programmer disregard of many calculations and focus on what the user should see. The program can be split up in an application and a frame listener. The frame listener takes care of all inputs and events that either the user or the interface makes and directs them to the corresponding function. The application is where what the user sees is defined.

The application consists of a start up routine where all objects are created. It is here the viewpoint is set up so the correct part of the model is shown and other simple attributes are declared. One main routine in the application is the createScene function where all objects are defined and all attributes to the objects are declared. It is here where the model of the hand is loaded and all angles to the fingers are updated. The structure of the program can be seen in the appendix.

4.2 Blender

Blender is a 3D modeling tool similar to any CAD program or 3D Studio Max. The hand is modeled in this environment and exported via a python script for use in Ogre3D. The usage of models from Blender in Ogre3D eliminates problems that might occur when creating the models directly by code. The Blender environment gives the possibility to test the model before loading it in the VRE. One good feature in Blender is the usage of armatures. An armature is seen as a bone and has a tip and root that can be attached to any place in any object and functions just as a hinge or bone in the human body. Seen in Figure 3 is the armature used in the VRE. The system of armatures and it's parts, the bones, are well developed and each part can be adjusted in a easy way to deform the vertices of the object as desired. Each armature can be split up in many bones and a parent-child relationship is created between them so when a parent is moved or rotated, all children are subject to the same transformation. This simplifies a more elaborated model since the model by itself takes care of all relationships between vertices and bones. The possibility to easy elaborate the model without altering the basic relationships of for example each finger greatly aids any future improvements if a more advanced sensor glove is to be used.[1]

4.3 Modeling of the hand

In Ogre3D all objects are assigned a relative position. It could be bound to the world, a parent or local. Each finger is a child of the palm and thus making the movement of the fingers relative to how the palm is rotated in space. If this was not the case, the fingers would bend in unnatural directions while the hand was rotated. At first a simple hand made by blocks is made to test the glove and see if all signals are mapped correct. See Figure 4. Since the model is only a shell for the inside armature, the model can be improved later on and textures, shaders and other visual effects may be applied to get a more natural and realistic look. Later a more realistic model was implemented seen in Figure 5 and finally the hand was extended with an arm seen in Figure 6 and the texturized model is seen in Figure 7.

The armature from Blender works like a bone structure and Ogre3D uses this information to manipulate the model in a kinematically correct way. There are no boundaries for how the joints can be rotated, only that they are attached to a bone and not able to translate in space if the parent bone is static. These boundaries can be programmed but it is, instead the actual physical hand that limits the movement of the hand. A person is not able to rotate his or hers fingers in an unnatural way and no calculations for this has to be done. The actual inverse kinematics and calculations of how the hand and fingers are rotated in space is calculated natively by Ogre3D.[1, 3]

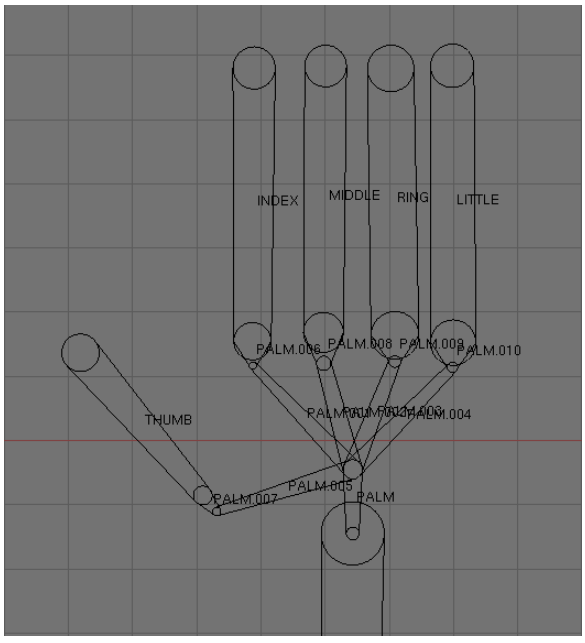

Figure 3: The bone structure of the hand.

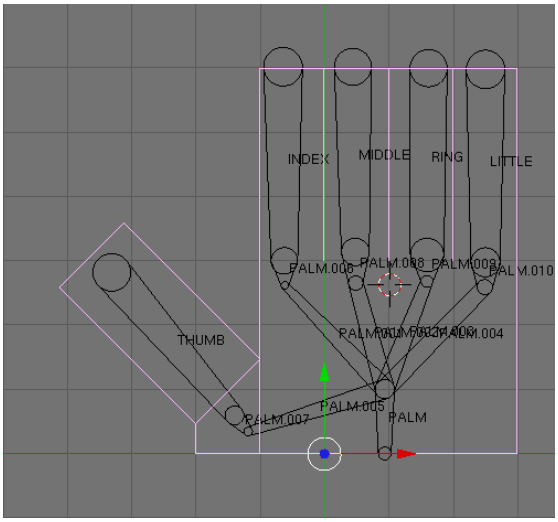

Figure 4: Early hand model and the bone structure inside.

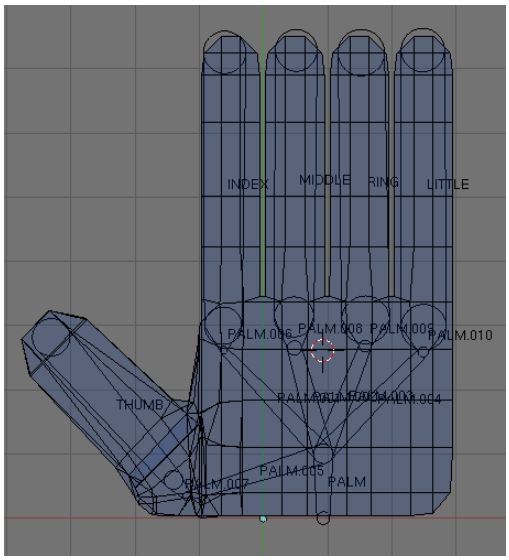

Figure 5: A more advanced hand model.

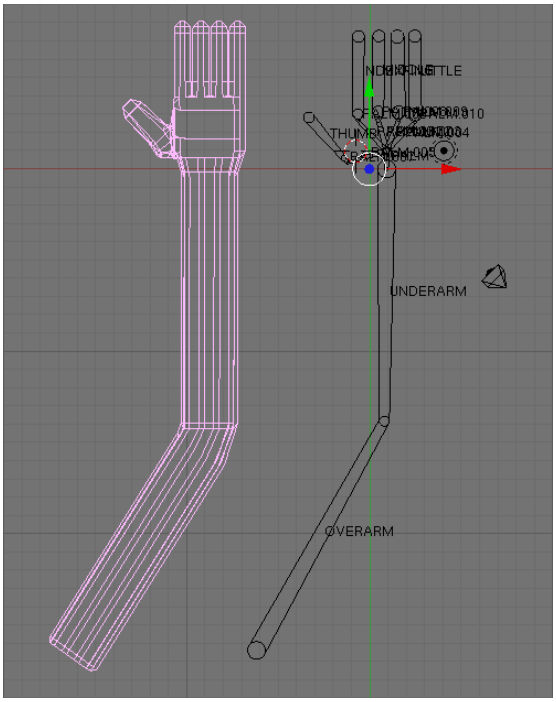

Figure 6: The final arm and the bone structure.

Figure 7: The final arm with texture.

5 Orientation estimation

As seen in Figure 8, pitch is the rotation around the x axis and roll around the y axis. Yaw is the angular offset from the gravitational z-axis and the accelerometer. These are also called Euler angles and are chosen for the simple calculations. It would be possible to use quaternions or similar to eliminate possible singularities and gimbal lock. The VRE does not need to be very accurate since the representation is mainly for visual feedback to the user and the raw values itself will be used for the neural training. Since the training is not in the scope for this thesis, it will not be further described. Using the pitch and roll values gives a good representation if neither the hand nor the fingers rotate more than approximately 85 degrees (theoretically 90 degrees). The yaw value in this application is not the rotation of the z-axis as in ordinary flight rotations but as described the angular offset from the z axis. Using the yaw offset value the model is able to tell if it is upside down or not. Even though this is calculated for all sensors, the fingers are flipped only if the palm is upside down since the joints for the fingers would be inverted otherwise. The environment will only handle static poses and not any dynamic poses since it does not integrate the accelerometer values.[20]

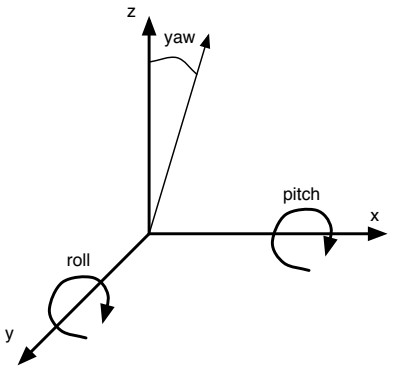

Figure 8: The relationship between pitch, roll and yaw.

5.1 Sensor glove

There are many sensor gloves on the market and they use various methods for sensing the pose of the hand. Some use resistors to read if the hand is open or closed and others use gyroscopes and accelerometers. The AcceleGlove used in this thesis uses accelerometers is seen in Figure 9 and have six sensors (one on each finger and one on the palm), all reading the acceleration in three axes. Each sensor reads the acceleration of the object it is attached to. It the object is static it will only read the gravity vector and from this information it is possible to calculate the orientation of the object. The placement of the sensors on the glove is seen in Figure 10.

This gives an 18 DOF system able to get an accurate pose of the hand, even while rotated. Each sensor has an orientation range of 180 degrees and an acceleration range of $+/-1.5g$. They measure data with a sensitivity of $0.0287m/s^2$ and has a resolution of 10bits from the A/D-converter. The maximum sampling rate from the glove is 35 Hz, which means 630 axes per second.

Figure 9: AcceleGlove sensor glove.

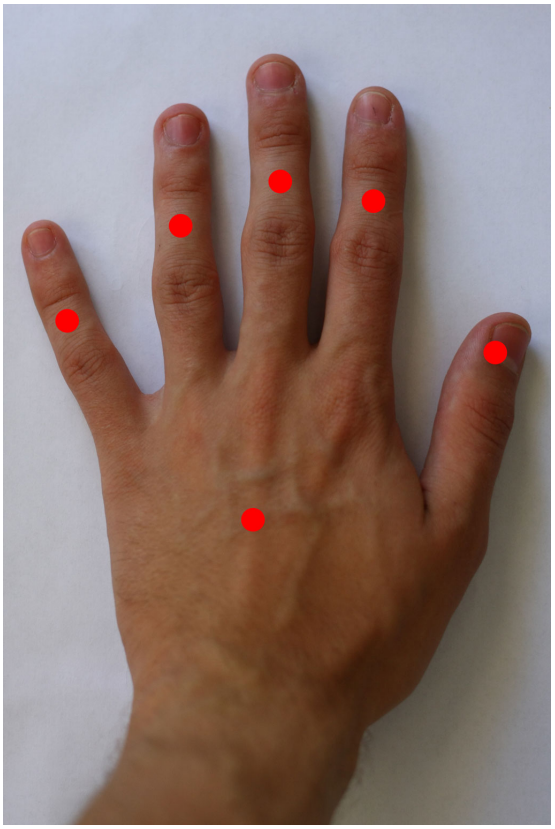

Figure 10: AcceleGlove sensor placement. One on each finger and one on the hand.

5.2 Reading the Sensors

The AcceleGlove sends an array of values if a letter is sent to the device [7]. The array is received by a buffer that is read by the software. In order to make sure all values are captured a loop is defined to receive a pre set number of values and if not all are received, the function tries again until all information is captured. This method assures that each sensor is placed in the right location in the value array and prevents mismatching due to any offset. The Openframeworks package is used to open the serial port and the framework sends error messages if something is wrong.

5.3 Filtering and angle calculation

The raw signals from the accelerometers on the hand need to be filtered to avoid a shaky appearance of the hand in the VRE. The noise from the accelerometers is about 1 degree while being static. This might not be much for this application

but the vibrations created in the model gives a unrealistic look of the VRE. If you slowly move your hand you will affect the accelerometers and a fast movement will give an unreliable result. A simple running average- and a Kalman filter is implemented. The running average filter was chosen for the simple implementation and the Kalman filter for its good noise reduction properties. The Kalman filter tracks the movement and calculates with statistics how the hand can be moved and how it is most probable to do. The program can easily be switched between different filter algorithms to evaluate which is the best one for the current task to be performed.

After the filter of choice has removed some noise from each sensor the orientation of each accelerometer set is calculated as

$$pitch = -tan^{-1}(a_y/\sqrt{a_x^2 + a_z^2}) \quad (1)$$

$$roll = tan^{-1}(a_x/\sqrt{a_y^2 + a_z^2}) \quad (2)$$

$$yaw = tan^{-1}(\sqrt{a_x^2 + a_y^2}/a_z) \quad (3)$$

where a_x, a_y and a_z are the accelerometer values from each axis. The yaw value is as stated not the rotation around the z-axis as usual in flight orientation calculations but the angular offset from the z-axis. This gives information if the sensor is held upside down or not. By removing the noise before the angle is calculated, the influence of the noise is less than if the angles are calculated before the noise is removed. This is due to the non linear calculations from accelerometer values to orientation angles.

5.3.1 Kalman filter

A Kalman filter was implemented and uses the accelerometer readings as input. The model the filter is following is a signal with added noise. Since the sensors are workingindependent of each another it is sufficient to assume this very simple model. This also makes the filter very fast. The linear Kalman filter thus only handles tilt in one dimension and this value is later used together with the other sensors to calculate the orientation of the three-dimensional accelerometer.

The data model used for the static pose is

$$x(k+1) = Ax(k) + w(k) \quad (4)$$

$$y(k) = Cx(k) + v(k), \quad (5)$$

where x is the state, A is the state-transition matrix, y(k) the measurement, C the measurement matrix, w(k) the state noise and v(k) the measurement noise. Since only one state is handled at a time and the model only assumes noise, the

state-transition matrix A and the measurement matrix C are set to one.

The noises v and w are assumed to be drawn from a zero mean multivariate distribution with their corresponding covariances R and Q.

The filter is initiated with x_k and P_k as 0 and the covariance R as 0.01 and Q as 0.001 and the Kalman gain is calculated

$$K = P_k(P_k + R)^{-1} \tag{6}$$

which is used to correct the estimated state

$$\hat{x}_k^+ = \hat{x}_k + K(y_k - \hat{x}_k) \tag{7}$$

where the Kalman gain K is used to update the estimate by modifying the difference between the measured value from the accelerometer y or the calculated angle and the previous estimated value. Finally the covariance value (a matrix if more states were taken into account) is updated:

$$P_k^+ = P_k - KP_k. \tag{8}$$

The time update for the next state is

$$P_k = P_k^+ + Q \tag{9}$$
$$\hat{x}_k = \hat{x}_k^+ \tag{10}$$

5.3.2 Running average filter

As a complement to the Kalman filter a simple running average filter was implemented. The running average filter is a FIR-filter of low pass kind that takes the latest readings from the sensors in a list and calculates the total average as output. In the next iteration the last sample is discarded and the newest sample is taken into account when calculating a new average, thus making a running average. If using a short filter length the filter removes small vibrations but if the filter is to long the result is going to be a much delayed motion and not very useful.

5.4 Frame update

The filtered angle values are sent to the Ogre3D VRE and each angle is set as an property of each bone in the hand and the view is then rendered. The Ogre3D

environment uses the bone armature to update the current pose of the hand and the result is shown to the user. This is done as fast as the VRE can and assumes all frames as static poses. It does not take any dynamics between the frames into account more than the Kalman filter calculations.

6 Results and discussion

The VRE is working satisfactory and samples all 18 states of the AcceleGlove and rendering them in the 3D model of the hand. The result can be seen in Figure 11. The system is able to render a model of a left or right hand and sample signals from either a left or a right glove. It is possible to use the right glove and render a left hand to simulate a non existing hand of the patient. The environment is also capable of taking signals from the user to state a specific pose or render a desired motion to show a patient how it is supposed to look.

Figure 11: The VRE in use.

The environment is capable of receiving values from the glove but also commands directly from the user and pre recorded values. EMGs can be converted to poses by the PRA and then sent to the environment in the same way as user commands.

The environment is also able to record motions from the user and play them at any time. Static poses are of course also possible to show.

A GUI in Matlab has also been developed to sample data from the sensors and plot the signals from each axis. The GUI can be seen in Figure 12. This simplifies testing new filters and methods before it is implemented in the VRE or any of the other parts of the hand prosthesis project. From Matlab, the Kalman filter was evaluated and in Figure 13 a plot with samples from pitch, roll and yaw from one sensor can be seen with their corresponding Kalman filtered values. The motions made are normal motions of the hand with a pitch motion followed by roll. The quotient of Kalman filter parameters q and r control the behavior of the filter. Particularly the trade-off between noise reduction and delay. A reduction in the quotient will lead to shorter signal delay but less noise reduction.

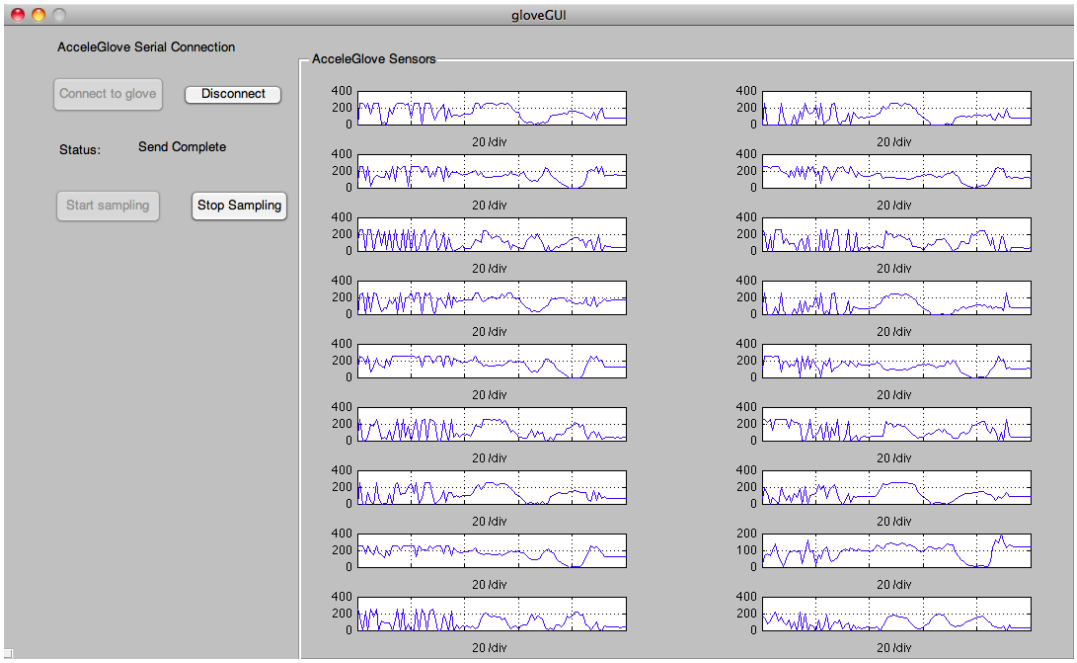

Figure 12: GUI in Matlab.

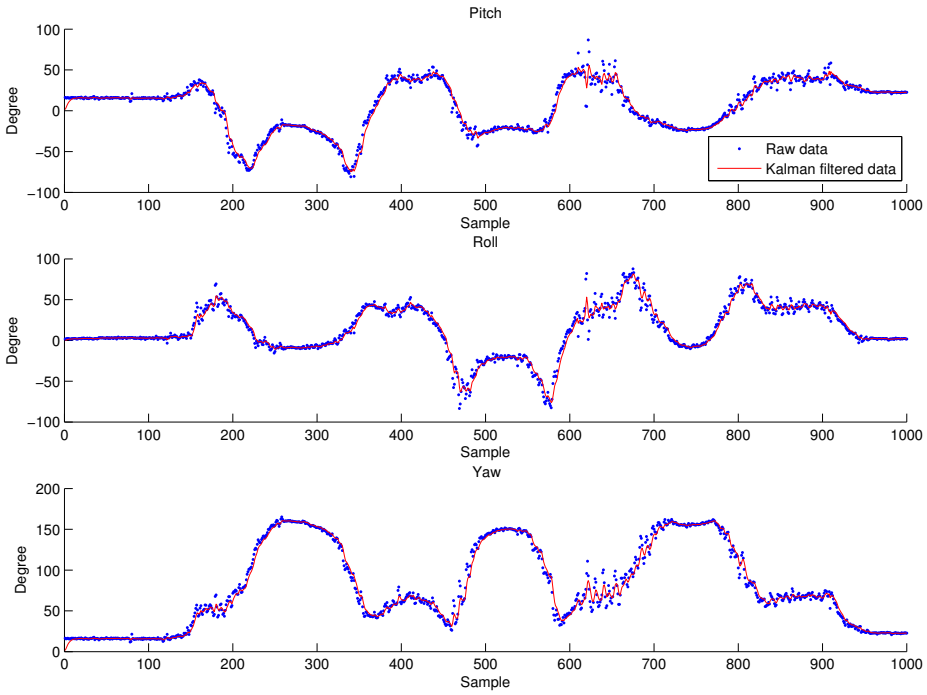

Figure 13: The kalman filtered signals from pitch, roll and yaw angles together with their corresponding raw values.

The filters are working and removes noise sufficiently for the application. The running average filter removes almost all noise if the filter is of length 4 or more. A higher value makes the hand move with a large delay and becomes more and more useless due to the long time between movement and rendering. The Kalman filter can be calibrated to reduce more noise if necessary for any other task but the noise reduction is sufficient for the representation of the model.

Since the VRE is mainly for visual feedback of static poses to the user, the values does not need to be very accurate and the most important is to make the representation smooth which the Kalman filter does.

Due to the fact that the glove only has one sensor on each finger and not placed at the tip, there is no way to track the movement of the separate joints of the finger. This leads to problems showing for example a grasping behavior as illustrated in Figure 14. To solve this, your glove must have more DOF than now available or a kinematic model of the finger movements must be employed. This is only a cosmetic problem and not useful for the PRA at this point. The VRE has no limits in improving the model to accept more states such as joint sensors or similar. The bone armature is very easy to split to include more bones using Blender and thus

be able to as mentioned above to show a grasping hand. Another issue concerning the sensors is the fact that they sample acceleration. As well as sensing the direction of the gravitational field the sensor will also pick up the acceleration when the hand moves. This is not a major problem but if moving horizontally, e.g. parallel to the gravitation the sensors are not able to notice any change. This makes many poses or changes impossible to observe without including a dynamic model of the movements of the hand and estimate absolute position by integrating the sensor signals. The VRE has not any limitations and is able to show any pose given.

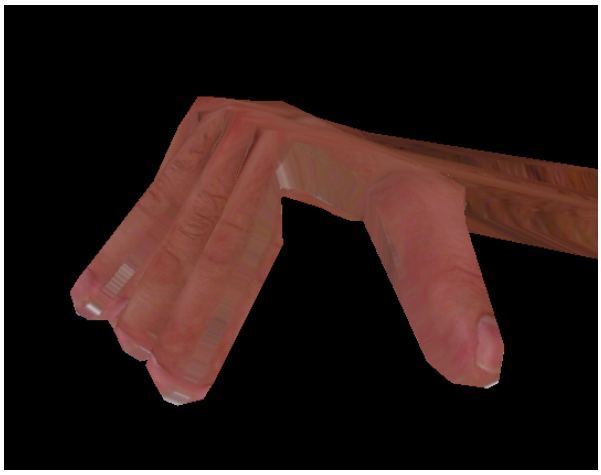

Figure 14: A problem concerning the use of few DOF.

The use of Ogre3D has been convenient due to its very intuitive objective programming. The framework is well arranged and tested which is noticeable when developing the environment due to very useful functions and good structure of the system. This will probably aid in any further development of this VRE. It is well known that it can be hard to jump in to another person's code but due to the easy layout of the framework the future development will probably be easier.

The usage of Ogre3D gives a useful base for further development and good possibilities to make the hand model more realistic and might help the patient to better accept the environment or to better understand the various poses and motions to be performed.
An arm, from wrist to shoulder, is included in the model even if it does at this time not have any sensors to control it. The AcceleGlove has a connection to a arm sensor embedded on the circuit board. However the arm sensor is not available on the market today.

7 Further development and future work

7.1 Dynamics

The accelerometer values can be integrated and thus provide a more dynamic behavior of the motions and then also be able to handle non static poses. This can be implemented to give a better visual feedback but it is not sure it it will give better values for the neural training.

7.2 Communication

The communication between different parts of the code and the other project at Integrum is not yet developed but will in the future be done using sockets. The sockets consist of a server and a client part. One server can be connected with many clients and new modules can be constructed if new methods or systems are to be evaluated. The socket communicates using packets that are pre defined and contains just the information needed for the system. This information is for example orientation of the sensors, if it is the right or left hand and other similar descriptors. One advantage of using sockets is that it is possible to communicate between various platforms such as Macintosh, PC and different programming languages such as C/C++, Java and Matlab. This makes it possible to have some parts of the project written in for instance C in an embedded environment and try new code developed in Matlab. This also gives the benefit of having different teams to try code in their favorite language and environment. Since the protocol were not operational during the time the VRE was developed, the sockets is not used but is relatively easy to implement when it is ready thanks to the modular structure of the environment.

7.3 Extensions

In the future it will be possible to observe the patient using a webcam and render a 3D figure of the virtual hand in connection with the patients limb using Openframeworks and its AR toolbox. A sketch of this can be seen in Figure 15. This gives the effect that a hand is attached to the limb and the patient is able to move the hand using his or hers healthy hand. If both hands are missing, the virtual hand can be controlled using a animation or with the EMG signals. A possibility is that the patient accepts the training program easier.

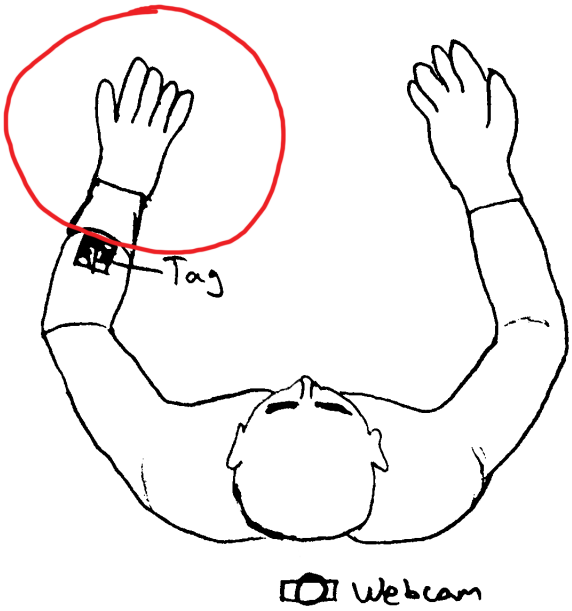

Figure 15: A sketch of the AR concept. The red ring shows what the AR toolbox adds to the user view. The left hand is a computer model added to let the user experience a hand. The tag tells the AR toolkit where the model should be attached.

8 Conclusions

The VRE meets the specifications that was set and will probably be tested with patients in a near future. To get it more integrated in the Integrum project, the communication protocol between the VRE and the ANN and the rest of the parts of the project has to be developed, i.e using sockets.

The use of the AcceleGlove was sufficient for the use for the project but to get more reliable poses and a more reliable graphical representation, a glove with more sensors has to be used. The fact that accelerometers are not affected by horizontal movement if the samples are not integrated (which will demand a much more comprehensive model) makes them less suited for more complex motions.

Ogre3D has been a very suiting environment in cooperation with the Blender tool. The intuitive structure and simpleness of importing new models has made the model of the hand graphically more realistic. Some problems with the export from Blender to Ogre3D took long time to solve but the usage of the two tools is recommended.

Even though Openframeworks is only used now as a serial port communications controller it can be of great usage in future development due to its large library. The AR package would be a good extension of this project to help the patient in the training.

There is still need for user or patient testing to evaluate the VRE and modifications will be done according to the feedback received.

References

[1] Blender.org, May 2011. URL `http://www.blender.org/`.

[2] Matlab real time toolbox, January 2011. URL `http://www.mathworks.com/products/rtw/`.

[3] Ogre3d webpage, Feb 2011. URL `http://www.ogre3d.org/`.

[4] Openframeworks, January 2011. URL `http://www.openframeworks.cc/`.

[5] Simulink 3d animation, January 2011. URL `http://www.mathworks.com/products/3d-animation/index.html`.

[6] Adel Al-Jumaily and Ricardo A. Olivares. Electromyogram (emg) driven system based virtual reality for prosthetic and rehabilitation devices. In *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '09, pages 582–586, New York, NY, USA, 2009. ACM.

[7] Inc. AnthroTronix. Acceleglove manual, January 2011. URL `http://www.acceleglove.com/AcceleGloveUserGuide.pdf`.

[8] A. Aristidou and J. Lasenby. Motion capture with constrained inverse kinematics for real-time hand tracking. In *Communications, Control and Signal Processing (ISCCSP), 2010 4th International Symposium on*, pages 1 –5, 2010.

[9] J. Broeren, A. Bjorkdahl, L. Claesson, D. Goude, A. Lundgren-Nilsson, H. Samuelsson, C. Blomstrand, K. S. Sunnerhagen, and M. Rydmark. Virtual rehabilitation after stroke. *Stud Health Technol Inform*, 136:77–82, 2008.

[10] J.M. Churko, A. Mehr, A.G. Linassi, and A. Dinh. Sensor evaluation for tracking upper extremity prosthesis movements in a virtual environment. *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 2392 –2395, september 2009.

[11] J.L. Hernandez-Rebollar, N. Kyriakopoulos, and R.W. Lindeman. A new instrumented approach for translating american sign language into sound and text. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 547 – 552, May 2004.

[12] T. A. Kuiken, G. Li, B. A. Lock, R. D. Lipschutz, L. A. Miller, K. A. Stubblefield, and K. B. Englehart. Targeted muscle reinnervation for real-time myoelectric control of multifunction artificial arms. *JAMA*, 301:619–628, Feb 2009.

[13] Alma S Merians, Gerard G Fluet, Qinyin Qiu, Soha Saleh, Ian Lafond, Amy Davidow, and Sergei V Adamovich. Robotically facilitated virtual rehabilitation of arm transport integrated with finger movement in persons with hemiparesis. *Journal of NeuroEngineering and Rehabilitation*, 8(1):27, May 2011.

[14] Max Jair Ortiz C. Biosignals acquisition and pattern recognition for robotic prostheses. M.sc. thesis, Department of Applied Physics, Chalmers University of Technology, Göteborg, Sweden, Jan 2010.

[15] P. Pyk, D. Wille, E. Chevrier, Y. Hauser, L. Holper, I. Fatton, R. Greipl, S. Schlegel, L. Ottiger, B. Ruckriem, A. Pescatore, A. Meyer-Heim, D. Kiper, and K. Eng. A paediatric interactive therapy system for arm and hand rehabilitation. In *Virtual Rehabilitation*, pages 127 –132, 2008.

[16] Fredrik Sebelius. Myoelectric control for hand prostheses. Technical report, Department of Electrical Measurements, Lund Institute of Technology, 2004.

[17] Fredrik Sebelius, Birgitta N. Rosén, and Göran N. Lundborg. Refined myoelectric control in below-elbow amputees using artificial neural networks and a data glove. *The Journal of Hand Surgery*, 30A(4):780–788, July 2005.

[18] S. Subramanian, L.A. Knaut, C. Beaudoin, B.J. McFadyen, A.G. Feldman, and M.F. Levin. Virtual reality environments for rehabilitation of the upper limb after stroke. In *Virtual Rehabilitation, 2006 International Workshop on*, pages 18 –23, 0 2006.

[19] Y L Antonov Tsepkovskiy, CV Kocev, and N Shoylev F Palis. Development of a 3d and vrml virtual hand models. In *Journal of the University of Chemical Technology and Metallurgy*, 43(1), pages 159–164, 2008.

[20] Kimberly Tuck. Tilt sensing using linear accelerometers. Technical Report Rev 02, Accelerometer Systems and Applications Engineering Tempe, AZ, June 2007.

[21] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM Trans. Graph.*, 28:63:1–63:8, July 2009.

[22] Wikipedia. Wikipedia - vrml, January 2011. URL `http://en.wikipedia.org/wiki/VRML`.

9 Appendix

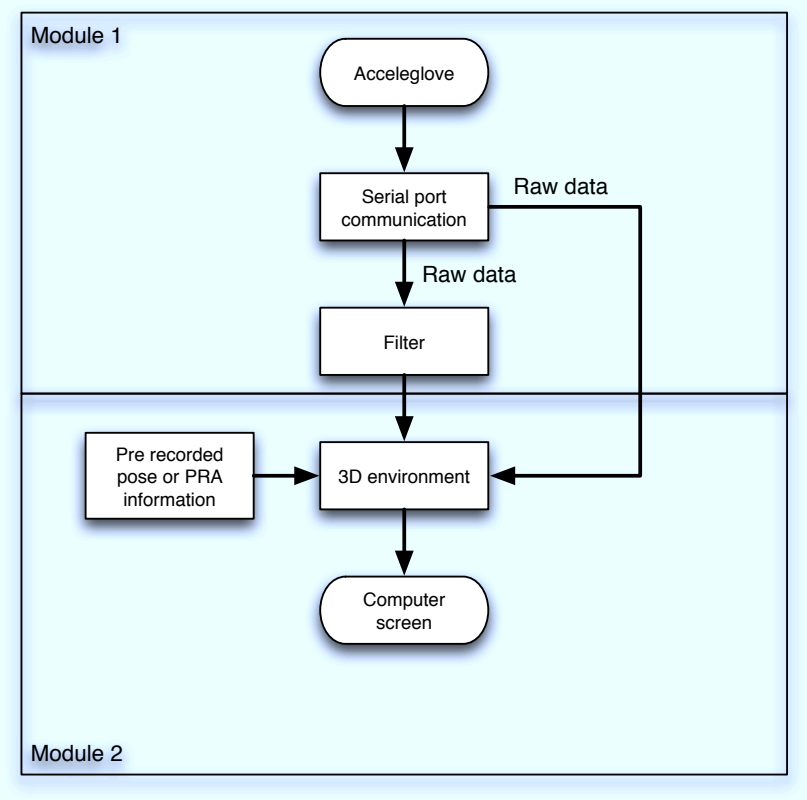

Figure 16: **The overall modular structure of the software.**

9.1 Ogre3D structure

Pseudo code of the functions concerning the Ogre3D functions.

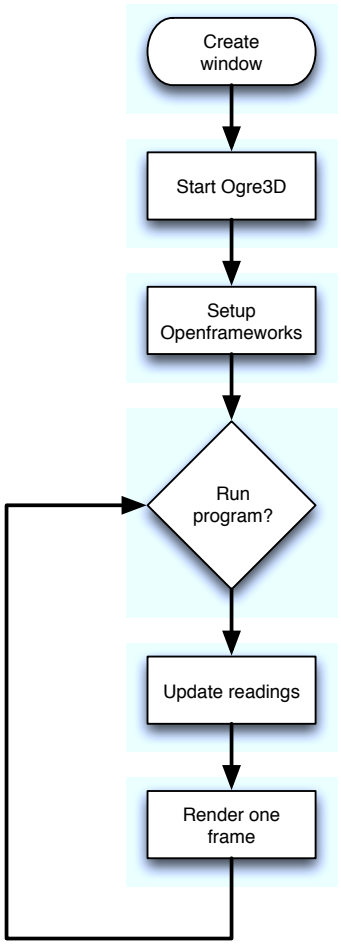

Figure 17: **Main loop for the VRE.**

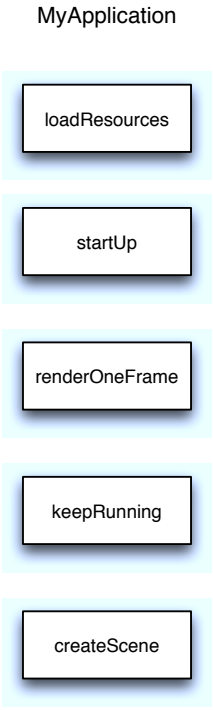

Figure 18: **The functions for MyApplication.**

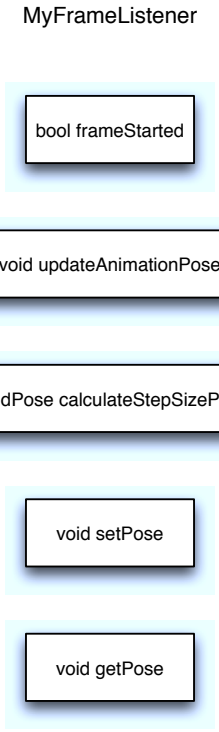

Figure 19: **The functions for MyFrameListener**

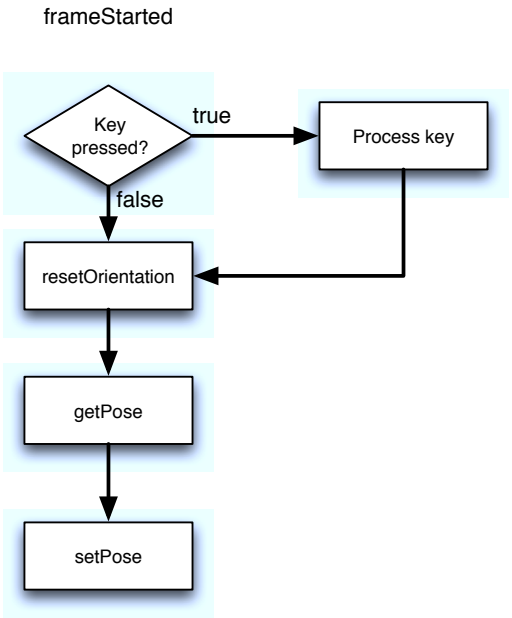

Figure 20: **Flow chart for frameStarted.**

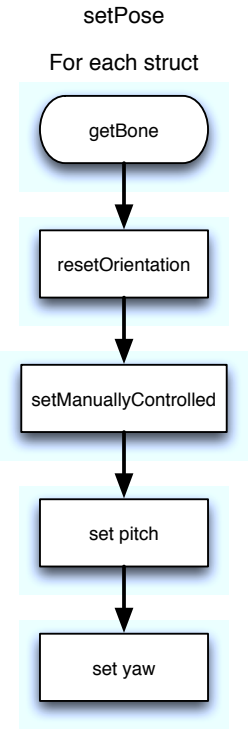

Figure 21: **Flow chart for setPose.**

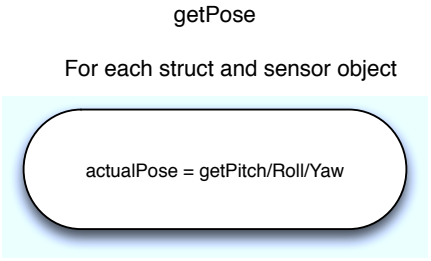

Figure 22: **Flow chart for getPose.**

9.2 Sensor functions

Pseudo code of the functions concerning the processing of the sensor signals.

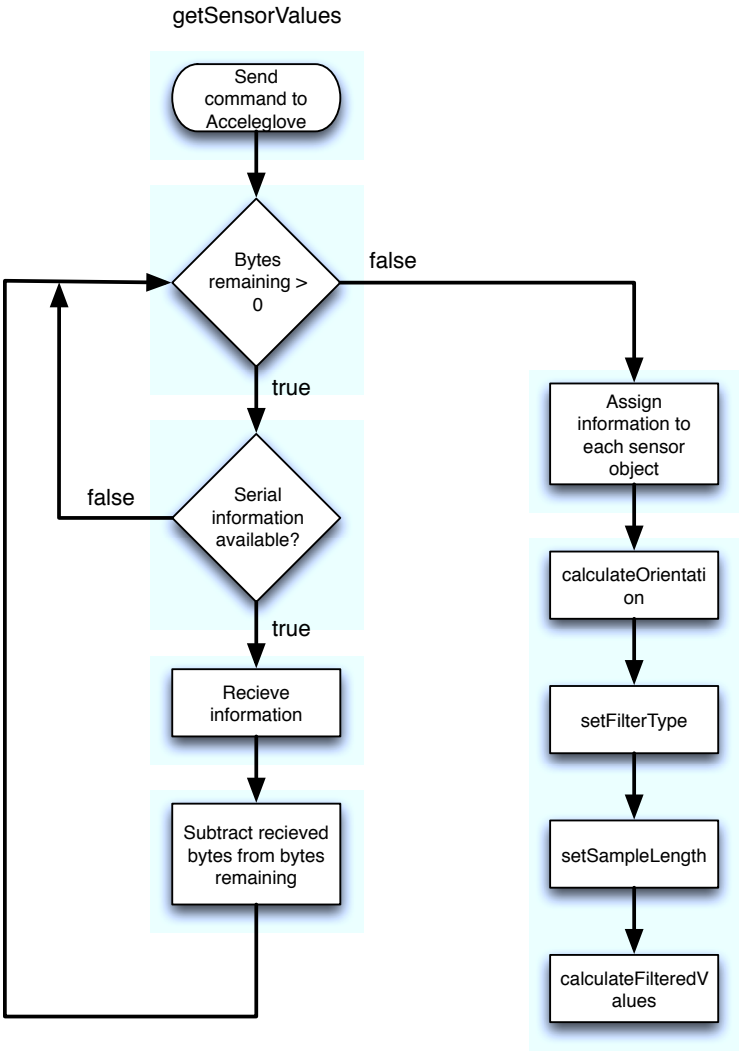

Figure 23: **Flow chart for getSensorValues.**

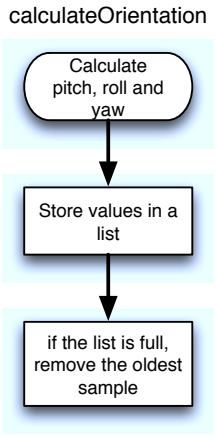

Figure 24: **Flow chart for calculateOrientation.**

Figure 25: **Flow chart for setFilterType (top), setSampleLength (middle) and clearSamples (bottom).**

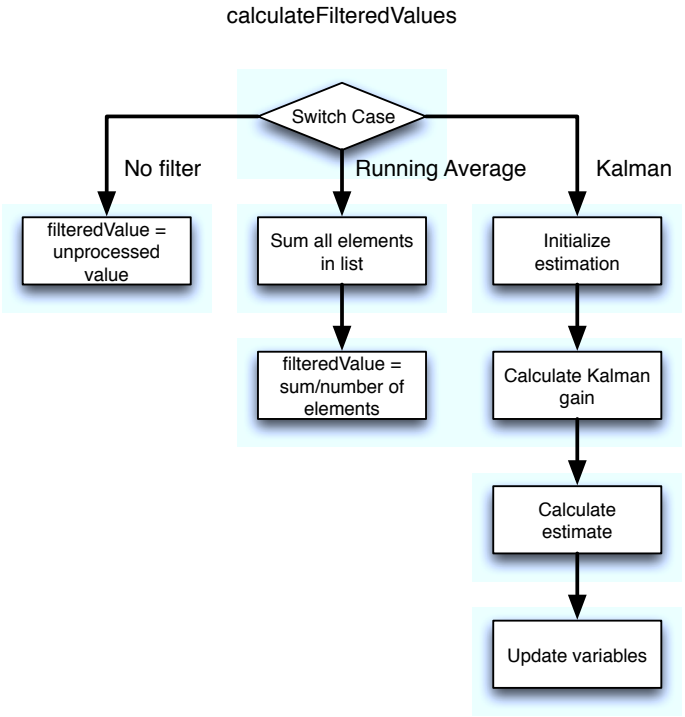

Figure 26: **Flow chart for calculateFilteredValues.**