

TECHNISCHE UNIVERSITÄT DRESDEN

FACULTY OF COMPUTER SCIENCE
INSTITUTE OF SOFTWARE AND MULTIMEDIA TECHNOLOGY
CHAIR OF COMPUTER GRAPHICS AND VISUALIZATION
PROF. DR. STEFAN GUMHOLD

Bachelor's Thesis

for obtaining the academic degree
Bachelor of Science

Development of a Natural VR User Interface Using Haptic Gloves

Lucas Waclawczyk
(Born 26. April 1998 in Bad Langensalza, Mat.-No.: 4686687)

Tutor: Prof. Stefan Gumhold

Dresden, September 6, 2020

Task Description

Motivation: Interaction within a VR environment – be it with application controls or the environment itself – is usually mapped to the manipulation of some controller in current consumer-grade VR systems and applications. This causes an additional level of abstraction from the virtual world that can be detrimental to immersion. However, an increasing number of alternative controllers, usually in the form of gloves, that more closely match the interaction modes of the user’s own hands are becoming available. They enable more accurate tracking of hand poses than is possible with affordable optical tracking technology while also being capable of providing haptic feedback. The goal of this bachelor’s thesis is the design, implementation and evaluation of a natural VR user interface that the user can manipulate directly using their hands. For matching the user’s hand movements, the limited finger tracking provided by the AvatarVR^[nd] gloves should be combined with a suitable hand pose generation strategy as well as Lighthouse-based position and orientation tracking^[lig].

Individual Tasks:

- Literature review on VR glove technology and applications
- Design of a hand data model that is suitable for use with both the AvatarVR API and the Leap SDK 2.0^[lea]
- Design and implementation of a strategy for context-sensitive generation of hand poses from AvatarVR finger orientation data
- Design of a suitable test scenario for a direct manipulation-based natural user interface (buttons, sliders, levers, etc.)
- Prototypical implementation of a CGV Framework plugin, with the following features: VR rendering of the designed test scenario
- Real-time coupling of AvatarVR finger tracking with Lighthouse-based tracking of the glove position and orientation, and generation of haptic feedback from the tracking results
- Stylized rendering of the user’s hand(s) using suitable Framework facilities (e.g. `box_renderer`)
- Qualitative evaluation:
 - Tracking precision and quality of generated hand poses

-
- User performance when interacting with the interface
 - Quantitative evaluation of tracking performance (e.g. latency)

Optional Goals:

- Comparative Evaluation:
 - Controller-based interaction, e.g. via some controller-tailored abstract interface, or pointing and grabbing of the natural interface
 - Full hand tracking via Leap Motion sensor
- Incorporation of real-world objects into the natural user interface (e.g. a real table on which the virtual controls are situated) and analysis of the effect on perception of the controls
- Design and implementation of additional natural interaction modes not based on direct manipulation, e.g. gesture recognition
- Rendering of the user's hand(s) as skinned mesh

Declaration of authorship

I hereby declare that I wrote this thesis on the subject

Development of a Natural VR User Interface Using Haptic Gloves

independently. I did not use any other aids, sources, figures or resources than those stated in the references. I clearly marked all passages that were taken from other sources and cited them correctly.

Furthermore I declare that – to my best knowledge – this work or parts of it have never before been submitted by me or somebody else at this or any other university.

Dresden, September 6, 2020

Lucas Waclawczyk

Kurzfassung

abstract text german

Abstract

abstract text english

Contents

1	Introduction	4
1.1	Principles of Data Gloves	4
1.2	Examples and Applications of Data Gloves	6
1.2.1	Sign Language Recognition	6
1.2.2	Daily Activity Recognition	7
1.2.3	Refined Myoelectric Control in Below-Elbow Amputees	9
1.2.4	A Lightweight Force-Feedback Glove	10
1.2.5	Commercial Products	11
2	Development	13
2.1	Devices and Software	13
2.2	Skeletal Hand Model for the Avatar VR Haptic Glove	15
2.2.1	Abstraction of the Human Hand Skeleton	15
2.2.2	Technology and Function of the Avatar VR	15
2.2.3	Using Data From the Avatar VR to Visualize the User's Hand	17
2.3	Project Idea and Structure	21
2.3.1	Main class	21
2.3.2	Bridge	21
2.3.3	Space, Targets and Phasers	21
2.3.4	Control Panel	22
2.3.5	Hands	23
2.3.6	Headup Display	23
2.4	Calibration	24
2.5	Space, Targets and Phasers	26
3	Evaluation	29
3.1	Participating Subjects and Preparation	29

3.2	Qualitative Evaluations and Suggestions	29
3.2.1	Usage of the Avatar VR	29
3.2.2	Haptip Feedback	31
3.2.3	Timing	31
3.2.4	Intuitive Grabbing and Lever Reaction	31
3.2.5	Slider Reaction	32
3.2.6	Targets and Phasers	33
3.2.7	Calibration	33
3.3	Quanitative Evaluation	33
4	Conclusion	35
	Bibliography	38

1 Introduction

In their review *Haptic display for virtual reality: progress and challenges*^[WGL⁺19], Wang et al. state that Virtual Reality (VR) was born in 1965 with Ivan Sutherlands proposal of a concept named "The Ultimate Display"^[Sut65]. In that report of the IFIP Congress, Sutherland names immersion, interaction and imagination as three important features of VR.

There has been a multitude of approaches to making interaction with virtual environments possible^[LHT⁺19]. Touch screens and stylus pens have become quite common in everyday life. Devices capable of capturing scene depth like Microsoft Kinect and Leap Motion make touchless interaction possible, e.g. during games. This thesis focuses especially on wearable devices called *data gloves*.

1.1 Principles of Data Gloves

The general principles of data gloves can be divided into two main parts: sensing and actuation. The first part deals with the issue of capturing a user's hand poses and motions and relaying that information to a computer for further processing. In most cases, a fabric glove is equipped with an inertial measurement unit (IMU) placed on each rigid hand part or flex sensors placed on joints. The latter have the advantage of measuring absolute values which, however, are only scalar and need further interpretation involving anatomical knowledge, e.g. that a sensor value of .7 corresponds to joint flexion of 45°. An inertial measurement unit (IMU), on the other hand, usually measures instantaneous rates of change, like accelerations, but can be designed to return 3D rotations and positions by accumulating these measurements^[imu].

The matter of actuation belongs to the broader field of haptic feedback. According to Wang et al.^[WGL⁺19], the visual and auditory feedback of VR systems has become fairly realistic over time. However, haptic feedback is still rather poor even though it is "[...] indispensable for enhancing immersion, interaction and imagination of VR systems. Interaction can be enhanced by haptic feedback as users can directly manipulate virtual objects, and obtain immediate haptic feedback. Immersion of the VR system can be enhanced in terms of providing more realistic sensation to mimic the physical interaction process. Imagination of users can be inspired when haptics can provide more cues for user[s] to mentally construct

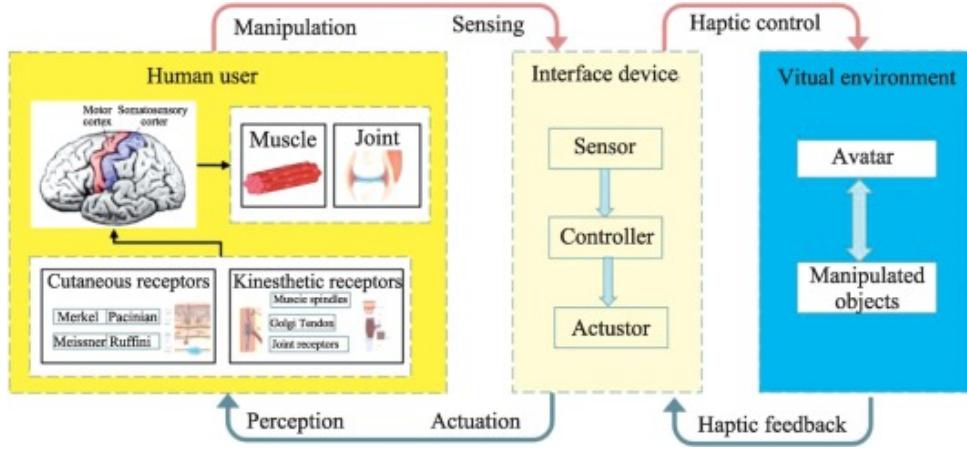


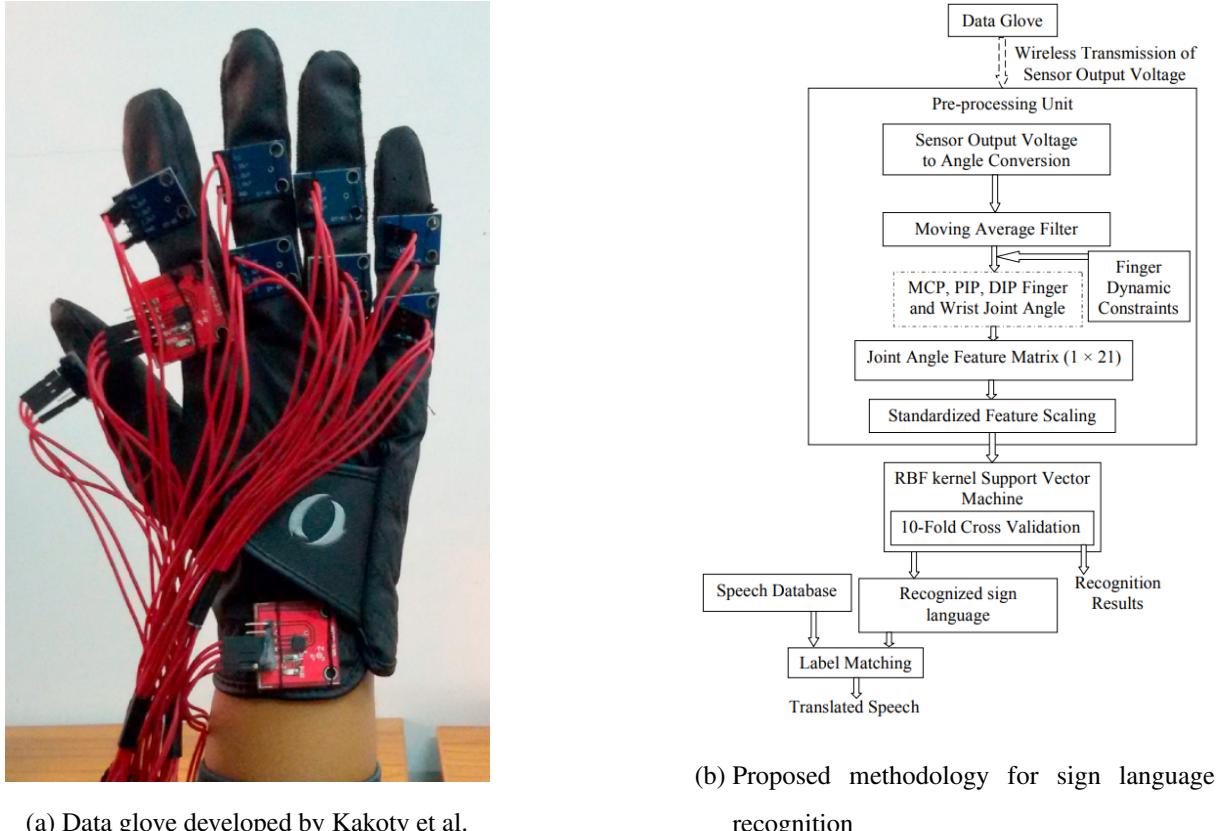
Figure 1.1: Schematic paradigm of haptic human computer interfaces consisting of three components, i.e. human user, interface device, and virtual environment; from [WGL⁺19]

an imagined virtual world beyond spatial and / or temporal limitations."

A schematic of the general paradigm of haptic human computer interfaces is shown in figure 1.1. It consists of three main components:

- the user, who perceives virtual objects through biological receptors, processes them in the somatosensory and motor cortex, and requests manipulation by mechanical movement,
- the interface device which causes perception via actuation and collects information about requested manipulation with its sensors,
- the virtual environment which evaluates the interface device's data, realizes the manipulation of virtual objects with possible constraints, and motivates actuation in the interface device via haptic feedback.

As the following examples will demonstrate, not all data gloves are designed with the capability of both sensing and actuation. Some are only used to evaluate the user's hand pose, and it is conceivable to deal only with the issue of actuation and utilize an external device for sensing, such as Leap Motion. However, actuation on its own does not seem useful for VR applications, as it needs to be variable based on the current hand pose for which tracking of the user's hand is required.



(a) Data glove developed by Kakoty et al.

(b) Proposed methodology for sign language recognition

Figure 1.2: A system for sign language recognition; from [KS18]

1.2 Examples and Applications of Data Gloves

1.2.1 Sign Language Recognition

In 2018, Kakoty et al. developed a data glove and processing system for the recognition of sign language alphabets (Indian, American and numbers) and their translation into speech^[KS18]. In their paper, several earlier projects with similar purposes are reviewed, most of which are vision based. The authors point out that glove based systems are more useful because they "[...] allow one with the freedom for locomotion during the recognition process and [are] independent of the environmental lighting conditions."

The data glove used for this project is shown in figure 1.2a. It was constructed from a leather glove by equipping it with ten angle sensors, a microcontroller, and a transceiver module. The original paper describes this design in detail.

A preprocessing unit as in figure 1.2b handles the conversion from sensor output voltage to denoised, constrained standardized feature vectors. These are then processed by a radial base function kernel support vector machine and finally, the recognized sign language is translated into speech using label matching with a pre-recorded speech database. The whole process is implemented to run in real-time

with the sensor data acquisition.

With an average recognition rate of 96.7%, this approach achieves similar results as previous projects while recognizing more alphabets. Kakoty et al. further conclude that their approach "[...] overcomes the limitations of vision based systems. [...] An extension of the presented work for sentence-based sign language recognition will lead to an enterprising device; which is part of on-going research."

1.2.2 Daily Activity Recognition

Alzheimer's disease is a common cause for the loss of cognitive abilities amongst the elderly. Well known symptoms include language problems, loss of memory and degrading functioning of the hand. To better understand the latter, Maitre et al. investigated a method to recognize daily activities using a data glove developed specifically for that project^[MRB+19].

Several reasons are named as motivation to develop the device from scratch:

- Prices for commercial data gloves are situated between ca. 600 USD and several thousand USD.
The authors propose a design that can be recreated for only about 220 USD.
- Commercial gloves are generally only compatible with Windows, but the SDK is not suitable for many other sensors.
- The new design can be specialized for recognizing the required types of grasp.

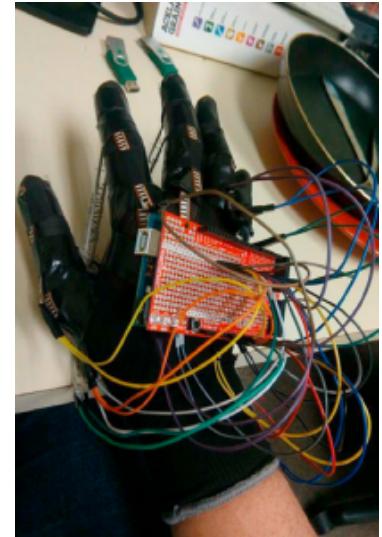
Discerning a daily activity is based on recognizing the involved item, e.g. a plate for the task *Hold a plate*. For this, the presence of an object needs to be detected and information about its shape is required. Figure 1.3a shows the resulting sensor locations, where red and yellow mark flex sensors and blue marks force sensors. A prototype device can be seen in figure 1.3b.

A group of nine participants was asked to perform the daily activities listed in the evaluation tables in 1.3c. The acquired data sets were used to train (70%) and test (30%) a k Nearest Neighbours (k-NN) model whose precision (P), recall (R) and f-score (F) can be found in the left table, and a random forest (RF) to which the right table refers.

Maitre et al. conclude that the performance of their approach is "[...] almost perfect [...]. This result is surprising and encouraging [...]." Another experiment shows that both models perform significantly worse on data collected from a person that was not recorded during training. The authors therefore suggest to train the models with data from a patient that is supposed to use the system. The device can then even be worn in the comfort of the patient's home to monitor their hand activity.



(a) Arrangement of sensors on the hand



(b) Prototype device

Table 1. Object recognition with k-NN, 70-30.

k-NN, $k = 5$, Euclidean distance, 70-30				
	P	R	F	Support
Global Accuracy	0.94			
Global Cohen's Kappa	0.94			
Hold a plate	0.9960	0.9880	0.9920	251
Hold a cup	0.9246	0.9510	0.9376	245
Hold a spoon	0.9427	0.9030	0.9224	237
Hold a knife	0.9098	0.8657	0.8872	268
Hold a fork	0.8992	0.9612	0.9292	232
Hold a pot	0.9835	0.9715	0.9775	246
Hold a pall	0.9395	0.9902	0.9642	204
Empty hand	0.9600	0.9375	0.9486	256

Table 2. Object recognition with RF, 70-30.

RF, $T = 100$, $N = \sqrt{30}$, 70-30				
	P	R	F	Support
Global Accuracy	0.99			
Global Cohen's Kappa	0.99			
Hold a plate	1.0000	1.0000	1.0000	251
Hold a cup	0.9796	0.9796	0.9796	245
Hold a spoon	0.9957	0.9873	0.9915	237
Hold a knife	0.9703	0.9739	0.9721	268
Hold a fork	0.9744	0.9828	0.9785	232
Hold a pot	0.9959	0.9959	0.9959	246
Hold a pall	1.0000	1.0000	1.0000	204
Empty hand	1.0000	0.9961	0.9980	256

(c) Evaluation of the recognition results

Figure 1.3: A data glove for recognition of basic daily activities; from [MRB⁺19]

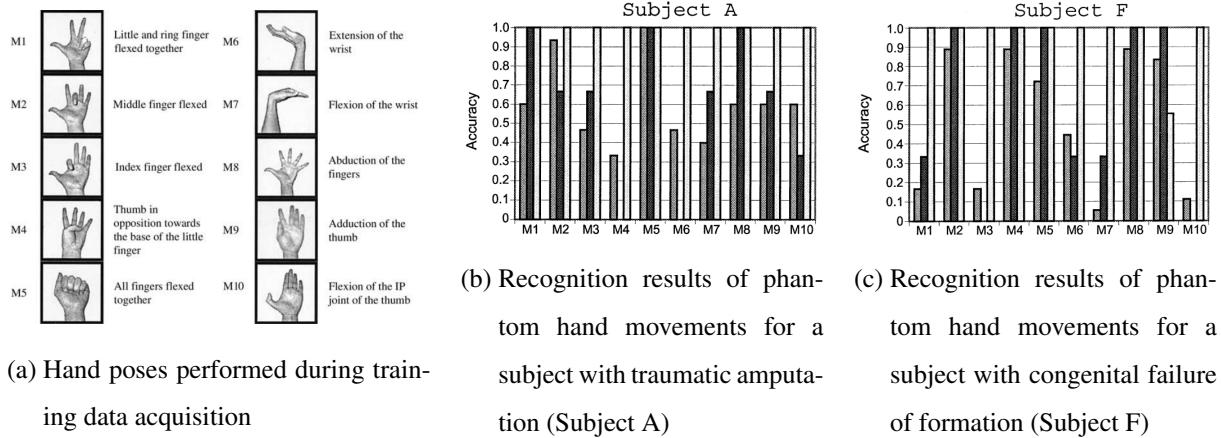


Figure 1.4: Hand poses used in [SRL05] and recognition results; in (b) and (c), grey bars depict average performance, black depicts best-session performance and white the reference glove performance; from [SRL05]

1.2.3 Refined Myoelectric Control in Below-Elbow Amputees

A group of PhD students in Sweden trained an artificial neural network (ANN) to recognize myoelectric activity of below-elbow amputees as certain hand poses^[SRL05]. To generate training data, five subjects with a traumatic amputation and one subject with a congenital failure of formation were asked to perform the movements shown in figure 1.4a with their healthy hand and their phantom hand simultaneously.

The healthy hand was therewith tracked with a data glove to match the performed movement to one of M1 - M10. The arm stump on the side of the amputation was covered with relatively many electrodes to record detailed information about the remaining myoelectric activity. The (pre-processed) data was then used to train a tree-structured ANN.

As shown in figure 1.4 and more diagrams from the original paper, certain poses were recognized especially well or badly for certain subjects. An interesting process to see was how the performance of the subjects became more accurate over the course of the experiment. As this revealed itself within minutes of a session, the authors theorized, it might be due to unmasking of existing brain structures instead of the formation of new ones.

The subject with congenital failure was of special interest because it showed that some movements could be accurately induced in the phantom hand even though they had never been executed before, possibly indicating the awakening of sleeping motor cortical areas in the brain that had been present since birth. Among the subjects with traumatic amputations, the accuracy and training effect did not seem related to the time since the amputation.

An important aspect for the training effect might be the simultaneous execution of movements with both

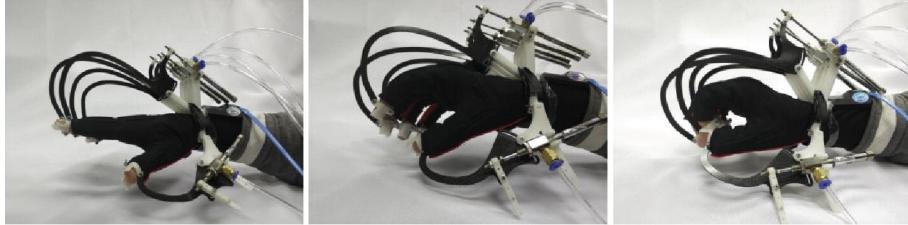


Figure 1.5: Physical prototype of a lightweight force-feedback glove with large workspace; from [ZWW⁺18]

the healthy and the phantom hand, as suggested in the original paper. The authors moreover suppose that a subject's performance might increase with visual feedback of a virtual phantom hand that was not given during these experiments. This thesis might be useful for that purpose, as it defines a virtual hand model for the use with data gloves, based on a well established hand model, in section 2.2. Work in this direction was continued by the authors of the original paper^[SEBL06].

1.2.4 A Lightweight Force-Feedback Glove

In terms of haptic feedback, a recent project is the *Design of a Lightweight Force-Feedback Glove with a Large Workspace*^[ZWW⁺18]. From their review of existing technology, the authors of the respective paper conclude the following "[...] requirements of ideal force feedback for VR interaction":

- The device weight should not exceed 100g. This value is based on the success of desktop force-feedback devices, e.g. the PHANTOM Desktop^[pha]. The prototype shown in figure 1.5 has a final mass of 245g which was achieved by a pneumatically actuated design with an outsourced construction for air compression and regulation. Force can be exerted onto the fingers by releasing compressed air into the pneumatic pistons (silver) which then pull back the links (black). Special materials were chosen for weight reduction: the links are made from 2mm thick carbon. Lightweight plastic was used for the base parts and finger caps (white).
- Typical grasping motions should be possible to simulate realistically. The device should therefore have a large motion and force range. The form of the links is specifically designed to ensure the necessary freedom of motion and makes it possible to mount the construction on the dorsal side of the hand. The authors emphasize that the key to obtaining optimal normal force at the finger cap while ensuring free movement of the finger joints is the curved profile of the sliding slot connecting the base components to the links. Its design and the respective calculations are described in detail in the original paper.
- "[...] the gloves should be easy to put on and remove, and should be adaptable for users with

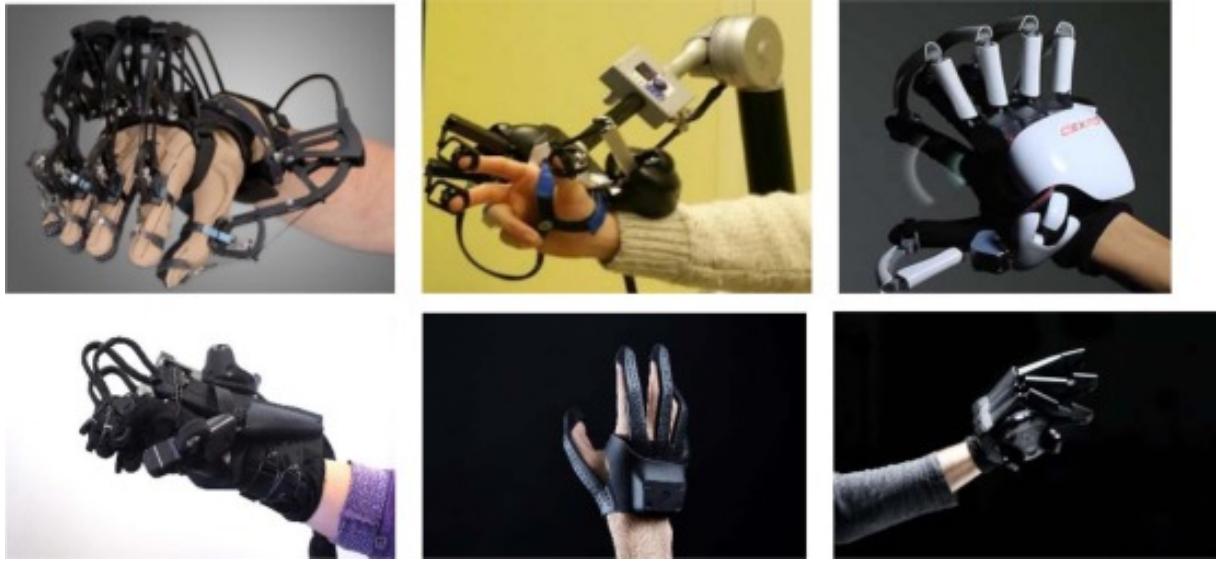


Figure 1.6: Examples of commercial data gloves as named in table 1.1 (fltr, top-row first); from [WGL⁺19]

different hand sizes." This could, however, not be achieved and is suggested for future work.

A commercial data glove was used to track the user's hand. The pose was then evaluated by a PC that coordinated the release of air from the compression system into the pistons via a microcontroller.

Experiments showed that the prototype developed by the Zheng et al. can exert forces between $0.1N$ during free motion of the finger joints and $4N$ for constrained space simulation. It does indeed have a large working space in which a minimal virtual sphere with a diameter of $30mm$ can be simulated.

However, the normal force at the finger cap could not be controlled accurately. According to the authors, a possible solution might be the use of a force sensor at the finger tip to design a closed-loop control system. Moreover, the force-feedback showed significant delay due to the time of airflow between compression system and pistons.

For future work, the original paper suggests the development of new control hardware and software. They see the possibility of integrating the glove with a head mounted device (HMD) for novel applications such as e-shopping and plan to add components for haptic feedback on the hand palm as well.

1.2.5 Commercial Products

Numerous types of data gloves have been available on the market over the past years. Some examples capable of haptic feedback are shown in figure 1.6 and a summary of their technological properties is compiled in table 1.1. Another example is the Avatar VR glove which will be used for the purposes of this thesis and explained in more detail in chapter 2.

Device name	Company	Motion tracking DoF	Force feed-back	Tactile feedback	Actuation principle	Sensing principle	Typical features
CyberGrasp	Immersion	—	One actuator per finger	—	Electric motor	22-sensor Cyber-Glove device	Feel the size and shape
H-glove	Haption	Each finger possesses 3 DoF	Force feed-back on 3 fingers	—	Electric motor	—	Can be attached to a Virtuose 6D
Dexmo	DextaRobotics	Tracking 11 DoF for hand	Force feed-back on 5 fingers	Three linear resonant actuators	Electric motor	Rotary sensors	Feel the shape, size and stiffness
Haptx glove	Haptx	Tracking 6 DoF per digit	Light-weight exoskeleton applies up to 4lbs per finger	130 stimuli points	Microfluidic array	Magnetic tracking	Feel shape, texture, motion in sub-mm precision
Plexus	Plexus	21 DoF per hand	—	One tactile actuator per finger	—	Using tracing adapters for other technology	Track with .01 deg precision
VRgluv	VRgluv	12 DoF for the fingers on each hand	Apply up to 5lbs of varying force per finger	—	DC motors	5 sensors per finger	Simulate stiffness, shape, and mechanical features

Table 1.1: Existing commercial haptic feedback gloves; adapted, from [WGL⁺19]

2 Development

This chapter names the technology used for this project, defines a skeletal hand model for the Avatar VR haptic glove, and describes the development of a natural VR user interface based on these concepts.

2.1 Devices and Software

For the purpose of thesis, I used a pair of haptic gloves called Avatar VR which is a registered trademark of NeuroDigital Technologies, S. L.^[nd], referred to as ND hereafter..

To use the Avatar VR on a computer, one needs to download, install and run the ND Suite. This program provides a background service and GUI for managing and accessing the devices. A connection can be established via Bluetooth after which all sensor data is graphically displayed in the GUI.

ND also supplies a developer's API in the form of three files (`NDAPI.h`, `NDAPI_x64.lib` and `NDAPI_x64.dll` or `x86` respectively) which need to be included in the project in a suitable manner. An instance of the `NDAPI.h` can be used to access the data of devices connected to ND Suite.

Additionally, a Vive tracker was used to determine the user's hand position and orientation. The Avatar VR can be mechanically connected to the tracker using a coupling that needs to be acquired separately. The devices named so far and the way of coupling them can be seen in figure 2.1.

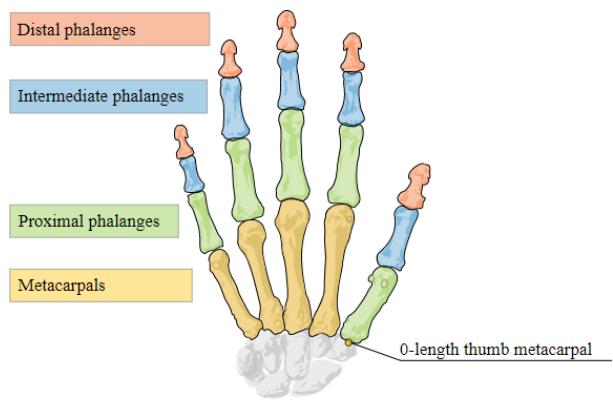
The foundation for the graphics software I developed is implemented in the `cgv` framework provided by the chair of computer graphics and visualization at TU Dresden. A user can view the plugin on a computer screen with the `cgv_viewer` or on a Vive head mounted device (HMD) simply by connecting one to the computer. The framework automatically interacts with the SteamVR and Lighthouse^[lig] technology required to determine position and orientation of the Vive trackers and HMD.



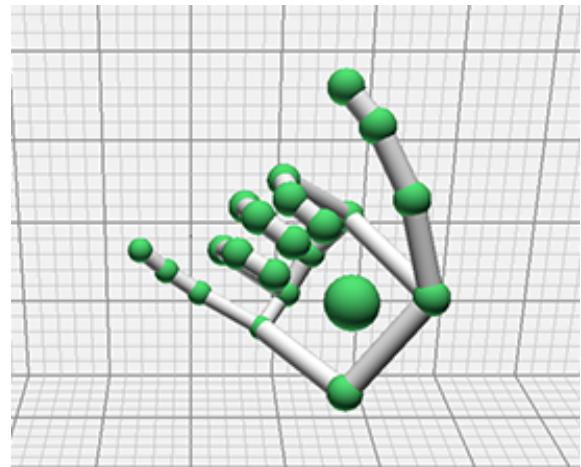
(a) Uncoupled devices, colors mark coupling points

(b) Coupled devices on user hand

Figure 2.1: Devices used to track the user's hand: Vive tracker, coupling, Avatar VR



(a) General bone structure of the human hand



(b) Screenshot of an example hand representation in the Leap Motion API version 2.0, cropped

Figure 2.2: A model of the human hand; from [lea]

2.2 Skeletal Hand Model for the Avatar VR Haptic Glove

2.2.1 Abstraction of the Human Hand Skeleton

The human hand can be roughly divided into the parts shown in figure 2.2a, i.e. the *metacarpals*, *proximal phalanges*, *intermedial phalanges*, and *distal phalanges*. A joint connecting a metacarpal to a proximal phalanx is called *metacarpophalangeal joint*, and followed by a *proximal interphalangeal joint* and a *distal interphalangeal joint*.

Even though the carpal bones (grey in figure 2.2a) are physiologically quite relevant for hand movement, they stay relatively fixed compared to the other bone sets, and can thus be omitted in our simplified model. In anatomical nomenclature, the thumb is composed of proximal phalanx and distal phalanx only and follows a metacarpal. However, this model assumes a missing metacarpal and existing intermedial phalanx instead which will be modelled by a 0-length metacarpal for uniformity reasons.

The Leap Motion API version 2.0^[leal] uses the abstraction described above and adds

- an "end" joint per finger (at the end of the distal phalanx)
- a joint diagonally across the palm from the metacarpophalangeal joint of the index
- a joint in the middle of the palm

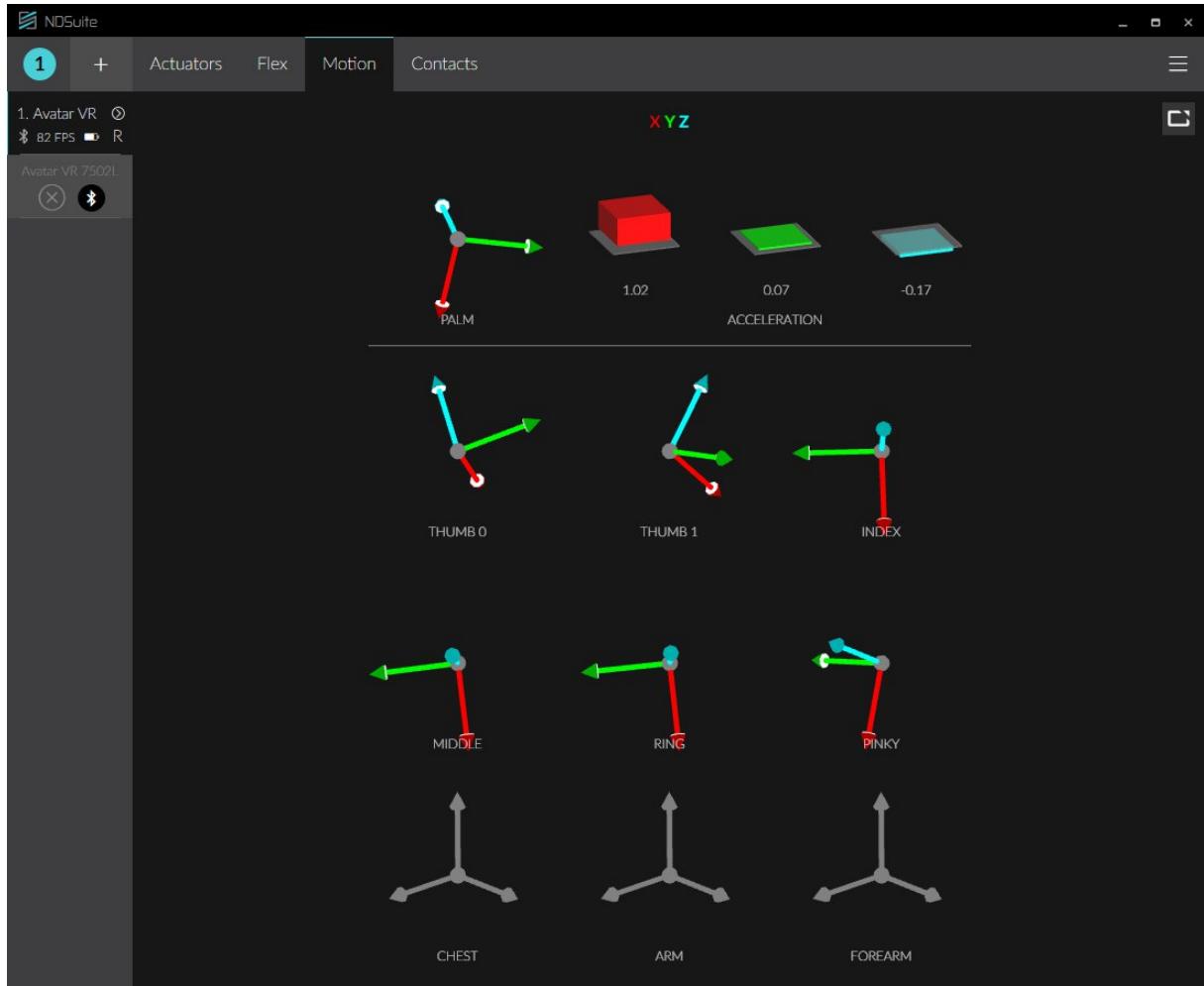
The most common representation includes cylinders for the bones and spheres for the joints. It is shown in figure 2.2b.

2.2.2 Technology and Function of the Avatar VR

The Avatar VR is equipped with three kinds of sensors:

- An *inertial measurement unit (IMU)* is located on the intermedial phalanx of each finger to measure its 3D rotation. The thumb is even equipped with two IMUs (one per phalanx) and the palm with an IMU that can measure both 3D acceleration and 3D rotation. The "Motion" tab of ND Suite shown in figure 2.3a depicts the resulting orientations as orthonormal bases.
- At the locations marked in blue in figure 2.3b, the fabric is made of an electrically conductive material. These parts are used as contact sensors (short: contacts) to determine which of the marked locations are joined.
- The thumb has a flex sensor, that will not be relevant for this thesis.

Furthermore, there are actuators at the locations marked in blue in figure 2.3c. These can be used to generate vibrations of variable intensity as haptic feedback in a way that "[...] the brain perceives it as



(a) "Motion" tab of ND Suite with orientations in an (optimistic) example hand pose



(b) Positions of contact sensors



(c) Positions of actuators

Figure 2.3: Sensors and actuators of the Avatar VR;
screenshots of ND Suite (b and c cropped)

«struct» joint_positions
<pre>// positions of hand joints in model space // format: hand_part<joint<position>> + positions: vector<vector<vec3>> // positions linearized in hand_part major format // set by make_array() + linearized: vector<vec3> // correspondence of indices in linearized // to double indices in positions + lin_to_anat: map<int, pair<int, int>> // rotate complete part + rotate(int part, quat rotation): void // translate complete part along neg. z-axis + translate_neg_z(int part, float z): void // translate all positions // used to move hand from construction origin // to model view location + translate(vec3 translation): void // scale all positions // used to realize hand size at construction origin + scale(float scale): void // generate linearized from positions + make_array(): vector<vec3></pre>

Figure 2.4: Struct for managing positions of hand joints, only most important parts included

'Real Touch' input", according to the producer.

2.2.3 Using Data From the Avatar VR to Visualize the User's Hand

The same model used by the Leap Motion API version 2.0 will also be used here. Leap captures absolute joint positions and optimizes them for visualization. In contrast, my approach for the Avatar VR is to use a fixed resting state geometry of the user's hand (bone lengths and resting state joint positions). At the time of a `draw()` command, the geometry is adjusted according to the data given by the glove and the Vive tracker. This is done with the help of the struct shown in figure 2.4 (only most important parts included).

First, the palm joints' positions are rotated according to the orientation given by the Vive tracker and saved in `positions`. The construction of each finger begins at the end joint of the distal phalanx which is initialized in `positions` as $(0, 0, 0)$, translated along the negative z -axis by the hard-coded length of the distal phalanx and rotated by the respective quaternion (see below). The other phalanges are constructed in the same manner, before the whole finger is translated to its metacarpophalangeal joint. Finally, the whole hand is scaled to model view space and translated to the model view position of the Vive tracker.

Because the Avatar VR only returns one quaternion per finger (except for the thumb), I had to think of a way to distribute the orientation along all three phalanges: First, the Euler angles are calculated from the

quaternion^[qua]

$$\begin{aligned}\alpha &= \arctan \frac{2(q_0q_1 + q_2q_3)}{1 - 2(q_1^2 + q_2^2)} \\ \beta &= \arcsin(2(q_0q_2 - q_3q_1)) \\ \gamma &= \arctan \frac{2(q_0q_3 + q_1q_2)}{1 - 2(q_2^2 + q_3^2)}\end{aligned}$$

where α is the roll angle, β the pitch angle and γ the yaw angle, and the quaternion can be written as (q_0, q_1, q_2, q_3) . For implementation, one has to use the `atan2` function, because `arctan` only returns values between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$.

The `NDAPI` quaternions live in a space where the x -axis points to the right, the y -axis points upwards, and the z -axis points into the picture. In terms of geometry, the coordinates are negatively oriented. One might know this as the "left hand rule".

However, `vr_ctrl_panel` is oriented with the negative z -axis pointing out of the picture (positively oriented, "right hand rule"). This results in opposite senses of rotation around the x - and y -axes in the two coordinate systems, so q_1 and q_2 from `NDAPI` need to be negated before use with `cgv`. Then roll corresponds to bending the finger (flexion / extension), pitch to a left-right-motion (abduction / adduction) and yaw to a twisting motion humans can hardly perform with their fingers (pronation / supination).

Since the interphalangeal joints act as revolute joints, only a roll component can be applied to them. As shown in the code below, the roll angle is distributed according to three floats (`rot_split`). After some experimenting, (.5, .5, .25) is an option for these numbers that gives a good approximation of most actual hand poses. It corresponds to phalanx rotation during grabbing. Yaw and pitch are completely executed in the metacarpophalangeal joint.

```
vec3 x(1, 0, 0), y(0, 1, 0), z(0, 0, 1);
recursive_rotations[finger][PROXIMAL] = palm_rotation
    * quat(z, yaw) * quat(y, pitch) * quat(x, rot_split.x() * roll);
recursive_rotations[finger][INTERMED] = quat(x, rot_split.y() * roll);
recursive_rotations[finger][DISTAL] = quat(x, min(1.4f, rot_split.z() * roll));
```

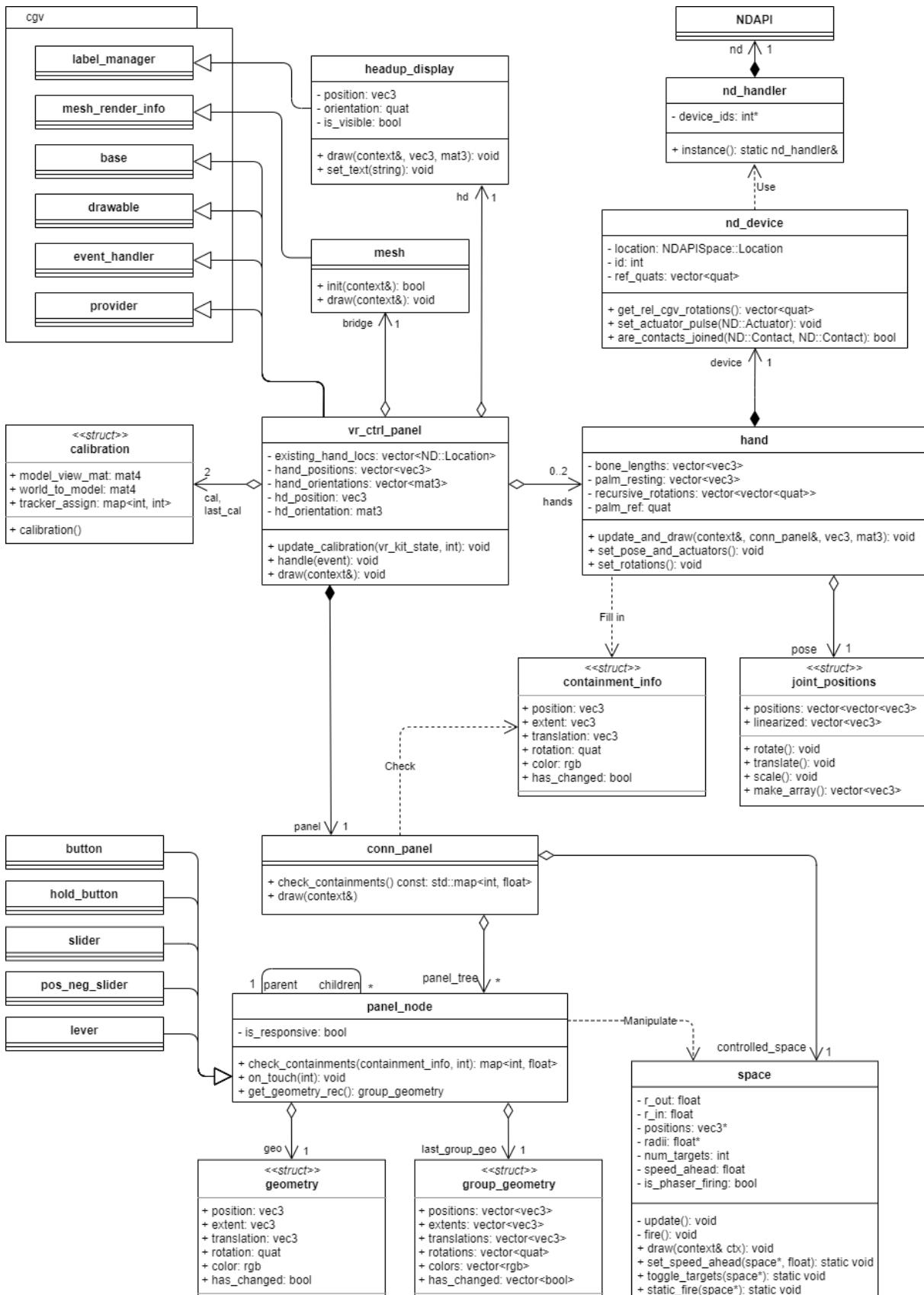
Figure 2.5 shows correspondences of user hand positions with virtual hand positions. A good approximation is reached, however, the resulting hand pose is not unique. For example, figure ?? shows two user hand positions that lead to approximately the same outcome as the virtual hand pose shown in the middle line of figure 2.5. This issue cannot be fixed because the Avatar VR does not provide more information about the finger poses.



Figure 2.5: Example pose correspondences of user and virtual hand; one pose per line; columns f.l.t.r.: user hand radial, virtual hand radial, user hand dorsal, virtual hand dorsal



Figure 2.6: Radial view of two user hand poses corresponding to a similar handpose as the one in the middle line of figure 2.5

Figure 2.7: Class diagram of `vr_ctrl_panel`, only most relevant elements shown

2.3 Project Idea and Structure

To give the VR user interface some context rather than just moving boxes, I decided to design it as a conn panel on the USS Voyager (Star Trek). The term *conn* is short for *control and navigation*, meaning the panel is used to steer the ship. The classes used to implement this and their interaction are shown in figure 2.7.

2.3.1 Main class

The main class and entry point carries the same name as the plugin itself. It is derived from the `cgv` classes `base`, `drawable`, `event_handler` and `provider`, and is used for scene management and interaction with the framework functionalities.

Its capability to handle `vr_pose_events` is used to catch tracker poses and HMD poses (position and orientation) which are forwarded to the `hands` (see section 2.2) and `head_up_display` (see part 2.3.6) respectively during the `draw()` call.

It distributes `draw()` commands among all relevant members (`hand`, `conn_panel`, `mesh`, and `headup_display`). In the process, it passes on the necessary arguments which ensures laziness: recalculations of geometry and containment checks are only done when they are necessary.

This class further manages the calibration routine (see section 2.4). It is the only class registered via object registration.

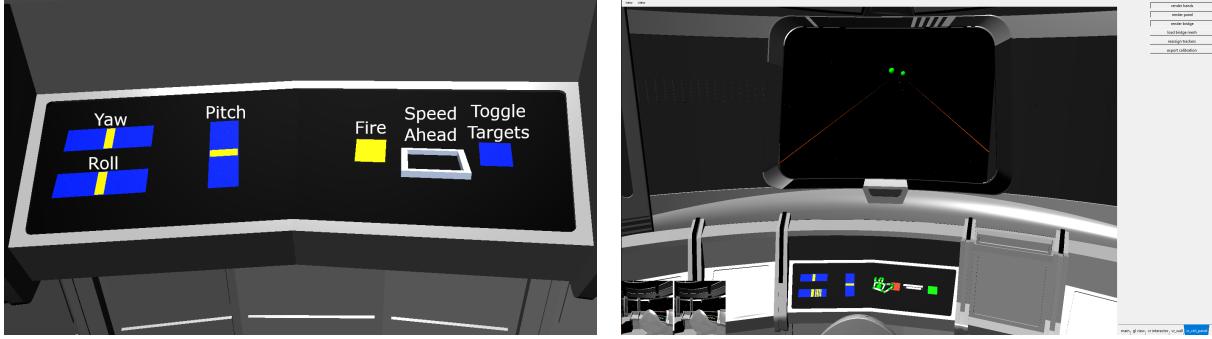
2.3.2 Bridge

The `mesh` class renders the Voyager bridge. It uses the basic functionality of `mesh_render_info` provided in `cgv` to load and draw. A triangular mesh was provided by Chainsaw_NL and Star trek Meshes (see Copyright). The faces belonging to the viewscreen were deleted to enable a view to the outside.

2.3.3 Space, Targets and Phasers

Around the bridge, space is simulated by a spherical shell filled with stars implemented in `space`. The user never actually changes position in model space. Instead, stars are moved in the opposite direction of the simulated motion. The speed ahead can be set via the following method:

```
static void set_speed_ahead(space* s, float val)
{
    s->speed_ahead = val * s->max_speed_ahead;
```



(a) Panel configuration with explanatory annotations
(added manually, not visible in `vr_ctrl_panel`)

(b) Screenshot of `cgv_viewer` showing bridge with conn panel and view screen while firing on targets

Figure 2.8: Setup of conn panel and bridge

}

The method needs to be static so it is possible to set it as callback method in objects outside this class. Analogous methods exist for all three rotational speeds that are named as for planes, i.e. pitch around the x -axis, yaw around the y -axis, and roll around the z -axis.

This class also manages targets whose visibility can be toggled with `toggle_targets()` and that can be eliminated with phasers (laser canons) via the `static_fire()` method.

More details and the geometry behind this class are described in section 2.5.

2.3.4 Control Panel

The actual interface is built and drawn by `conn_panel`. This passes the `controlled_space` and a callback function on to each `panel_node` during construction and the `containment_info` coming from a `hand` during `checkContainments()`.

In its `draw()` method, `conn_panel` queries the current panel geometry with `panel_tree->get_geometry_rec()` and renders the result with a `box_renderer` provided by `cgv`. The `conn_panel` is positioned right on top of the mesh's `conn`.

Each `panel_node` is represented as a box or combinations of boxes placed relative to their parent element. During an `on_touch()` call, the non-trivial derived classes (`button`, `hold_button`, `slider`, `pos_neg_slider` and `lever`) use their callback function to manipulate the `controlled_space`.

The `conn_panel` configuration used in `vr_ctrl_panel` is shown in figure 2.8a. The pitch and yaw slider behave like a control onboard a plane, e.g. sliding the pitch away from the user initiates downward rotation. Figure 2.8b shows a screenshot of the panel in action, while a user is trying to hit a target.

2.3.5 Hands

Interaction with the Avatar VR is managed by the singleton `nd_handler` which can be used to pass function calls through to an instance of `NDAPI`. This is frequently done in `nd_device`, a class for querying sensor data from the gloves and converting it for better compatibility with `cgv` and `vr_ctrl_panel`, e.g. the sign flip explained at the end of section 2.2.

Finally, the representation of the user's hands is implemented in `hand`. It keeps track of several points and joints the details of which are described in section 2.2. During a `draw()` call, `vr_ctrl_panel` passes a pointer to its `conn_panel` to each `hand` which then passes on information about the current hand geometry wrapped as `containment_info` to the `conn_panel` for containment check. Each `panel_node` containing some hand part can then react with its `on_touch()` method. A `map<int, float>` describing the contained positions and suggested vibration strengths is returned to the hand, enabling it to react as well.

The vibration strengths are calculated as an interpolation between a minimum and a maximum value:

```
float strength = min_strength + sqrt(1 - dist / rad) * (max_strength - min_strength);
strength = max(0, strength);
```

where `dist` is the distance between the joint and the panel and `rad` is the radius of the sphere representing the joint. This yields a `strength` of `min_strength` when the sphere is not touching the panel and increases to `max_strength` like a square root with the sphere moving into the panel. This kind of interpolation is motivated by the perception of physical objects which also increases quickly when first touching an object and then less quickly when increasing the pressure.

2.3.6 Headup Display

To convey written information to the user, a rectangular white region showing black text can be displayed in front of their eyes. This is implemented in `headup_display` which uses `label_manager` provided by `cgv`. Its visibility can be controlled by setting an empty text (invisible) or a non-empty text (visible).

Whenever the display is invisible, its `draw(context& ctx, vec3 p, mat3 ori)` call simply returns. Otherwise, its position and orientation are updated so it is rendered .6 units in front of and .1 units above the user's eye position:

```
vec3 vs_hmd = ori * vec3(.0f, .1f, -.6f);
position = pos + vs_hmd;
orientation = ori;
```

2.4 Calibration

To make the use of `vr_ctrl_panel` in different rooms and setups possible, a calibration routine is required. It follows the state diagram shown in figure 2.9. Generally, joined contacts are interpreted in the following way:

- thumb + index: acknowledge
- thumb + middle: decline
- palm + index: choice 1
- palm + index: choice 2

The routine can be activated by joining index and thumb of one hand for three seconds. The hand requesting calibration is saved. The calibration routine does not react to the contacts of the other hand. This is done for reasons of implementation and because users found it natural to control the calibration with a single hand. The current calibration is saved so the user can return to it after aborting.

While in the "requested" stage, the headup display shows a countdown until calibration. After that, none of the objects in the scene are rendered anymore. The user is asked whether their position should be taken from the HMD or set manually. In case of the latter, one can tap thumb and index above their head to set the position from the current tracker pose.

After that, the hands are rendered again and a countdown of 3s is displayed. The user is instructed to hold their hands straight with the fingers pointing forward. During this countdown, the hands and the model view matrix are constantly recalibrated as the following pseudo code shows:

```

vec3 panel_origin = tracker_pos_average;
vec3 new_z = user_pos - panel_origin;
new_z.normalize();
float angle_y = angle between new_z and (0, 0, 1);

// panel_origin becomes coordinate origin
mat4 model_view_mat = translation(panel_origin)
// -new_z becomes view direction
    * rotation((0, 1, 0), angle_y)
// bridge center becomes coordinate origin
    * translation(panel_pos_on_bridge);
mat4 world_to_model = inv(model_view_mat);

vector<quat> ref_quats = nd.get_quats_from_sensors();
for (size_t i = 0; i < num_imus; i++)
{
    ref_quats[i] = ref_quats[i].inverse();
}

```

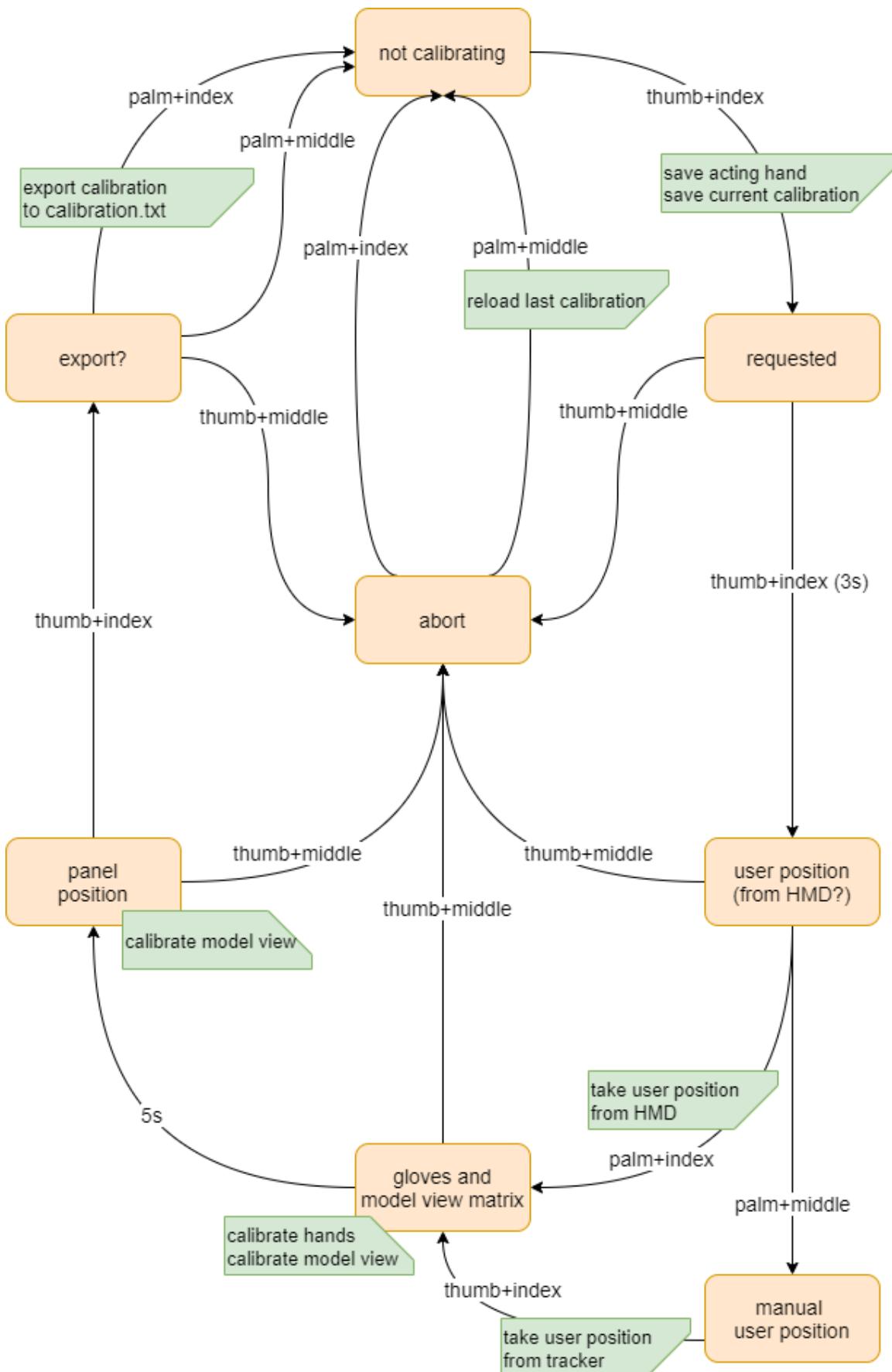


Figure 2.9: State diagram of the calibration routine

Now, we have the following equalities:

```
new_z == model_view_mat * (0, 0, 1, 1)
(0, 0, 1, 1) == world_to_model * new_z

unit_quat == ref_quats[i] * nd.get_quats_from_sensors()[i]    // for all i
```

The tracker positions are now always transformed to model space by multiplication with `world_to_model` and the quaternions from the sensors are multiplied with the `ref_quats` before use. Therefore, the current calibration pose is assumed as the new "unit", meaning it looks like a trivial pose.

Finally, the conn panel reappears close to the hands. One can continue to adjust the panel position by moving their hands until satisfied. A final acknowledgement by tapping index and thumb ends the calibration. It can then be exported to a file which is automatically loaded at plugin start.

Between two calibration stages, a short feedback pulse is generated by all actuators simultaneously to notify the user of the change. When the calibration is finished, a final pulse runs through the actuators successively.

At any time, the calibration can also be aborted by joining thumb and middle finger which results in three short actuator pulses. One can then choose to return to the previous calibration or keep the partially altered one.

The whole process is communicated to the user on the headup display. It appears with the initial countdown and disappears when returning to the "not calibrating" stage.

2.5 Space, Targets and Phasers

A schematic of the spherical shell used to simulate space is shown in figure 2.10. The outer radius determines the overall size of the sphere. A spherical region around the user that does not contain any stars is defined by the inner radius. This should include the complete bridge mesh to assure stars are only visible to the user on the view screen. Congruent with other parts of the plugin, the flight and view direction points along the negative z -axis.

Stars are rendered as small spheres whose radii follow a normal distribution. The shell is initialized with a constant number of stars with uniformly distributed positions. It saves the forward and angular speeds as float values. During a `draw()` call, the method `update()` calculates new positions for all stars. It can be summarized as follows:

for all stars do

Move star away from origin

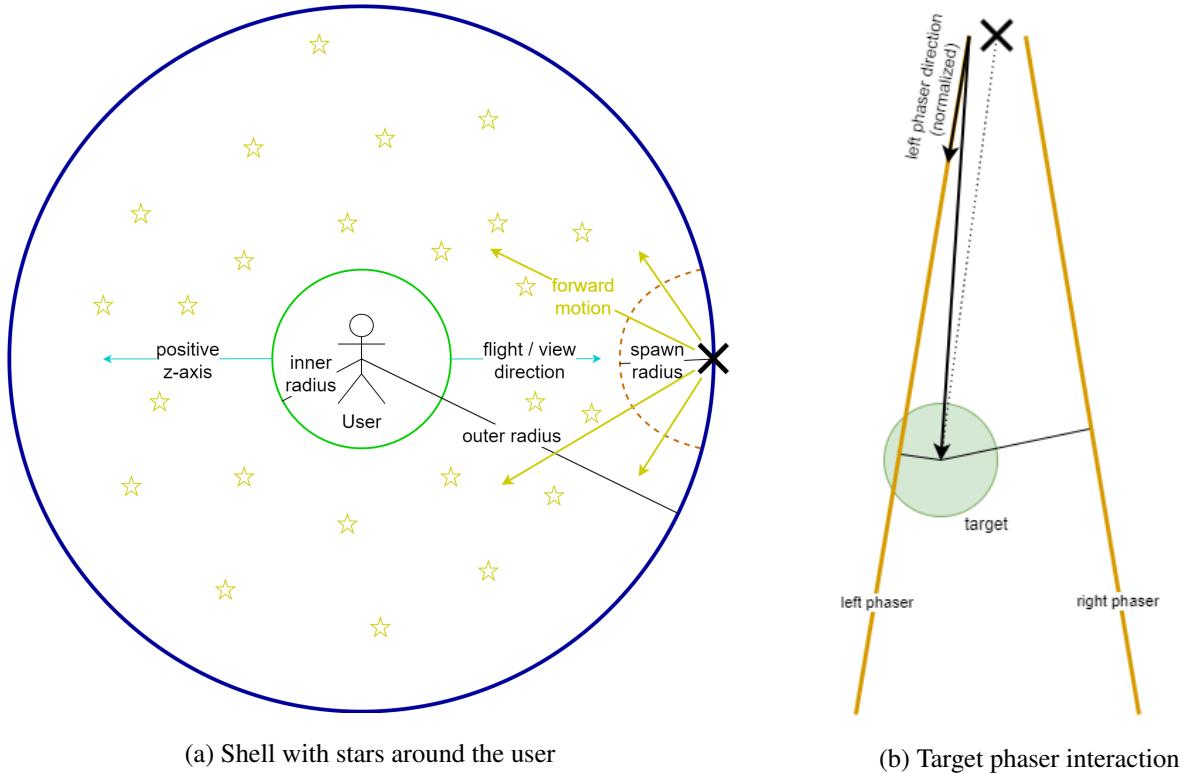


Figure 2.10: Schematics of the geometry behind the `space` class, see text for explanations

```

if star is outside of shell then
    Respawn star
else
    Rotate star around user
end if
end for

```

To somewhat simplify the math behind this calculation, I used the point on the outer hull right in front of the user as origin for the shell space, instead of the center of the shell. The origin is marked with a cross in figure 2.10. A model view matrix then translates the shell to model space.

When moving dead ahead (angular speeds are zero), stars are moved along the yellow "forward motion" arrows in figure 2.10a. The bigger the angle between their position vector and the positive *z*-axis, the slower they move. More precisely, the length of a star's position vector is increased by the product of the cosine of the angle between the position vector and the positive *z*-axis, and the general distance it has covered, calculated from `speed_ahead` and the time since the last update.

If a star moves outside the outer hull of the shell, its position is re-initialized randomly somewhere on a sphere of the spawn radius around the origin. It cannot be simply set to the origin because it needs a

direction and the spawn radius should not be too small in order to avoid the impression of stars spawning very densely. Stars moving into the free sphere around the user are mirrored at the user. As this only happens when a forward motion is applied, the star then keeps moving away behind the user.

An additional rotation is realized by simply rotating the stars around the user. After some experimenting, I realized the spawn sphere needs to be rotated inversely around the user as well to achieve a realistic experience and because otherwise, the newly spawned stars converge to a line at slow forward motion and fast rotations.

Because they behave quite similar to the stars, the targets are managed in `space` as well. Their positions are updated as described for the stars. Additionally, their radii are continuously adjusted as in the following pseudo code:

```
// value between 0 (at user_pos) and 1 (at outer shell)
float dist_frac = target_pos_rel_to_user.length() / r_out;
// value between 0 (at outer shell) and 1 (at user_pos)
dist_frac = 1 - dist_frac;
// realistic size change
dist_frac = sqrt(dist_frac);
// scale to basic target radius
target_radius_loc = dist_frac * target_radius;
```

This results in a radius of 0 at the outer shell and in a radius of `target_radius` at the user position which is never reached due to mirroring at the inner shell. The targets are not active at plugin start, meaning they are neither updated nor rendered. They can be activated using `toggle_targets()` and deactivated with the same method.

When the `fire()` method is called, phasers are rendered as cones running from the left and right of the view screen to the left and right of the origin (cross in figure 2.10) where there radius becomes 0. This gives the impression of infinite rays. All targets are checked if they have been hit as shown in the schematic in figure 2.10b (viewed from above) and the following pseudo code:

```
vec3 target_rel_to_left = target_pos - left_phaser_end;
vec3 orth_proj = dot(target_rel_to_left, left_phaser_dir) * left_phaser_dir;
float dist = (target_rel_to_left - orth_proj).length();
bool hit = dist < target_radius_loc;
```

In figure 2.10b, the dotted line corresponds to `target_pos`, the long arrow to `target_rel_to_left` and the short arrow to the normalized vector `left_phaser_dir`.

If a target has been hit, it is mirrored at the user position removing it from the view screen.

3 Evaluation

3.1 Participating Subjects and Preparation

For the purpose of a user study, I chose two Star Trek Fans (A: female, 21 years; B: male, 48 years; called *Trekkies*) and two people not familiar with the franchise (C: female, 49 years; D: female, 11 years; called *Non-Trekkies*). All of them were instructed how to wear the Avatar VR and the HMD, and had approximately 15min to familiarize themselves with the environment, i.e. the conn panel, wearing the HMD, moving through space and eliminating targets.

3.2 Qualitative Evaluations and Suggestions

The subjects were then asked for their opinion and suggestions regarding the design and behavior of the plugin which are summarized below.

3.2.1 Usage of the Avatar VR

The Avatar VR has an inner and an outer layer between which the electrical components are situated. The inner layer is made from a stretch fabric, probably to adjust to different hand sizes. However, the fabric does not slide onto the hand well and the layers are only sparsely connected, allowing them to shift against each other. This made it difficult for the subjects to put on the glove for the first time. It took about 30s, not counting the time needed to correctly place the actuators on the fingers. With some practice, the whole process can be completed in about 10s. Putting on a second glove, locking the wrist band, and activating the gloves usually required outside help.

When activating the glove, an internal calibration sets the current pose of each IMU to a trivial one as shown in figure 3.1a. The producer^[nd] actually intends for the gloves to be calibrated while laying on a flat surface. However, when putting the glove on after this and laying the respective hand on the same surface in a similar pose, the measured orientations exhibit a noticeable difference from the user's hand pose as shown in figure 3.1b.

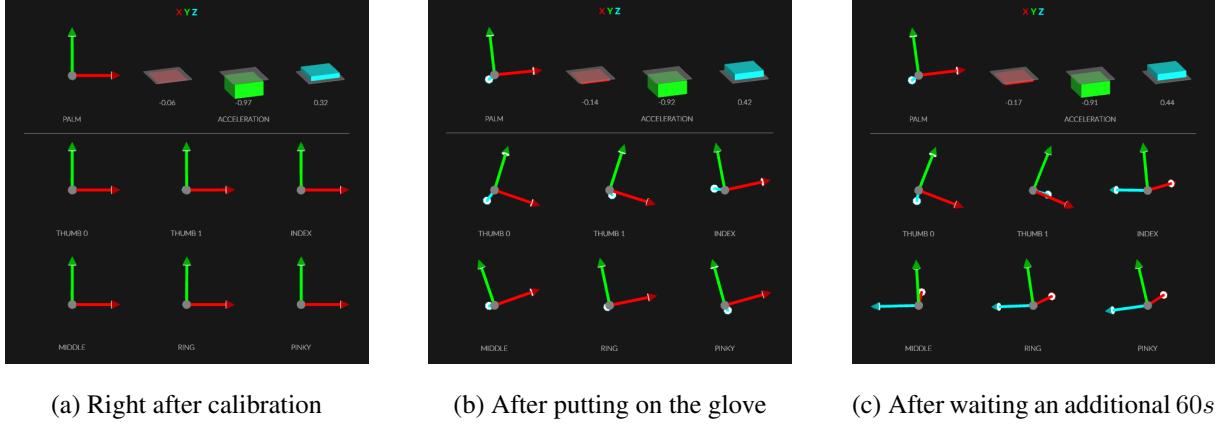


Figure 3.1: "Motion" tab of ND Suite at different times after the Avatar VR's intrinsic calibration; middle and right picture taken after calibration as suggested by the producer

Furthermore, after performing some movements with the glove, the measured orientations sometimes keep changing, even when resting the glove on a surface again. The stronger the accelerations during these movements are (e.g. during shaking), the stronger this drifting effect becomes. Figure 3.1c shows the measured orientations after resting the hand for 60s which implies that even putting on the glove carefully involves sufficient motion for drifting to occur.

In my experience, putting on the gloves first and activating them afterwards yields a more stable result, so the subjects were asked to do that. Then however, the gloves should not move during calibration time which most of the subjects found hard to achieve.

The recalibration of the glove described in section 2.4 can compensate for the absolute difference between the user's hand pose and the measured orientations, but not for the drifting effect. As it is the only way to trigger an intrinsic calibration, the gloves needed to be switched off and back on frequently.

Since the computer used during the study does not have Bluetooth capabilities, an additional USB Bluetooth connector was used to establish a connection with the gloves. However, this proved unstable for a single glove which disconnected in a matter of minutes or when moving further than 1m away from the connector, and it was almost impossible to connect both gloves at the same time. One possible explanation might be an insufficiency of the external connector, as the connection was stable for a longer time on a device with integrated Bluetooth capabilities. Also, the Bluetooth capable Vive trackers placed directly on the glove might have an interfering influence.

To avoid interruptions, the gloves were connected to the computer via Micro USB for the user study. The cables had to be fixed on the users' arms with a sweatband to relieve the physical strain on the gloves' Micro USB ports.

The subjects described the gloves as "restrictive" when touching physical objects which can be attributed especially to the actuators that are situated on the most sensitive spots of the skin. However, usage in VR was regarded "unhindered". Subject A mentioned that a grabbing motion is slightly obstructed. The glove is light and, generally, the fabric moves easily.

A single IMU per finger makes a perfect reconstruction of the user's hand pose impossible. Nonetheless, all subjects regarded the approximation described in part 2.2.3 as "natural", "well usable", or similar.

3.2.2 Haptip Feedback

As an essential part of the whole project, the haptic feedback strength (see part 2.3.5) was adjusted in discussion with the subjects. After that, all subjects regarded the feedback when touching a virtual surface as "strangely realistic" or similar.

I then applied different maximum vibration strengths for various panel components which none of the subjects found helpful for distinguishing between the components. Instead, all but subject B regarded it as "unrealistic" and "confusing", so I went back to the original approach of a single maximum vibration strength throughout the whole panel. A good value for this strength in terms of the Avatar VR seems to be .2.

When asked to reach through the conn panel, the subjects described a sense of intuitive hesitation that I could also see in their movements. Apparently, their brain had somewhat accepted the physical barrier formed by the virtual panel.

3.2.3 Timing

During all sessions, the plugin ran smoothly. None of the subjects mentioned glitches or holds.

To test the reaction time of the glove actuators, the conn panel was calibrated right onto a table edge. This was possible with the see-through function of the HMD that is meant to keep users from running into physical objects, like a wall. All but one subject (D) found the time difference between touching the table edge and haptic feedback from the gloves negligible, i.e. not sensible. During simulation, all subjects regarded the reaction time when touching a virtual object as negligible.

3.2.4 Intuitive Grabbing and Lever Reaction

Three of four subjects (except B) performed a grabbing motion intuitively as shown in figure 3.2 with the index folding onto the thumb, above the contact on the palm. When designing the lever on the conn



(a) Radial view



(b) Palmar view

Figure 3.2: Intuitive grabbing performed by three of four subjects; index folds onto thumb instead of palmar below thumb

panel, I had anticipated a more palmar grabbing and therefore required both index and middle to be joined with the palm contact for the lever to become responsive. This is meant to keep the lever from changing its angle and value when incidentally touching it and rather require an active grabbing to use it. However, it made the lever responsiveness unintuitive for these subjects.

To compensate, the condition of joined index and palm contacts was deleted. This does not significantly undermine the requirement of an intentional grabbing, as index extension combined with flexion of the other fingers is a difficult motion. The grabbing can thus still be sufficiently recognized by the middle and palm contacts.

Subjects A, C and D described the adjusted lever control as "much simpler" and "more natural". Subject B stated that he did not feel a difference, supporting the statements above.

3.2.5 Slider Reaction

In the initial implementation, the sliders on the left side of the panel had been designed to react only to touches close to their current setting. This was meant to give the impression of a slider rather than buttons. However, the participants found it hard to match the current setting which resulted in problems with changing the slider value. This was adjusted so the user can tap anywhere on the slider to immediately change its value. A field was added in the middle of the slider that can be used to set its value exactly to 0.

Subject	A	B	C	D
Session 1	1:04	1:06	2:50	1:48
Session 2	0:41	1:25	1:51	1:30
Session 3	0:27	0:37	0:52	0:58

Table 3.1: Time required to eliminate 5 targets

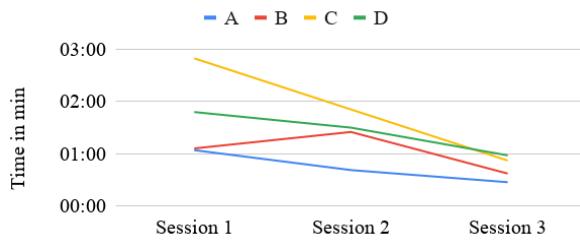


Figure 3.3: Time required to eliminate 5 targets

3.2.6 Targets and Phasers

The target color was changed from red to green increasing the contrast to the orange phasers. The phaser color, position, and direction were adjusted to match the Trekkies impression of a phaser from the movies. The size of the targets represented by `target_radius` above was adjusted until the subjects found it challenging, but possible to eliminate them. The "Toggle Targets" button was changed to stay green while the targets are active. The maximum rotation speed was adjusted until the subjects were comfortable with the resulting slider sensitivity.

3.2.7 Calibration

Instructions on the headup display during calibration lead to misunderstandings and miscalibrations. They were discussed with the users and adjusted until agreed upon.

Irritation was expressed when the panel reappeared during calibration because the user's hands ranged into the panel triggering continuous vibration feedback. I added a vector `hand_vs_panel_for_calibration` in the `calibration` struct which moves the panel below the hands during calibration to avoid this.

3.3 Quantitative Evaluation

Also quantitatively, the computation time turned out to be sufficiently small. The calculation took around $2 - 5\text{ms}$ per frame which is significantly lower than the 11.1ms ensuring 90fps . When the plugin is started, it loads the bridge mesh within less than 3s . The gloves transfer data with a framerate of ca. 80fps when connected via Bluetooth and 200fps when using their Micro USB port.

To measure the user performance, I measured the time subjects needed to eliminate 5 targets. Before each session, the gloves were intrinsically recalibrated by turning them off and back on. The targets were deactivated and the speed ahead set to maximum without rotation. The timer was started when pushing the "Toggle Targets" button. Each subject ran three such sessions.

Table 3.1 and figure 3.3 display the results of this experiment. All subjects increased their performance with each session, except subject B whose jumpy movements during session 2 caused his virtual hands to quickly diverge from the actual hand pose. This made it hard to use the controls with his fingers, and he had to finish the session using only the rigid parts of the palm to interact with the conn panel. The Trekkies performed noticeably better than the Non-Trekkies, who stated they had only little interest in and experience with VR in general.

The overall improvement of the session times despite an earlier familiarization with the plugin might be due to the change of situation and the associated time pressure. In any case, the results prove an acceptable usability of the plugin.

4 Conclusion

For this thesis, I extensively reviewed literature on the principles of data gloves, available technology and some applications. There have been numerous, quite diverse approaches to human computer interaction involving only the user's hands. Many already are or might yet become useful, not only for VR, but medical and other projects as well. A few of them are summarized in section 1.2.

Furthermore, I familiarized myself with the Avatar VR as well as tools to connect it to a computer, access its sensor data and control its actuators (ND Suite and `NDAPI`). A plugin coupling Avatar VR finger tracking and Lighthouse position and orientation tracking was developed. It uses a hand model designed on the basis of human anatomy and the model used in Leap motion API version 2.0.

The plugin scenario sends the user to the bridge of the USS Voyager where they can fly through a simulated space. Targets can be activated, deactivated, and eliminated with phasers. The geometry used for the simulation of space and for the phaser target interaction was explained thoroughly. To make the use of the plugin in different settings possible, a calibration routine was established. A user study led to numerous improvements of the plugin and revealed insights into its usability.

Some issues regarding the Avatar VR can be found in part 3.2.1. Especially the fact that it has only one IMU per finger made it challenging to recreate the user's hand pose realistically, which was possible by distributing the orientation of the intermedial phalanx onto the other phalanges in a physiologically plausible way. However, the device is most likely unfit for applications requiring more precision, e.g. the use with virtual musical instruments. In total, the glove was quite difficult to handle, already because of its instable connection to the computer and the drifting effect.

Nonetheless, I could personally gain a lot from this project. It deepened my understanding of programming in C++ and my knowledge about the `cgv` framework which has even more features than I expected. The development of `vr_ctrl_panel` was an interesting challenge, the result of which works satisfactorily well and is even fun to use.

Glossary

framework provided by the chair of computer graphics and visualization at TU Dresden; foundation for the graphics software developed in this thesis. 2, 7, 8

plugin developed in this thesis. 2, 9, 10, 14, 16

vr_ctrl_pAvatar VR haptic glove used in this thesis. 2, 6, 10, 17

distal interphalangeal joint joint connecting an intermedial phalanx to a distal phalanx. 2, 10

distal phalanx finger bone farthest from the palm of the human hand, shown in figure 2.2a. 2, 10, 11, 13, 18

head mounted device device worn in front eyes for immersive visualization. 2, 7, 19

intermedial phalanx finger bone of the human hand between proximal phalanx and distal phalanx, shown in figure 2.2a. 2, 10, 11, 18

intertial measurement unit technical device for measuring rotation or acceleration. 2, 3, 11, 19

metacarpal set of bones in the palm of the human hand, shown in figure 2.2a. 2, 10, 11, 18

metacarpophalangeal joint joint connecting a metacarpal to a proximal phalanx. 2, 10, 11, 13, 14

NeuroDigital producer and trademark owner of the Avatar VR. 2, 19

proximal interphalangeal joint joint connecting a proximal phalanx to a intermedial phalanx. 2, 10

proximal phalanx finger bone closest to the palm of the human hand, shown in figure 2.2a. 2, 10, 18

Vive VR system; tracker and headset used in this thesis. 2, 7

Acronyms

HMD head mounted device. 2, 7, 8, 16

IMU inertial measurement unit. 2, 3, 11

ND NeuroDigital. 2, 7, 11, 12

VR Virtual Reality. 2, 3

Bibliography

- [imu] Inertial measurement unit. accessed 06 Sep 2020.
- [KS18] Nayan M. Kakoty and Manalee Dev Sharma. Recognition of sign language alphabets and numbers based on hand kinematics using a data glove. *Procedia Computer Science*, 133:55 – 62, 2018. International Conference on Robotics and Smart Manufacturing (RoSMa2018).
- [lea] Introducing the skeletal tracking model. accessed 18 Aug 2020.
- [LHT⁺19] Yang LI, Jin HUANG, Feng TIAN, Hong-An WANG, and Guo-Zhong DAI. Gesture interaction in virtual reality. *Virtual Reality & Intelligent Hardware*, 1(1):84 – 112, 2019.
- [lig] Steamvr, lighthouse.
- [MRB⁺19] Julien Maitre, Clément Rendu, Kévin Bouchard, Bruno Bouchard, and Sébastien Gaboury. Basic daily activity recognition with a data glove. *Procedia Computer Science*, 151:108 – 115, 2019. The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops.
- [nd] Neurodigital technologies, s. l.
- [pha] Phantom desktop device, 3d systems inc. accessed 06 Sep 2020.
- [qua] Conversion between quaternions and euler angles. accessed 19 Aug 2020.
- [SEBL06] Fredrik Sebelius, L Eriksson, Christian Balkenius, and T Laurell. Myoelectric control of a computer animated hand: A new concept based on the combined use of a tree-structured artificial neural network and a data glove. *Journal of medical engineering & technology*, 30:2–10, 02 2006.
- [SRL05] Fredrik C.P. Sebelius, Birgitta N. Rosén, and Göran N. Lundborg. Refined myoelectric control in below-elbow amputees using artificial neural networks and a data glove. *Journal of Hand Surgery*, 30(4):780–789, Jul 2005.
- [Sut65] Ivan E. Sutherland. The ultimate display. In *Proceedings of the IFIP Congress*, pages 506–508, 1965.

- [WGL⁺19] Dangxiao WANG, Yuan GUO, Shiyi LIU, Yuru ZHANG, Weiliang XU, and Jing XIAO. Haptic display for virtual reality: progress and challenges. *Virtual Reality & Intelligent Hardware*, 1(2):136 – 162, 2019.
- [ZWW⁺18] Yukai Zheng, Dangxiao Wang, Ziqi Wang, Yu Zhang, Yuru Zhang, and Weiliang Xu. Design of a lightweight force-feedback glove with a large workspace. *Engineering*, 4(6):869 – 880, 2018.

Acknowledgments

I'd like to thank...

Copyright Information

Voyager Bridge Mesh

Acquired from <https://www.trekmeshes.ch/> on 20 June 2020.

This mesh remains the property of Chainsaw_NL and Star trek Meshes. It however may be used to create images and scenes for non-profit use only.

Any images produced with this mesh do NOT have to be credited to the mesh author. But it would be nice.

Star Trek and Enterprise are copyright Paramount Pictures.