Modelling natural human pointing for target prediction in VR

TOR-SALVE DALSGAARD
mhb558@ku.dk - University of Copenhagen
02. March 2020

MASTER'S THESIS

SUPERVISOR
Joanna Bergström - joanna@di.ku.dk

ABSTRACT

ACKNOWLEDGEMENTS

Thanks to all the members of the HCC-section at DIKU, for their help and patience with my different problems. Thanks to Carlos for the help with 3D-printing, Jess for the help with OptiTrack, Jonas and Klemen for all the weird ideas.

A special thanks to Joanna for guidance, ideas and all the feedback you could wish for.

sd

CONTENTS

Abstract 2

Acknowledgements 3

1 Introduction 5

2 Related Work 6
2.1 Pointing as a referential gesture . . . 6
2.2 Body and mind . . . 7
2.3 Target selection in VR . . . 7
2.3.1 Ray casting . . . 7
2.3.2 Virtual hand . . . 7
2.4 Target selection trigger . . . 7
2.5 Systematic pointing error . . . 8
2.6 Pointing and Machine Learning . . . 8
2.7 Summary . . . 8

3 Capturing pointing movement 9
3.1 Apparatus . . . 9
3.1.1 Tracking . . . 9
3.2 VR Application . . . 10
3.2.1 Targets . . . 10
3.2.2 Virtual avatar . . . 11

4 Collecting pointing movement 13
4.1 Measures . . . 13
4.2 Tasks . . . 13
4.3 Logging . . . 15
4.4 Procedure . . . 15
4.5 Participants . . . 16
4.6 Error sources . . . 16

5 Modelling pointing movement 17
5.0.1 Terms and definitions . . . 17
5.1 Collection . . . 18
5.1.1 Exploring the pointing data . . . 19
5.1.2 Take-aways from the calibration data . . . 19
5.2 Preprocessing . . . 19
5.2.1 Normalizing by participant height 20
5.2.2 Moving the starting point . . . 20
5.3 Transformation . . . 20
5.3.1 Index finger position . . . 22
5.3.2 HMD position . . . 24
5.3.3 Orientations . . . 24
5.3.4 Elbow flexion . . . 26
5.3.5 Shoulder abduction . . . 27
5.3.6 Horizontal shoulder angle . . . 28
5.3.7 Index finger above head . . . 28
5.3.8 Index finger above hand . . . 29
5.3.9 Index finger position relative to body . . . 29
5.3.10 Distances . . . 30
5.4 Selection . . . 32
5.4.1 Selected features . . . 33
5.5 Modelling . . . 33
5.5.1 Pipeline . . . 34
5.5.2 Classification . . . 35
5.5.3 Regression . . . 37
5.5.4 Summary . . . 38

6 Discussion 39
6.1 Collecting the pointing movement . . . 39
6.2 Modelling the pointing movement . . . 39
6.3 Pointing movement as interaction technique . . . 40

7 Future work - Future direction 41
7.1 Next steps . . . 41
7.1.1 Improving features . . . 41
7.1.2 Exploring boundaries . . . 41
7.2 Near future . . . 41
7.2.1 More realistic data . . . 41
7.2.2 Predict targets based on whole movement . . . 41
7.3 Far future . . . 42
7.3.1 Building an interaction technique 42
7.3.2 Using other tracking technologies 42

8 Conclusion 43

References 44

List of figures 46

List of tables 48

A Short names for data points and body fields 49

B Feature selection 50

C Machine learning 51
C.1 Model parameters . . . 51
C.2 Regression: predicted targets . . . 52

1 INTRODUCTION

Pointing devices play an important role for users interacting with a computer. Computers need to interpret interaction targets and users can indicate their intentions by the use of such a device. Traditionally, mice are used for this task and for the past many years, users have been able to use their fingers to point on touchscreens and touchpads. Similarly do *Virtual Reality* (VR) applications support interactions by using a controller as pointing device. These pointers are typically based on gaze, rey casting or the virtual hand of the user. VR enables a wide range of applications, that are not possible in a classical 2D application, also because users of VR have the possibility of fully immersive and authentic experiences and use their bodies as an interaction device. Applications that focus on the naturalness and realism of the experience also need pointers for interactions, but pointers based on eg. rey casting easily break the immersion of the application. The virtual hand, which is based on the hand movement of the users, is not always desirable, since it does not work well for far targets. We thus introduce the idea of an interaction technique based on the natural human pointing. We build a model for how the human pointing movement can be used for such an interaction technique.

Ever since early childhood, humans point at things of desire and intention. This skill is important for infants, while they can not express these intentions through speech yet. But humans use the pointing gesture as a reference to the thing of interest and not as an accurate interaction technique [14]. Early efforts by Bolt [5] show that mid-air pointing is suitable for selecting targets in a 3D environment. Since then pointing has been used more and more, but typically in connection with a controller for rey casting. Plaumann et al. [25] uses cursorless mid-air pointing in a smart-home environment without a controller and finds humans to be inaccurate especially on distant targets. Efforts to correcting for these inaccuracies of the human pointing have been made by Mayer et al. [20]. We explore the human pointing based on numeric attributes, by reducing the complexity of the body movement to positions and orientations.

VR opens up for many opportunities and experiences. Not all current input techniques support the immersiveness and the feeling of naturalness of these experiences, since they introduce some abstract cursor or some rey, which is used to point at targets. Humans have capabilities to point at desired objects by nature. In this work we want to use those capabilities to construct a model that can be used to build an input technique that supports the naturalness of pointing and equally usable as other selection techniques. Such a technique is usable for applications, where the users hands need to be freed for other tasks, while allowing for target selection. For instance do Plaumann et al. [25] outline the use-case of smart homes, in which the proposed technique also could be used in.

These recent efforts show that a lot of factors play a role, when constructing a model for human pointing. The model needs to encapsulate all these factors to make sense of mid-air pointing. One approach to do this encapsulation, can be done using *machine learning*. Machine learning offers great flexibility, since algorithms themselves can decide which characteristics and peculiarities of the human users it wants to weight more or less. In addition to this we can emphasize certain characteristics, that we find important. Analysing these weights can also give further insights of what, in a numerical sense, is important for a general model for human pointing.

In this work, we conduct a data collection study, where we capture the natural human pointing movement. Participants point at fixed targets and thus provide data on how humans point at objects. The collected data is used as a basis for a machine learning model, that then gives insights for how to model the human pointing. From the data and the machine learning model we identify characteristics of human pointing, eg. the position of the index finger. With this we show that it is possible to model human movement using machine learning. This work does not apply the found model to a selection technique, but discusses this next step.

source	chaining	raw results	selected results
ACM Digital Library		4724	150
Google Scholar		22470	150
Poupyrev et al. [26]	forward	187	5
Schwind et al. [29]	backward	48	5
Mayer et al. [21]	backward	13	4
Yu et al. [33]	backward	56	1
Total		27498	315

Table 1. Number of papers and articles found and selected in search

2 RELATED WORK

In this section we review related work to paint a picture of currently used technology and methods to model body movement. The review aims to find answers to the questions,

(1) *Which pointing techniques are commonly used in applications?*
(2) *Can we assume that the movement of the arm is different when pointing at targets with varying distance?*

We focus on these questions, but also explore other important topics regarding application design in VR and themes such as *"pointing in a social context"*.

This review is based on several papers and articles, which were searched and found on the ACM Digital Library and Google Scholar. Keywords in these searches where the phrases `pointing`, `ray casting` and `target selection`, always in conjunction with the phrase `(VR or "Virtual Reality")`, to make ensure relevance of the results. Additionally, the search was restricted to the past 5 years. Only the first (up to) 50 for each search where include, to keep the number of results reasonable. To that comes some papers found later in the process. These results were found by forwards and backwards chaining. For a complete overview see table 1. Afterwards, the remaining papers were first filtered by title, then by abstract and at last by skimming the text, leaving 40 papers.

2.1 Pointing as a referential gesture

Pointing is an important method of communication between humans, making it one of the most important referential gestures. The gesture of pointing is classified as being deictic, meaning that it is a gesture learned in early childhood and used as a method signify intention and give sentences and actions context. With pointing a human can reference an object that is far away or close by, making this gesture very versatile. The pointing gesture is not restricted by hands or index finger and can also be conducted with the head, the feet, the thumb and even the eyes. Pointing is used to indicate an object, a location, a direction or another human.

This deictic gesture, when used by infants and children, is divided in three purpose-areas: *Imperative*, *declarative* and *epistemic*. Imperative gestures are used to describe the desire to obtain an object of interest, the declarative is used to direct attention to an object and the epistemic is used when pointing is used to receive new information. [7, 18] Kendon [14] describes different postures when pointing different distances and objects. For instance does the extended index finger refer to a specific object, location or human, while an open hand gesture indicates a more loose direction or similar. Kendon also observes that

different distances are indicated by different poses of the arm: When the arm is stretched the object of interest is further away, compared to when the arm is angled.

2.2 Body and mind

Body ownership is an important topic in *human-computer interaction* (HCI), since it determines immersiveness of VR-applications. Immersive VR is often desired in games and for educational purposes [32]. Slater et al. [30] defines presence in VR as the subjective feeling of *"beeing there"*. The perceived presence is affected by the fidelity of the virtual avatar, that is controlled by the user, and the overall virtual environment. Among others has Schwind et al. [28, 29] conducted studies that compare different avatars, eg. male vs. female hand and abstract vs. human-like body. The results of this research indicate that more realistic representations of avatars are more effective than more abstract ones. Additionally the perceived body ownership can be preserved when displacing the hand in virtual space from physical space [10] or when elongating the arm (with the hand attached) [17].

2.3 Target selection in VR

Target selection is important in all HCI since it lays a foundation on how humans communicate with a computer. Just as in the real world, does pointing on the interface show human intent. The interaction with the classical cursor (eg. the mouse) on a two-dimensional interface (eg. the desktop) is well understood, compared to the interaction of a cursor in three-dimensional environments (eg. VR) [31]. In VR target selection becomes non-trivial since the selection technique needs to be precise and fast while dealing with eg. occlusions. This problem is less prominent in two-dimensional spaces [33]. Different techniques have been proposed for target selection in VR.

2.3.1 Ray casting. Ray casting is a widely used target selection technique. It moves a cursor based on a ray originating somewhere off the user, travelling through the room until it crosses paths with some target. Argelaguet et al. [2] categorizes ray casting methods into two families: *hand-* and *eye-rooted* techniques. Prominent hand-rooted ray casting methods include *Index Finger Ray Cast* (IFRC), which casts a ray extending the index finger [8]. An additional hand-rooted method is the, where, similarly to IFRC, the ray is an extension of the forearm [22]. Eye-rooted techniques are split further into two methods: *Gaze Ray Cast* (GRC), which uses the eyes gaze to cast a ray [22] and *Eye-Finger Ray Cast* (EFRC), which casts a ray based on the line between the eye and index fingertip [21]. Other applications use a ray originating from the centre of the *Head Mounted Device* (HMD).

2.3.2 Virtual hand. Other techniques for selecting targets in VR fall under the category of the *virtual hand*. With the virtual hand, a hand is rendered in the VR environment, based on the movement of a controller or other tracking technology. This technique requires the user to be close to the target that is to be selected since it needs to be "touched" [33]. Poupyrev et al. [26] has proposed the well-known *Go-Go* interaction technique, where the user's arm is non-linearly elongated to reach distant targets. Among other do Gallo and Minutolo [12] use a Wiimote as a pointer to select targets.

2.4 Target selection trigger

A challenge in many 3D selection tasks is how to trigger a selection action. The typical approach is to use a button on a controller, which either is used for the pointing action (eg. in virtual hand) or hold in the hand, that is not used for pointing. Another method for triggering a selection is to require the user to dwell on a target for some fixed amount of time. Nukarinen et al. [24] make a comparison of three techniques, that use these approaches and they show that a virtual hand technique with a trigger button performs the best. Quantitative results show that a dwelling approach is the most unnatural and suffers from accidental selections, while a gaze-based approach seems the most natural and makes a hands-free interaction possible.

2.5 Systematic pointing error

Humans have a limited accuracy when pointing, making systematic errors, as they overreach targets on distant targets [11]. Ray casting is thus imprecise at longer distances [33]. Mayer et al. [21] has proposed a model for correcting this error. This model is user-, pose-, and distant-independent which makes it rather robust. Batmaz et al. [4] found that humans have depth perception issues in VR and also [25] describes large errors on distant targets, but connects these errors with eye dominance and handedness.

2.6 Pointing and Machine Learning

Not much of the recent research effort in target selection in VR has been conducted with the help of machine learning. Noy [23] predicts targets in 2D already in 2001, while Nickel and Stiefelhagen [22] recognize the pointing gesture in 2003. More recently Casallas et al. [6] did use decision trees for predicting moving targets on large displays, using a wand as the pointing device.

2.7 Summary

Target selection is a much researched topic in HCI. This review aimed to answer the following questions,

(1) *Which pointing techniques are commonly used in applications?*
There are multiple interaction techniques used for pointing. The most common is ray casting. These methods work well at a distance but are imprecise at close targets. Additionally is the ray cast using eg. EFRC not natural by design, for instance, if a human points towards the ceiling and hold the index finger at eye level, the ray would still be cast straight in front of the user. Human pointing and thereby ray casting directly based on human pointing accuracy (eg. IFRC and FRC) suffers from a systematic error, that is so severe, that the need for compensation arose [4, 20, 21].

(2) *Can we assume that the movement of the arm is different when pointing at targets with varying distance?*
Distant targets have proven to be hard for humans to point at. [11, 33] Nickel and Stiefelhagen [22] shows that human gestures can be recognized by a machine, meaning that we can differentiate between gestures and that there is some underlying pattern to human movement. Mayer et al. [21] builds a model for correcting systematic pointing errors, showing that there is some pattern to the general human pointing movement. We can thus assume that there is a difference in human pointing pattern based on different target positions.
Kendon [14] describes different gestures for different pointing actions and intends, meaning that there is a difference in gestures based on the action/intend. We can use this to argue that we can construct a machine learning algorithm that is not based on pure correlation. In this work, we are interested in the declarative deictic gesture, described by Cochet and Vauclair [7].

There has been much research in target selection techniques. Ray casting-based techniques are popular and have been improved by correcting for human errors. It seems that no recent research uses machine learning for target selection. Some problems with selection and pointing have been solved, while others, such as "naturalness of interaction" have not been researched fully.

3 CAPTURING POINTING MOVEMENT

To model the human pointing, we need to collect data that describes the movement while pointing. For this purpose, we need an application that can collect this exact data. We use VR to present the user with a controlled environment, in which the user points at some targets in the space. In the following we will describe the technical properties of the application, that later is used to collect data in a study.

We define the *standard pointing position* as the body-position in which a person lifts their arm to a point, where the arms stands in a right angle to the body and points to the front of the user. Additionally we will use the format `(0, 0, 0)` to denote target coordinates in the virtual space, with real world units.

3.1 Apparatus

To run the application we use a Windows 10 PC with a Nvidia GeForce 1080 graphics card and to track markers, an OptiTrack motion capturing system with 24 cameras. The VR application is simulated using the HTC Vive Pro HMD with a wireless setup. The wirelessness has advantages in the physical space, since it gives freedom when moving, but has the disadvantage of requiring a battery to be carried by the user. This disadvantage is solved by providing a small backpack. Additionally a Vive Pro Controller is used for the action of triggering the target selection.

To be able to conduct the study, we constructed a cube of aluminum rails, with 24 OptiTrack cameras attached, tracking markers inside the cube. The cube has a dimension of approximately 2 x 2.5 x 3 m and thus walkable from the inside. Additionally, a desk with the mentioned computer, running OptiTrack and the VR application, is present.

3.1.1 Tracking. OptiTrack offers an implementation of a full body skeleton based on marker sets, but these are hardly customizable, meaning that either we have to use a full skeleton, which require many markers on the full body or we track the participant based on rigid bodies. The first solution has pitfalls, since we would need to attach the markers in similar positions on every participant. This could have been solved by a suit where the markers are attached, but then again, we would need multiple suites to fit different participants. Thus the second solution is a more sensible approach, also because we are not interested in full body movement, but rather just the arm and index finger movement. The challenge with this solution is to translate the real world movement to the virtual environment through inverse kinematics.

Therefore OptiTrack markers are attached to the participant's left and right shoulder, upper arm, forearm, hand, index finger and the HMD. We mount the upper arm, forearm, hand and index finger marker sets on the participant using four 3D models, that we printed, based on the open-sourced models designed by Mayer et al. [20][1]. These models are depicted in figure 1 without the OptiTrack markers attached. The markers are attached to bolts, that are stuck through fixed holes in the models.

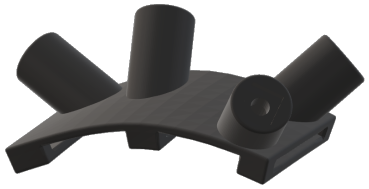
The upper arm, forearm and hand marker mounts are attached to the participant by use of 25 mm hook-and-loop fasteners (also known as Velcro straps). The straps are modified with elastic band to make them more comfortable to wear. Without the elasticity, the marker sets can be an irritation to wear, since the straps are too tight when the participant eg. is contracting their upper arm muscles while pointing.

To track the index finger a marker set (fig. 1a) is mounted on the participant, by sticking their index finger into a ring with three markers attached. To accommodate different diameters of fingers, the participants index finger will first be wrapped in play dough, which fills the remaining space between ring and finger. This is done to ensure a similar placement of the marker set between participants.

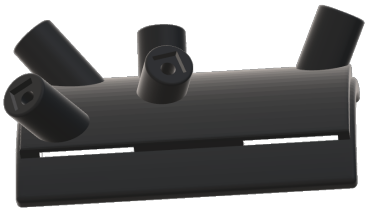
The left and right shoulder marker sets are based on OptiTracks "Rigid Body Marker Base" with four markers attached to each. These two marker sets are attached to the straps of the backpack, using Velcro. The backpack doubles as storage space for the HMD's battery.

[1] https://github.com/interactionlab/htc-vive-marker-mount *(accessed: 11.12.19)*

(a) Close-up on the 3D model of the index finger marker set, which holds three markers with a diameter of 1 cm.



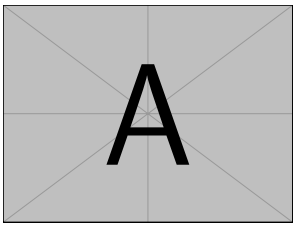
(b) Close-up on the 3D model of the hand marker set, which holds four markers with a diameter of 2 cm.



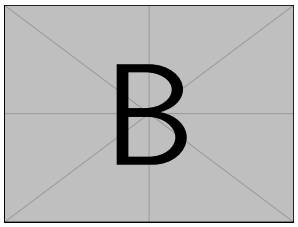
(c) Close-up on the 3D model of the forearm marker set, which holds five markers with a diameter of 2 cm.

(d) Close-up on the 3D model of the upper arm marker set, which holds five markers with a diameter of 2 cm.

Fig. 1. Close-ups on the 3D models of the marker sets used for tracking with OptiTrack. Designed by Mayer et al. [20].



(a) A user, facing front, with the marker sets attached.



(b) A user, facing back, with the marker sets attached.

Fig. 2. A user wearing the OptiTrack marker sets.

Also the HMD is modified with markers, such that the marker set is composed of four markers, each one on a OptiTrack "Marker Base". For completeness: The movement of the camera in the application is controlled by normal SteamVR tracking using Valve Base Stations and not through the attached marker set.

Participants are thus equipped with 7 sets of markers for a total of 29 markers. The full setup is depicted in figure 2, attached on a user.

3.2 VR Application

The actual application visible in the HMD is implemented using Unity 2019.2.6f1. The virtual world is kept simple, such that the user can focus on the pointing movement without distraction.

3.2.1 Targets. Also the targets are kept simple. The users point at spherical targets with a green color, such that they are visually distinct from the bright background and have a single size parameter, i.e. the diameter. Other, more realistic, but also more complex shapes have many parameters and can thus be hard to evaluate [31]. The diameter of the targets in this application is 15 cm.

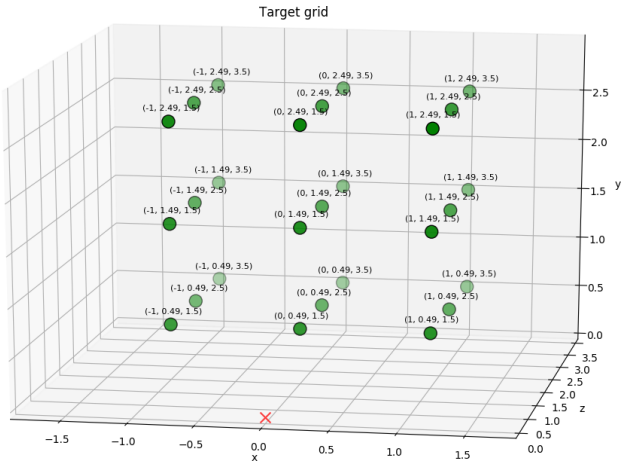


Fig. 3. Each target (green) labeled with its positon arranged in the 3 x 3 x 3 grid. The red cross is the position users stand on. [m]

Fig. 4. The virtual hand pointing.

Possible targets are arranged in a 3 x 3 x 3 grid. We name the dimensions of this 3D grid row, column and aisle. The row in the horizontal dimension, column the vertical dimension and aisle the depth dimension. The closest three rows are 1.5 m away from the user baseline and the furthest are 3.5 m away. The center columns are directly in front of the user, while the other two columns are offset with 1 m to the left and the right respectively. The bottom aisles are 0.49 m above the ground, while the highest aisles are 2.49 m above the ground. Between all consecutive rows, columns and aisles is a space of 1 m. This means that the middle aisles are at the average human shoulder height, 1.49 m [13][2]. Figure 3 shows a depiction of the target grid, with its positions in meters as label above the targets.

The arrangement in a grid allows for control over the tested targets. This is in line with the initial research question, since we hope to "provoke" changes in arm movement when changing the target. A random arrangement of targets would allow for more generalizability, but would also require many more data points, to ensure differences in the data and that the distribution of target positions is relatively evenly spread across the dataset. This would the require participants to point at many targets, such that individual bias would be reduced, but this would consequently lead to more participant fatigue, which again leads to some bias.

When selecting a target, a user needs to do some kind of defined action to trigger an action a computer. Eg. does the user need to click a button on a mouse or press a button on a controller in VR. Similarly we need an action to signal the computer that the prior movement pattern defines a pointing action at a specific target. For this purpose a HTC Vive Pro controller is placed in the left hand of the user. To start a repetition the touchpad is pressed and to stop it the trigger is pressed. The reason for binding the start and stop actions to two distinct buttons is for avoidance of accidental presses and unwanted double clicks.

3.2.2 Virtual avatar. The user controls a human-like virtual hand avatar. Schwind et al. [29] has explored the effects of different types of avatar in pointing tasks and concluded that human-like avatars are most precise. The avatar used in this application is depicted in figure 4. Users are used to rely on visual clues for pointing, thus it is important to show an avatar, that follows the actual movements of the user, to avoid the pitfall of unintentional redirection.

[2]Data collected from U.S. adults. See also: https://www.ergocenter.ncsu.edu/wp-content/uploads/sites/18/2017/09/Anthropometric-Summary-Data-Tables.pdf *(accessed: 14.01.20)*

Rigidbody	**Transformation vector** [mm]	**Orientation vector** [degree]
HMD	(-200, 0, 0)	(0, 0, 0)
Left Shoulder	(-70, 0, 0)	(0, 0, 0)
Right Shoulder	(-70, 0, 0)	(0, 0, 0)
Upper arm	(0, -45, 0)	(0, 0, 0)
Forearm	(0, -35, 0)	(0, 0, 0)
Hand	(-50, -10, 0)	(90, 90, 0)
Index finger	(25, -20, 0)	(0, 0, 0)

Table 2. Transformation and rotation vectors for the rigidbodies. The unit of the transformation vectors is millimeter and degrees in the case of the orientation vectors. The coordinate system is in y-up fashion.

The marker sets described before and shown in figures 1 and 2, are grouped to rigidbodies using OptiTrack and send off to Unity for further visualization. Each rigidbody has a position and orientation, which can be customized as needed. All rigidbodies are at their identity orientation when a is in user the standard pointing position. After that a transformation and rotation was applied to each rigidbody, as seen in table 2. The transformations are done to estimate the center of the body-part the rigidbody is attached to. Eg. is the upper arm marker set positioned on the biceps and thus the rigidbody is positioned 35 mm below the marker set to approximate the center of the upper arm. This is done to get more precise orientation. The orientation is mostly left as is, but in some cases it is necessary to adjust the orientation, such that the virtual avatar aligns with the real world hand.

For computing the scale of eg. the avatar, we need to find the function translates distances in the real world to distances in the virtual world. This function can be found by using an *OptiTrack Calibration Square (OCS)*, which is used for calibrating the ground plane in OptiTrack. The OCS consists of two arms, which are at a right angle to each other and equipped with a total of three markers, one at each end and one at the corner. The distance between the end-markers and the corner-marker is 40 cm and 30 cm respectively. By placing the OCS in the tracking space and computing the distances between the markers in Unity we find the translation function to $f\left(x\right) = \frac{2x}{100}$, where x is in cm. Thus 100 cm in the real world corresponds to 2 units in Unity.

4 COLLECTING POINTING MOVEMENT

The data collection study aims to generate a dataset, that contains information about the natural human pointing movement. The emphasis in this case is on the naturalness of the movement, such that each participant unconsciously decides their pointing gesture and how they aim. This is done to assist the overall goal of building a model for human pointing, that feels free and natural when used as an input technique.

The experiment follows a within-subjects design wherein the variation is the 27 different targets. Each participant points a each target five times, which gives $5 * 27 = 135$ movement patterns for each participant. The order of the 135 pointing actions are randomized to counterbalance across participants.

4.1 Measures

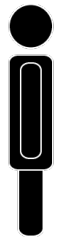
We want to collect participant measures that are meaningful for later modelling of the pointing movement or needed for arguing for generalizability. Before each participant starts their trial, we collect the following measurements:

- Demographics. We collect the age, gender and handedness of the participants. This information has no expressiveness but is collected for purposes of comparability and generalizability. The information also has no influence on the actual VR application.
- Height. The height is important for calibrating the application, since the size of the virtual avatar needs to be in line with the proportions of the participant. The height is a property that is interesting to look at while creating a model, since it has an influence on the viewing angle and thus on for instance the absolute height of the index finger while pointing. Note that the height is provided by the participant themself, and not measured as part of the experiment.
- Upper arm. We want to measure the general length of the upper arm and, since one of the OptiTrack marker sets is attached to the upper arm, the distance between the center of the marker set and the elbow.
- Forearm. Also the length of the forearm is measured. Together with the length of the upper arm, the full length of the arm can be computed by summing the two lengths. Similar to the upper arm, does the forearm have a marker set attached and thus the distance between marker set and elbow has to be measured. With these informations and using trigonometry, we can compute the position and angle of the elbow, given the position of the upper- and forearm rigidbodies and these elbow-marker set distances.
- Right shoulder. Just as we want be able to compute the position of the elbow, do we want to find the position of the right shoulder. We can do this by measuring the horizontal and vertical distance from the right shoulder marker set and the actual shoulder. Adding both values to the right shoulder marker set position results in the shoulder position.

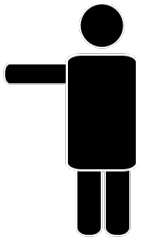
4.2 Tasks

The participant is asked to do two different tasks; a calibration task and a collection task. In both tasks does the participant have a controller in the left hand and is wearing the marker setup shown in figure 2. The controller has two buttons, we will call A and B.

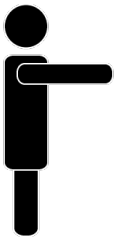
The calibration task serves two purposes, firstly to gather biometric data from the participant, which later is to be used for normalization of the dataset, and secondly to take measures such that the virtual avatar has similar body proportions as the real body. In addition to the participants height, will the calibration data help ensure a more realistic representation of the avatar. Figure 5 shows the three different body positions necessary for completion of the task. All positions are variations of the standard pointing position. In the first task the participant lowers their arm to a zero degree angle (fig. 5a). In the second task the participant assumes the standard pointing position (fig. 5c) and in the third task the participants



(a) Calibration task (1): Side view of a participant with their arm pointing down.

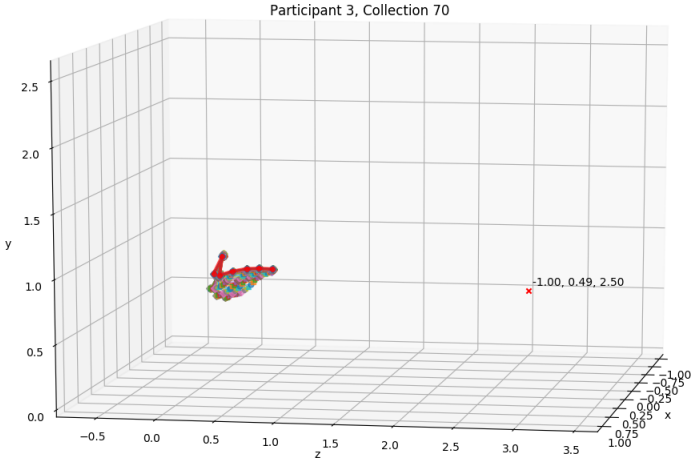


(b) Calibration task (2): Front view of a participant with their arm pointing to the right.

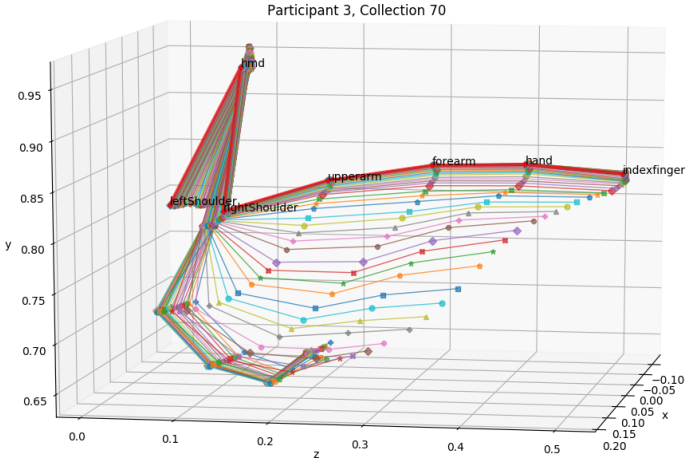


(c) Calibration task (3): Side view of the standard pointing position.

Fig. 5. A depiction of the calibration tasks.



(a) Visualization of a logged pointing movement. The red cross is the position of the true target. [m]



(b) Zoom in of the movement pattern visualization in figure 6a and with labels on the rigidbody positions. [m]

Fig. 6. Example of a visualization of the collected movement data. Each line represents a sample of a timestamp, while the slightly thicker red line is the final position of the arm. Each point on the line is the position of a rigidbody tracked. This movement took 1.8 seconds to complete.

points the arm to the right, still with a 90 degree angle to the body (fig. 5b). Each time a participant assumes the required position and then presses button B on the controller.

In the collection task, the participant is asked to point at the shown target. They point at each target five times, for a total of 135 repetitions. Before the each repetition the participant is asked to put the arm in a predefined position. The participant now presses button A to start the repetition. This makes a target appear, which is to be pointed at. When the participant is confident that their finger points at the target, they press button B. This concludes one repetition.

This task is performed in a very controlled setting and is designed to collect the movement data between the two button presses. The design of the task aims to promote natural pointing movement, even though accurate pointing is not the strong suit of humans.

4.3 Logging

For this experiment we want to collect data, that expresses the movement of a participant, while pointing at a known target. The logs are saved as `.csv`-files and to avoid saving for instance the participant data repetitively, the data is scattered across different files. All positions are in logged in a y-up fashion.

- Participant. The participant data, described in section 4.1, is saved to a `.csv`-file, where each line represents one participant, identified with an unique integer.
- Movement pattern. For each repetition we log the movement pattern of the right arm and upper body of the participant, with a frequency of 50Hz. The log for one repetition consists of a number of samples. One sample consists of the position and orientation of each tracked rigidbody and a timestamp, which is the time in seconds after the repetition was started. When the participant finishes the pointing movement, a final sample is logged. Thus each collection of samples fully describe the pointing movement of one task repetition. An example of such a collection is visualized in figure 6. The orientational data is logged as euler angles (a triplet of pitch, yaw and roll, similarly used by Deldjoo and Atani [9]). The data is logged such that the positions of rigid bodies are in meters.
- Calibration. The calibration task is logged by taking a sample, when the participant triggers the calibration. Thus the resulting calibration `.csv`-file consists of three lines, one for each sample created in the calibration task.
- True target. The target the participant is asked to point at is the denoted the true target. This target is logged in a separate `.csv`-file for each participant. In this file each line represents the position of a true target and has an identifier, that fits to a collection.

4.4 Procedure

A small pilot of the experiment with 3 participants showed that the initial position of the arm is influential on the movement pattern of the arm. Generally, when the arm was pointing straight to the floor to begin with (similar to the position in figure 5a) the participant tended to have keep the arm stretched, which is unfortunate for validity of the initial research question. On the other hand when participants had a more relaxed arm to begin with, the elbow was bend more reliably.

The experimenter followed this process to guide a participant through the experiment:

(1) Introduction. To begin with we want the participant to be comfortable and know their task. Thus the experimenter should go through the experiment and answer potential questions. The goal of the project is introduced as creating a model for the movement pattern when humans point. Important to know for the participant is that they can not point wrong or do anything else wrong. They should point as they would do if they would direct a friends attention the target.
(2) Attach the marker sets. After the introduction the equipment is mounted on the user. This includes the marker sets, but not the HMD, to allow the user to see the real world, while the participant is introduced to the application. This is also a good time to measure the discussed measurements.
(3) VR. To make sure that the participant knows which buttons to be pressed in the tasks, the controller is shortly introduced. After attachment and measures the participant is mounted with the HMD to experience the VR application. If the participant is not familiar with VR or the participant would like to, they have time to look around and get accustomed in the virtual world.
(4) Calibration task. The participant starts with the calibration task. For this task the participant should point straight in the mentioned directions, one after the other.
(5) Participant behavior. The participants are instructed to start each repetition by relaxing their arm and placing their hand on their stomach near the navel. This forces an initial bend on the arm and thus helps a more natural pointing movement on the way. The participant is asked to make a natural movement when pointing. They should not think too much about their movement, but "just do it".

	mean	std	min	max
Forearm length	0.28	0.02	0.24	0.33
Forearm marker set distance to the elbow	0.12	0.03	0.07	0.18
Index finger length	0.09	0.01	0.07	0.12
Upperarm length	0.29	0.03	0.26	0.36
Upper arm marker set distance to the elbow	0.12	0.03	0.10	0.17
Height	1.79	0.10	1.65	1.94
Horizontal distance from right shoulder marker set to right shoulder	0.05	0.01	0.03	0.07
Vertical distance from right shoulder marker set to right shoulder	0.11	0.03	0.06	0.15

Table 3. Overview of the participant measures. [m]

(6) Data collection task. The first repetition of the data collection task should be guided by the experimenter, such that the participant can focus on pointing. Generally the learning curve of learning this task is not steep and thus most participants can continue the task after the first guided repetition. Otherwise should the experimenter not intervene when the participant is completing the experiment.
(7) Closing remarks. If the participant has comments or questions, they may ask them at the end.

4.5 Participants

The collection study was conducted with 13 participants, with a mean age of $\mu : 24.46$ $\left(\sigma : 1.98\right)$. Of the participants where five female, seven male and one identified as another gender. All participants where right handed and most participants had no prior experiences with VR. The participants where rewarded with commodities that corresponds to approximately 100 dkk in value. In table 3 we generally can see that the participants are within a margin of the general public in terms of body measures when compared to the work Gordon et al. [13] have done.

4.6 Error sources

The design of the study aims to mitigate errors and meaningless data where possible. For instance where the marker mounts fastened with flexible material, such that they are as unintrusive as possible as the participants where bending their arm. But there will always be some unknown factor of how much the marker mounts changed the participants pointing behavior.

Additionally there might be inaccurate participant measures, due to human error. Also, as mentioned before, was the height of the participants not measured during the experiment, but rather it was self reported.

Lastly there also might be some misalignment between the real hand and the virtual avatar, which might lead participants to point inaccurate or create an uncanny valley effect, that lead to irritations and thus inaccurate pointing.

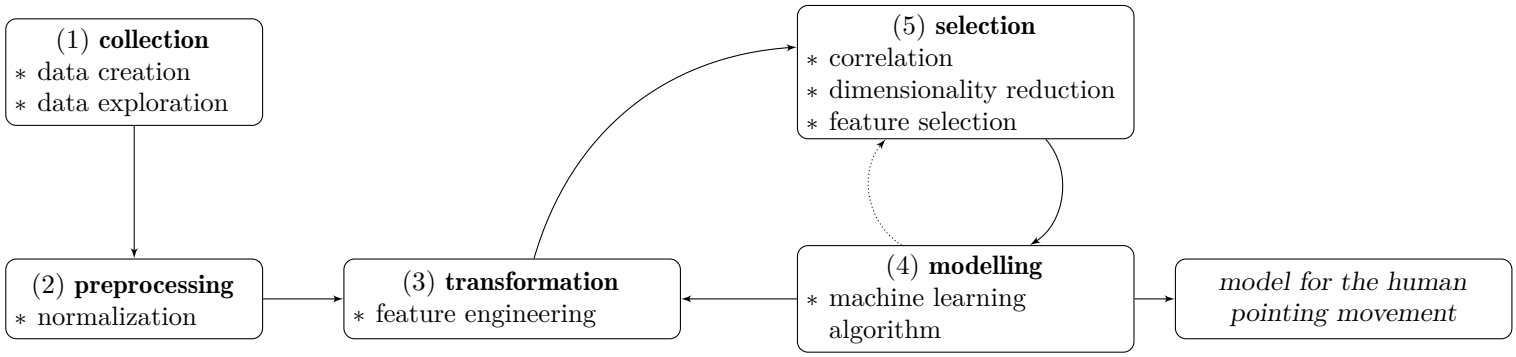


Fig. 7. The process-diagram of modelling the human pointing movement.

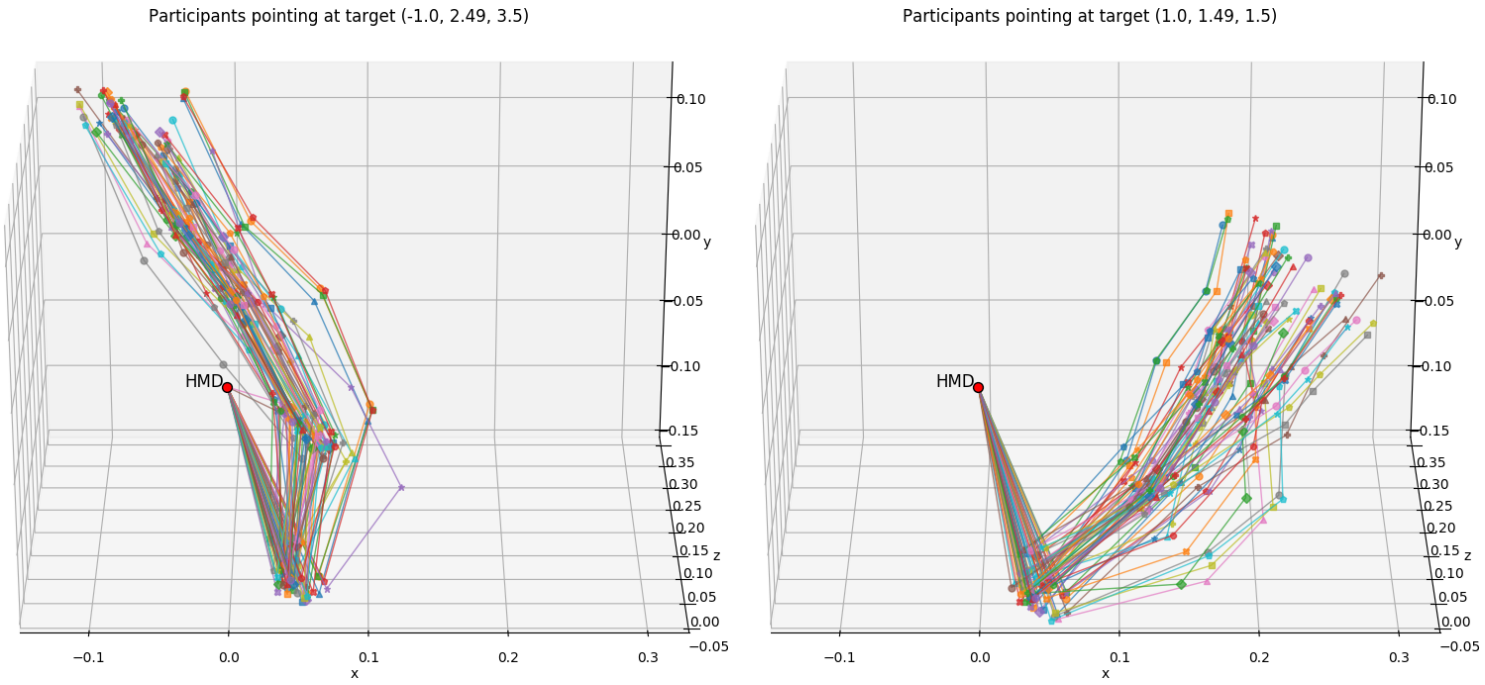
5 MODELLING POINTING MOVEMENT

In this section we want to structure the process of modelling the human pointing movement in a meaningful way. The process is outlined in figure 7 and consists of five phases, with an iterative part between two of them. Rawat [27] proposes a similar process for building machine learning models.

(1) Collection. The first phase of the modelling process consists of collecting the data and of exploring the collected data. This exploration gives insights about the data and might validate some assumptions about the data and also help validate the overall study.
(2) Preprocessing. The data needs to be processed, since we work with humans, where the body proportions are different by nature. Thus we need to find a formula to normalize the data, in order to be able to compare data points across participants. This phase should also be used to deal with missing data points and outliers.
(3) Transformation. To create a model for human pointing, it is important to engineer features, that describe the pointing position in a meaningful and concise way. In this phase features are constructed, both based on assumptions rooted in the collection step, on correlation and on other, more general feature engineering techniques. Features are a way of expressing the raw data in different ways and thus allowing for a new angle on the data set as a whole. This is the most time consuming, but also most important phase in the process, since the soundness of the model depends on the underlying features.
(4) Selection. In the selection phase we select features by analysing the data using different techniques. Features are iteratively constructed in the transformation phase and then evaluated based on results obtained in the modelling phase, meaning that this phase tests the constructed features, based on some score produced by the modelling phase and correlation and other statistical methods. The resulting selected features are the basis for the resulting model for human pointing movement.
(5) Modelling. The modelling phase models the pointing movement based on the constructed and selected features. It is part of the iterative process of engineering/selecting/modelling features, since it can give features a quantitative score, which can be used to select features.

This process results in a a machine learning model that can describe the human pointing. The features selected in the fourth phase then describe the properties of the human pointing movement, that are interesting for modelling the human pointing.

5.0.1 Terms and definitions. For the modelling process we define some notations and terms, allowing clearer formulas and wordings.



(a) Visualization of all endpoints that where produced by participants pointing at the target with coordinates `(-1, 2.49, 3.5)`. We can see a concise image of how the participant position their arm, hand and index finger, with a few outliers.

(b) Visualization of all endpoints that where produced by participants pointing at the target with coordinates `(1, 1.49, 1.5)`. The participants are less consistent in their arm, hand and index finger placement, when pointing at this target.

Fig. 8. Visualization of all endpoints that where produced by all participants pointing at a particular target. The data is adjusted such that the HMD-position is at `(0, 0, 0)`, in order to be able compare the endpoints in a meaningful way. Additionally, the left shoulder marker position is not shown on these visualizations for clarity.

- Endpoint. As described before, for each repetition, samples of the participants movement are logged with 50Hz. We call the last sample in each collection of samples an endpoint. Note that all data analysis presented here is based on endpoints only.
- Position data. All marker sets in each sample are logged as a three dimensional position: (x, y, z). x describes the horizontal position, y the vertical position and z the depth of the position.
- Accessing values. We construct a notation for accessing a value in an endpoint with index i and coordinate c in an endpoint e: $e\left[i_c\right]$, eg. $e\left[hmd_y\right]$ to access the y-coordinate of the HMD from endpoint e, similar to indexing notations in programming languages such as Python. The notation is rather flexible, eg. $e\left[hmd\right]$ denotes the (x, y, z)-position of the HMD and $e\left[hmd_x, hmd_y\right]$ denotes the (x, y)-position of the HMD. Similarly does the notation $p_e\left[i\right]$ access some measurement i of participant p (that produced endpoint e) and the notation $c_e^j\left[i\right]$ access some positional value in index i from the calibration c_e^j (produced by participant p_e), where j is the calibration number, see figure 5.

5.1 Collection

In this phase data is collected as described in-depth in section 4. This data consists of a set of participants, endpoints and calibrations.

HMD			right shoulder			left shoulder		
x	*mean*	-1.81	x	*mean*	9.05	x	*mean*	-9.88
	std	3.70		*std*	4.33		*std*	17.22
y	*mean*	166.33	y	*mean*	144.62	y	*mean*	141.88
	std	7.19		*std*	8.48		*std*	8.46
z	*mean*	-13.11	z	*mean*	-21.46	z	*mean*	-21.73
	std	14.93		*std*	15.37		*std*	16.41

Table 4. Overview of the selected calibration data points. [cm]

5.1.1 Exploring the pointing data. The data was produced by 13 participants repeating a pointing trial 135 times, for a total of $13 * 135 = 1755$ endpoints. Each participant produced 3 calibration samples for a total of $13*3 = 39$ samples. On average each participant completed a trial after $\mu : 1.798$ sec $(\sigma : 1.114$ sec$)$. The visualizations in figure 8 show some endpoints plotted where the participants pointed at a particular target. We can see in these visualizations that movement patterns across different humans can be inconsistent. This is what Plaumann et al. [25] and Mayer et al. [20] discuss as being one reason for the imprecise pointing of humans. From this we get reassurance that the resulting model should be flexible, such that it is able to recognize the correct intended target.

5.1.2 Take-aways from the calibration data. Table 4 shows the mean and standard deviation of the positions of a selected number of marker sets, recorded during calibration across all participants. We can use the information provided in the table to sanity check the correctness of the log and a design choice.

The correctness of the log can be checked by looking at the vertical position of the HMD and comparing with the participant height, since these values should correlate strongly. The vertical position of the HMD has a mean of $\mu : 166.33$ cm $(\sigma : 7.19$ cm$)$, while the participant height has a mean of $\mu : 179.23$ cm $(\sigma : 9.61$ cm$)$ (table 3). Note here that the vertical position of the HMD is approximately in the center of the participants head, compared to the height, which is at the top of the head. The distance between the two means is 12.90 cm. According to Gordon et al. [13], the mean distance between eye height and stature (height) is $\mu : 11.50$ cm, which, when taking into account potential errors in the measures and relatively small sample size in this study compared to Gordon et al. [13], is roughly equal to the before mentioned 12.90 cm.

A design choice was made in the collection study, regarding the height of the targets: the center rows of targets was set to a height of 149 cm, which is the average human shoulder height according to Gordon et al. [13]. The mean for the vertical position across the left and right shoulder in the calibration data is $\mu : 142.18$ cm $(\sigma : 9.52$ cm$)$. Both shoulder markers sit slightly below the actual shoulder, but this difference is recorded initially for each participant. The mean of that difference is $\mu : 10.88$ cm $(\sigma : 2.51$ cm$)$, meaning that the average shoulder height for the participants in this study is $\mu : 154.09$ cm. This tells us that, the participants from this study have a 5.09 cm higher shoulder height than the average human. The participant shoulder height is roughly 10% larger, which might be an error source. These speculations do not take potential measurement errors into account.

5.2 Preprocessing

To make the data comparable, we want to normalize it in some meaningful ways. Without normalization, the data does not compare well, since participants have differently proportioned bodies and might also have slightly different starting positions. These differences are expected, but nevertheless is it still necessary to

account for them, since else there might be unrecognized patterns in the data, which may have contributed to a better model if recognized.

5.2.1 Normalizing by participant height. To account for differences in the participants body proportions, we want to normalize the data by some known measure: the height. We do this by defining a function for each participant:

$$f_p\left(v\right) = \frac{v}{p_v\left[Height\right]} \tag{1}$$

where v is some positional value in the dataset and p_v is the participant p that produced the value v. This means that these functions transform the positional data to a percentage of the participants height, ie. $f_p\left(p\left[Height\right]\right) = 1$, but note that values may become larger than 1, eg. when the participant lifts their index finger above their head. This makes data points comparable across participants. There might still be proportional differences, in eg. shoulder height or arm length, but it is difficult to find a normalization that accounts for all the different proportions.

5.2.2 Moving the starting point. Even though participants where asked to stand on the same spot at the beginning of each repetition, it is unrealistic to expect the same exact starting position in each sample. Thus we want to move the starting point to a more adequate position, which also might not be perfect, but again has better comparability. The basic idea is to move the movement pattern, such that the starting point is centered around `(0, 0, 0)` (origin). Intuitively this means that we want to move the participants feet to stand on the origin. But since we do not record the position of the feet, we need to take the next best known position: the head/HMD, since that position is assumed to be centered over the participants feet. We also assume the participant to stand straight at each repetition. When normalizing the data we can use these two assumptions to say that the x and z values of the feet center-position is equal to the head positions x and z values. Only the y values are different. Thus we can construct a function to rectify the x and z values:

$$g\left(e\right) = \begin{pmatrix} e_x - s_e\left[HMD_x\right] \\ e_y \\ e_z - s_e\left[HMD_z\right] \end{pmatrix} \tag{2}$$

where e is an endpoint and s_e is the starting point of the pointing movement that resulted in the endpoint e.

5.3 Transformation

In the following, we present features found in the iterative transformation process. This transformation is done using feature engineering, a technique that aims to change the representation of some input data into features that describe the data in a more accurate and comprehensible way. In this case for instance, we do not want to include both the index finger position and the hand position as they have a very strong correlation and causality and thus would introduce an unnecessary bias, since redundant data introduces such biases [19]. In the transformation phase, the goal must be to construct features, that help describe the target position, based on the positional and orientational data collected in the conducted study. Most often one feature is not enough to tell us everything we want to know about the target position, which is why we change the representation of the data, until all features together can describe the target position. A collection of good features are the key to building a well performing machine learning model.

A feature can describe different types of data, such as categories or continuous values. There are different methods for constructing features, for instance

- Including raw data. One method for feature generation is to use the input data as is, making an informed choice, based on correlation and domain knowledge. For instance, in figure 9 we can see a correlation matrix that shows strong correlation between the vertical index finger position and

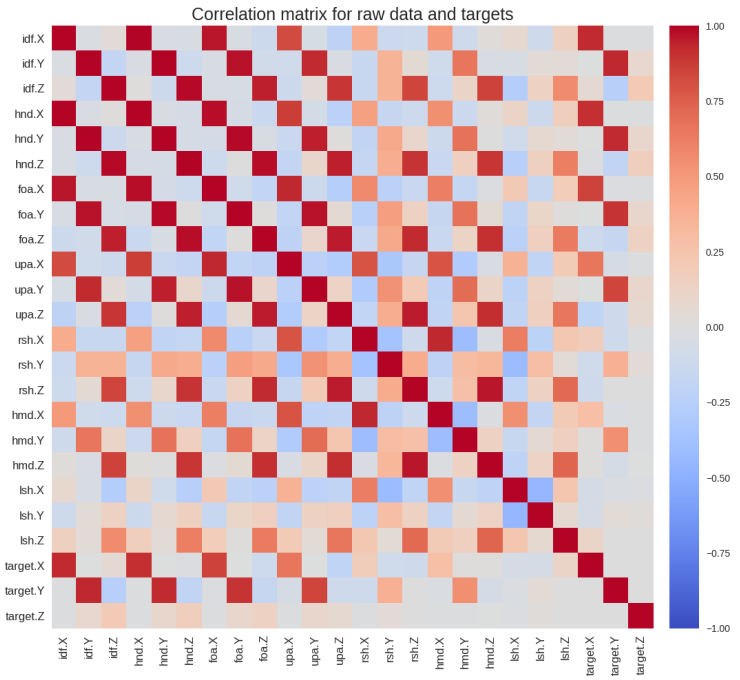


Fig. 9. Correlation matrix, showing the relationship between the marker set positions and the target position. Note that the labels are shortened, see appendix A for an overview. The lower left corner shows a strong correlation between the index finger position and the target position.

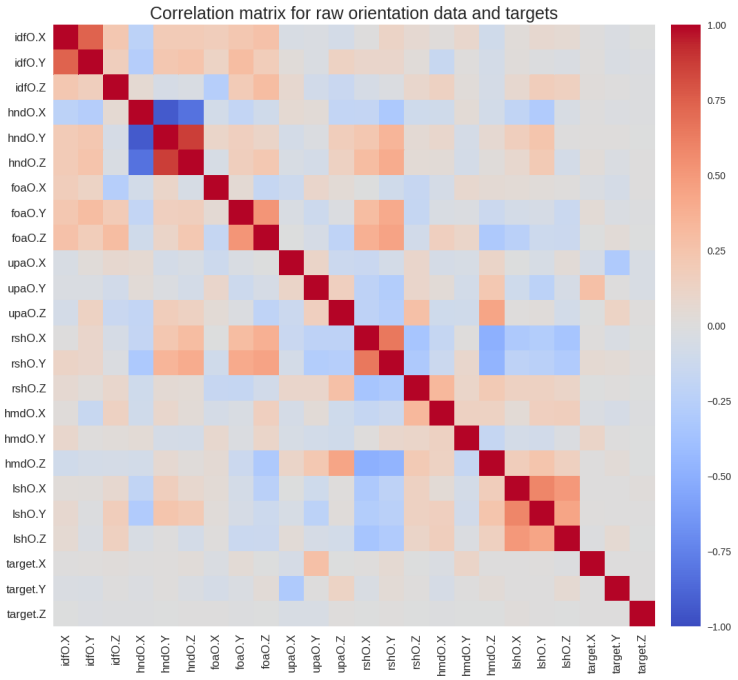


Fig. 10. Correlation matrix, showing the relationship between the marker set orientations and the target position. On this matrix we see some correlation between the upper arm orientation and the target positions in the lower center.

the corresponding target positional value. This is in line with the assumption, that humans raise their hand to point at higher targets and thus gives reason for including the raw vertical positional data as a feature for later modelling. In this case do we work with continuous data only and thus produce continuous features, when using this method.

- Binning. Sometimes we want to emphasize a property of some data point. By converting data to categories, based on their values, we can give the data more expressive power about a particular property. This can be done by binning or thresholding the data. For instance can we emphasize the vertical position of the index finger, by giving it a category, based on its position relative to the users body. Doing this we loose some information, for instance is there some correlation between the vertical index finger position and the depth of the target (figure 9), but we may gain certainty about the vertical position of the target. This results in a categorical feature.
- Combining. By combining data points, we can gain insights, that are not clear to begin with. The combining is done by some well defined formula, that is based on domain knowledge, eg. it might be useful to multiply some data points that are related. Here we could for instance combine the data points of the upper and forearm to find the angle of the users elbow, which is expected to correlate with the depth of targets.
- Automated feature engineering. Since feature engineering can be very time consuming, but also is very important for successfully building a machine learning model, it is desired to build a concept of automating this process. Typically this is done by selecting some set of operators, that is applied to the raw data [3, 15, 16]. This requires some domain knowledge, since it not always is meaningful and can be computationally costly to apply operators on all permutations of raw data.

In the transformation phase features are computed, partly with some operations that are reused. Here these shared operations are listed here for clarity:

- Angles in triangle. Using the law of cosines, an angle in a triangle can be computed, given the three sides[3]:

$$angle\left(a,b,c\right) = cos^{-1}\left(\frac{b^2+c^2-a^2}{2bc}\right) \tag{3}$$

  where $angle$ is computes the angle $\angle A$ in the triangle ΔABC, consisting of the sides between vertices A and B, with length c, vertices B and C, with length a and vertices A and C, with length b. $\angle A$ is the angle opposing the side between B and C.
- Shoulder height. The shoulder height is computed from the right shoulder marker position and the collected distance measure, that describes the vertical distance from the marker to the shoulder. To make the measure more reliable, we take the mean of the three collected vertical shoulder marker positions from the calibration.

$$shoulderheight\left(e\right) = p_e\left[RightShoulderVerticalMarkerDistance\right] + \langle c_e\left[RightShoulder_y\right]\rangle \tag{4}$$

  where e is an endpoint, c_e are all calibrations by participant p_e and $\langle c_e\left[i\right]\rangle$ is the mean of the field i across all calibrations in c_e.

In the following we will describe the constructed features, in addition to give a plot of the relevant values and a formula for computation.

5.3.1 Index finger position. Even though humans are able to move their index finger independently of the hand, they tend to stretch it when pointing at specific objects [14]. This tendency results in a strong correlation between index finger and hand positions. Also the wrist is freely movable, but this is seemingly not been taken advantage of by the participants, which makes the hand and forearm correlate strongly (see figure 9). Also the upper arm correlates to the forearm, although less strongly. This is because there seems to be a difference in angles of the elbow, that are not correlated to the positions of fore- and upper arm. From these observations we can say that including all these positions, without transformation, would introduce redundancy in the data and might then, in the modelling phase, weight some of the features inappropriately. Of the mentioned values only the positional values of the index finger are directly included in the feature set.

- Horizontal position. The index finger horizontal position feature is computed using the following formula:

$$f\left(e\right) = e\left[IndexFinger_x\right] \tag{5}$$

  where e is an endpoint. Figure 11 shows boxplots of the horizontal position of the index finger, grouped by target position. We can see that there is a correlation between the horizontal value of the target and the horizontal value of the index finger. Also, the plot shows a small correlation between target depth and the horizontal index finger position, although this effect is not consistent for targets with position `(0, y, z)` and for other targets it is hard to determine whether the target z-value increases or decreases, when the horizontal index finger position increases.
- Vertical position. The index finger vertical position feature is computed using the following formula:

$$f\left(e\right) = e\left[IndexFinger_y\right] \tag{6}$$

  where e is an endpoint. In figure 12 are boxplots of the y-value of the index finger depicted. Here we can see a correlation between the vertical index finger position and the y- and z-values of the target. There is also some correlation between this feature and the depth of the target, but it is hard to say whether the value in- or decreases, when the vertical index finger position increases. What we

[3] https://en.wikipedia.org/wiki/Solution_of_triangles#Three_sides_given_(SSS) *(accessed: 24.01.20)*

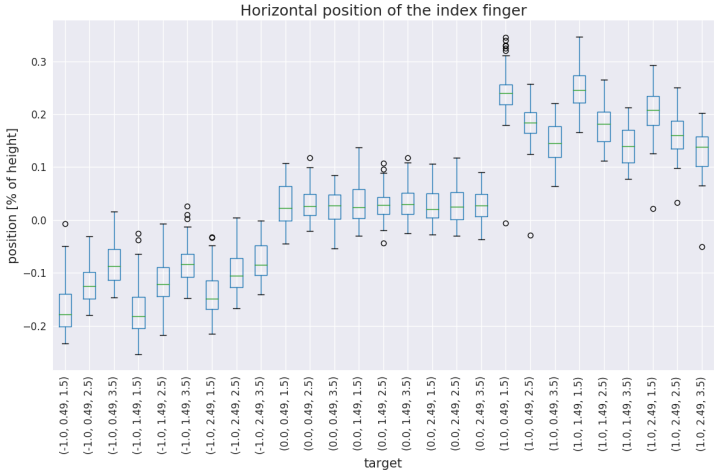


Fig. 11. Boxplot of the horizontal index finger position, grouped by target position. 0 is the vertical center of the participant.

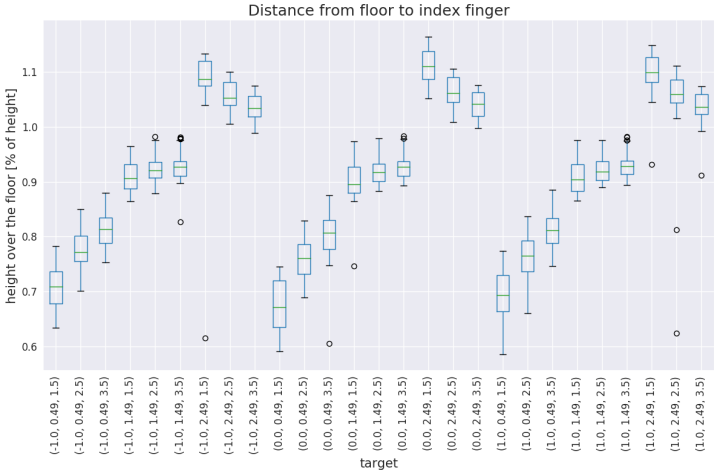


Fig. 12. Boxplot of the vertical index finger position, grouped by target position. 0 is the floor level, at the feet of the participant.

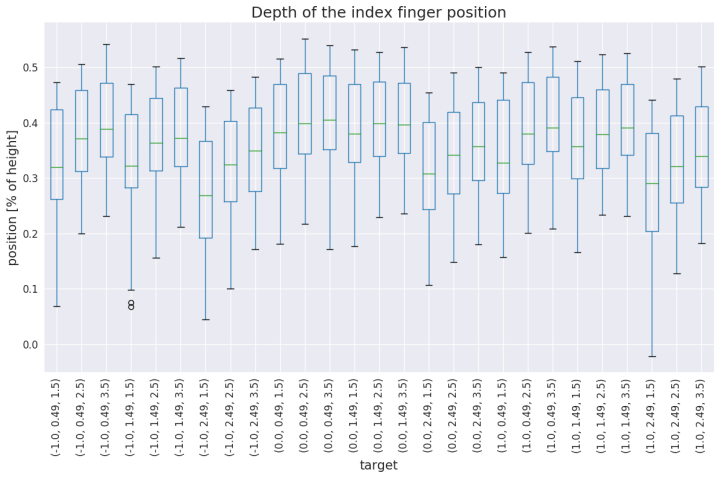


Fig. 13. Boxplot of the depth of index finger position, grouped by target position. 0 is the horizontal center of the participant.

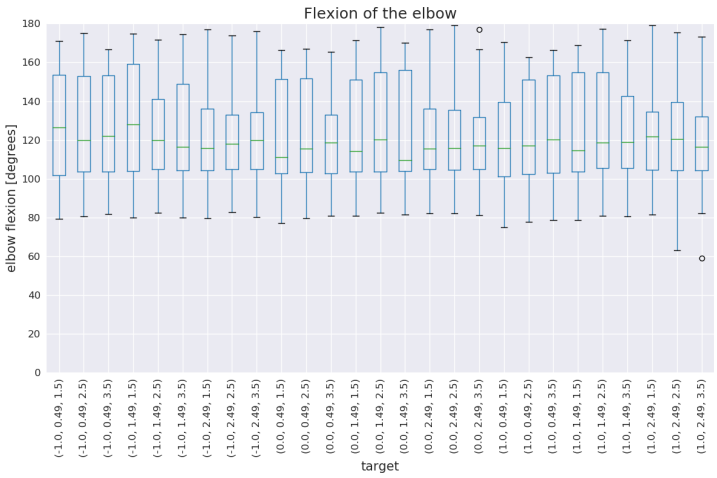


Fig. 14. Boxplot of the elbow flexion, grouped by target.

can say is that as the vertical index finger position increases, the depth increases for targets with coordinates $(x,\ 0.49,\ z)$ and that the depth decreases for targets with coordinates $(x,\ 0.49,\ z)$.

- Depth. The index finger depth feature is computed using the following formula:

$$f\left(e\right) = e\left[IndexFinger_z\right] \tag{7}$$

where e is an endpoint. Figure 13 shows boxplot of the presented feature. The data is spread much more than before and we can see different trend: Even thought the median of the data points for targets with coordinates $(-1,\ y,\ z)$, $(0,\ 2.49,\ z)$ and $(1,\ y,\ z)$ still correlates well with the z-values of the index finger, we now see that targets with coordinates $(0,\ 0.49,\ z)$ and $(0,\ 1.49,\ z)$ are much harder to seperate, looking only at the index finger z-value.

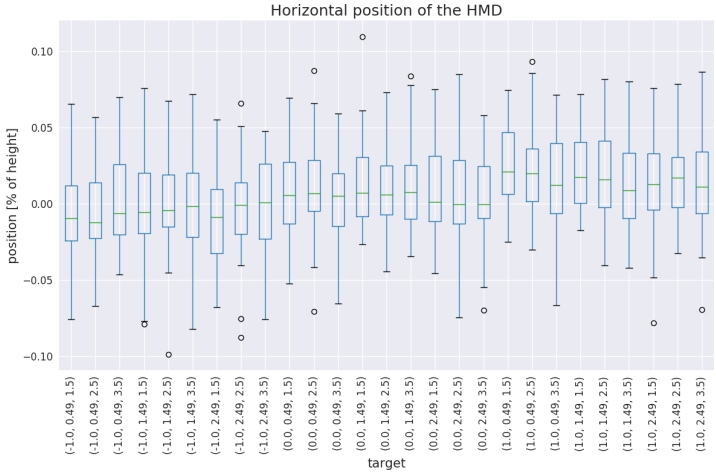


Fig. 15. Boxplot of the horizontal HMD position, grouped by target position. 0 is the vertical center of the participant.

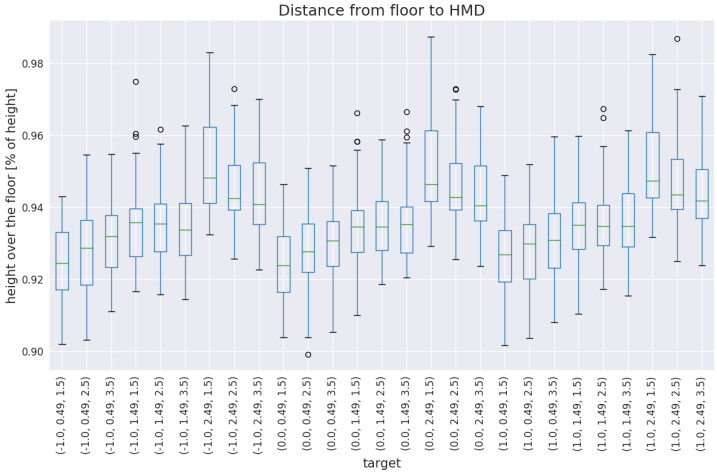


Fig. 16. Boxplot of the vertical HMD position, grouped by target position. 0 is the horizontal center of the participant.

As discussed before, do the positions of the index finger, hand and forearm correlate strongly, and thus compress the positional data of these markers into these three index finger related features.

5.3.2 HMD position. From the correlation matrix in figure 9 we can see some correlation between the target position and the HMD position, most prominent in the vertical HMD values.

- Horizontal position. The *HMD horizontal positional feature* is defined very similar to the index finger positional features:

$$f\left(e\right) = e\left[HMD_x\right] \tag{8}$$

  where e is an endpoint. This feature shows that even though there is some correlation between this feature and the horizontal target coordinate, it is not obvious where the correlation comes from, when looking at the boxplots in figure 15. If we take a closer look at the median, we can see slight differences across the different horizontal target positions (targets with coordinates $(-1,\ y,\ z)$, $(0,\ y,\ z)$ and $(1,\ y,\ z)$ respectively). In fact, is the median for targets with coordinates $(-1,\ y,\ z)$ around -0.005, $(-1,\ y,\ z)$ around 0.005 and $(-1,\ y,\ z)$ around 0.015, which seems to be enough to make the horizontal HMD position correlate with the horizontal target position.
- Vertical position. The *HMD vertical positional feature* is defined as:

$$f\left(e\right) = e\left[HMD_y\right] \tag{9}$$

  where e is an endpoint. Figure 16 shows boxplots of the vertical position of the HMD. We do not expect large changes, since the participants where asked to stand relatively still in the collection phase. The changes would be large if for instance the participant jumped or ducked down. The boxplots show that the medians of the vertical position are around 92% and 95% of the participants height. But we can see a similar trend, as we saw in the vertical index finger feature, as the vertical HMD position grows, the vertical target grows.

5.3.3 Orientations. In addition to positional data, orientational data was collected. These values are interesting as features because they express information about ie. the approximate gaze direction (HMD orientation or differences in hand orientation, that are correlate with pointing at targets in different positions. Figure 10 shows a correlation matrix between the raw orientation data and the target position. Generally, there is less correlation between these values and raw positional data, but there seems to

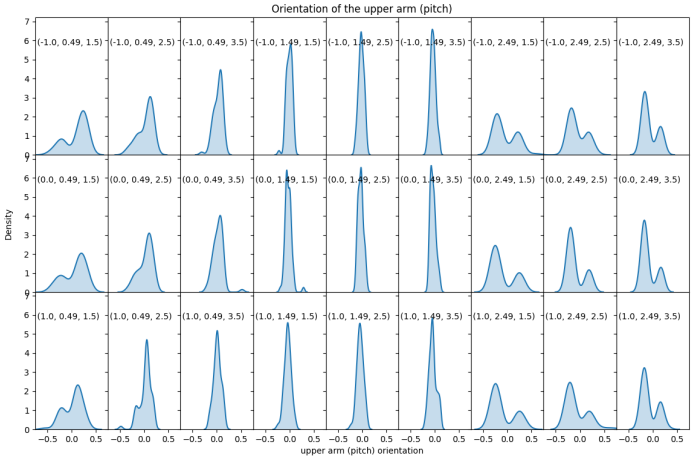


Fig. 17. Density plot of the upper arm pitch, grouped by target position.

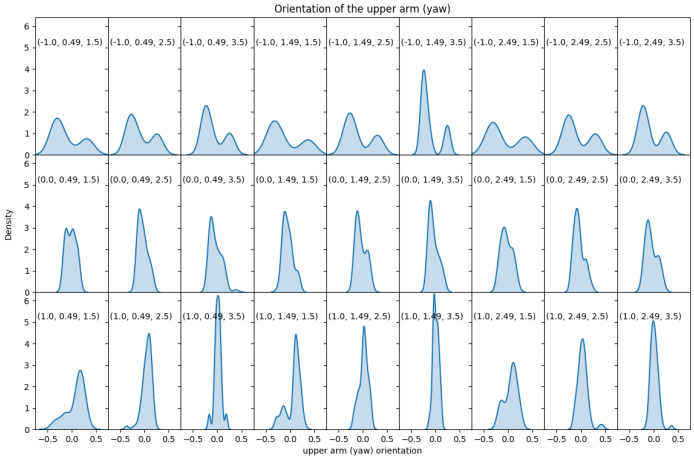


Fig. 18. Density plot of the upper arm yaw, grouped by target.

be some correlation between the upper arm orientation and some target positional values. We can use this information to build features build on this orientational data. Additionally, we will use our domain knowledge to construct features based on the HMD orientation.

- Upper arm pitch. The *upper arm pitch* is simply the raw horizontal orientation data of the upper arm.

$$f\left(e\right) = e\left[UpperArmOrientation_x\right] \tag{10}$$

  where e is an endpoint. Figure 17 shows a density plot of the upper arm pitch. We can see here that the upper arm yaw generally around 0 when pointing at targets with coordinates $(x,\ 1.49,\ z)$, above 0 when pointing at targets with coordinates $(x,\ 0.49,\ z)$ and below 0 when pointing at targets with coordinates $(x,\ 2.49,\ z)$. This fits the before mentioned correlation between the horizontal orientation and the vertical value spotted in the correlation matrix (figure 10).
- Upper arm yaw. We define the *upper arm yaw* as:

$$f\left(e\right) = e\left[UpperArmOrientation_y\right] \tag{11}$$

  where e is an endpoint. The yaw of the upper arm has been seen to correlate with the horizontal target position in the correlation matrix (figure 10) and this seems to be confirmed in figure 18, which shows a density plot of the yaw, grouped by target. We can see that the absolute orientation value generally is higher when pointing at targets with positions $(-1,\ y,\ z)$. This is expected since the right upper arm is turned towards the target.
- HMD pitch. The *pitch of the HMD* describes how much up or down the participant looks. We compute this by the formula:

$$f\left(e\right) = e\left[HMDOrientation_x\right] \tag{12}$$

  where e is an endpoint. The value is expected to grow larger as the target vertical value grows, but as the correlation matrix in figure 10 suggests is there no larger effect to be seen, apart from the fact that pitches generated while pointing at targets with coordinates $(-1,\ y,\ z)$ are generally closer to 0 than when pointing at other targets.

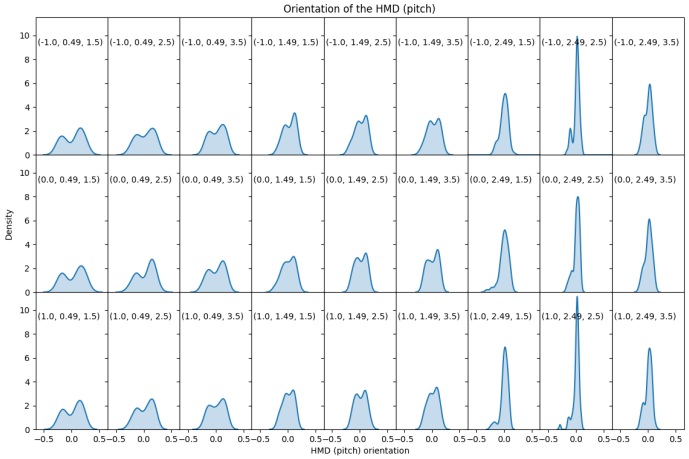


Fig. 19. Density plot of the HMD pitch, grouped by target.

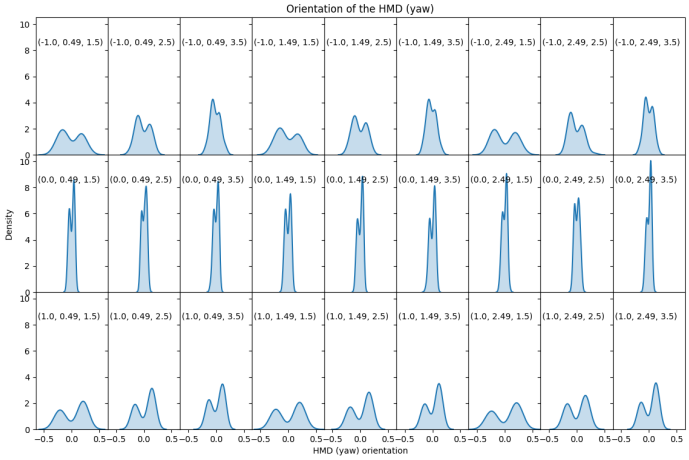
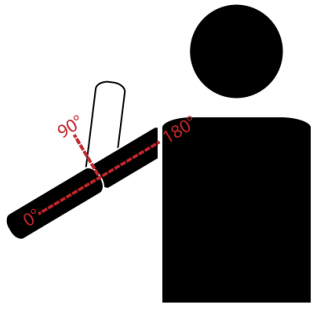
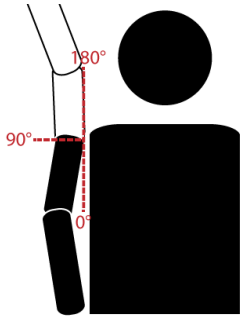


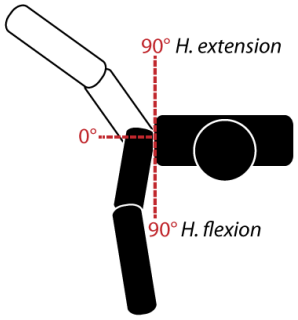
Fig. 20. Density plot of the HMD yaw, grouped by target position.



(a) Definition of elbow flexion.



(b) Definition of shoulder abduction.



(c) Definition of shoulder horizontal flexion and extension.

Fig. 21. Definition of elbow and shoulder angles. Figure and definition inspired by Anz et al. [1].

- HMD yaw. The *HMD yaw* describes the horizontal direction the participant is looking at, at pointing time. We define the yaw as:

$$f\left(e\right) = e\left[HMDOrientation_y\right] \tag{13}$$

where e is an endpoint. The density plot in figure 20 shows what was expected by the domain knowledge: the participant looks to the side when pointing at targets with coordinates $(-1,\ y,\ z)$ and $(1,\ y,\ z)$, while looking straight while pointing at targets with coordinates $(0,\ y,\ z)$.

5.3.4 Elbow flexion. The *elbow flexion* feature is based on the idea that the angle in the elbow gets smaller when humans point at distant targets, ie. humans stretch their arms when pointing at far targets. The aim of this feature is to describe the depth of the targets. Anz et al. [1] defines the elbow flexion as shown in figure 21a, although in a different context. This features shows the angle in the elbow and, in this context, is computed using the following formula:

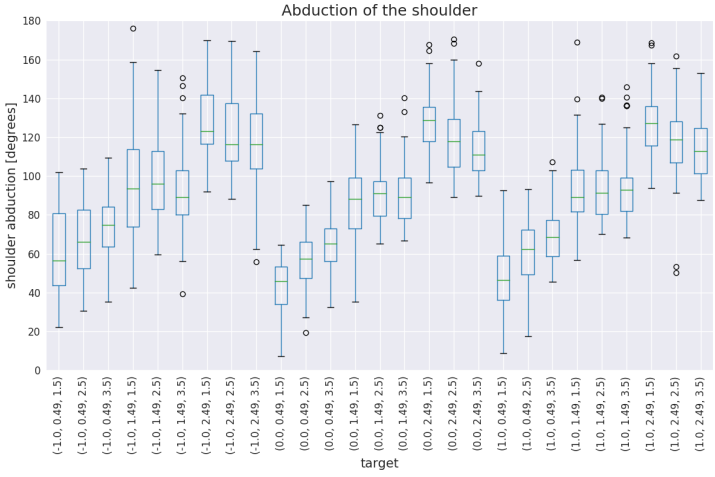


Fig. 22. Boxplot of the depth of the shoulder abduction, grouped by target position.

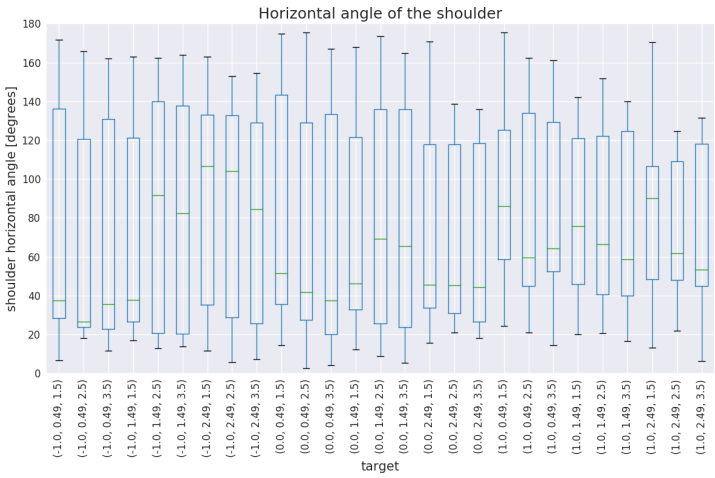


Fig. 23. Boxplot of the shoulder horizontal angle, grouped by target.

$$
\begin{aligned}
a_e &= d\left(e\left[Forearm\right], e\left[UpperArm\right]\right) \\
b_e &= p_e\left[UpperArmMarkerDistance\right] \\
c_e &= p_e\left[ForearmMarkerDistance\right] \\
f\left(e\right) &= angle\left(a_e, b_e, c_e\right)
\end{aligned} \tag{14}
$$

where e is an endpoint, p is the participant that produced the endpoint and $d\left(a, b\right)$ is the euclidean distance between a and b. The idea behind this is to convert the task of finding the elbow flexion to a trigonometry problem, where we know the lengths of all three sides of a triangle and want to compute the angle in a relevant vertex. Here we define the three vertices as the forearm position, the upper arm position and the elbow position. The goal is to compute the angle of the elbow vertex. Although we do not know the elbow position, we know the distance between forearm and elbow and the distance between upper arm and elbow, from the collected participant measures. By computing the distance between forearm and upper arm from the marker set positions, we know the length of all three sides of the triangle and can thus compute the angle in the elbow vertex. Figure 14 shows the resulting elbow flexion angles, grouped by targets. This figure shows that the flexion of the elbow does not show much about the target position, contrary to initial assumptions.

5.3.5 Shoulder abduction. This feature describes the *shoulder abduction* of the participant while pointing. The idea behind this features is that humans raise their arm more, when pointing at high targets, compared to low targets. Shoulder abduction is defined as shown in figure 21b and describes the vertical angle in the shoulder. The abduction is computed as follows:

$$
\begin{aligned}
a_e &= d\left(e\left[UpperArm_x, UpperArm_y\right], cal_e^1\left[UpperArm_x, UpperArm_z\right]\right) \\
b_e &= d\left(e\left[RightShoulder_x, RightShoulder_y\right], cal_e^1\left[UpperArm_x, UpperArm_y\right]\right) \\
c_e &= p_e\left[UpperArmLength\right] - p_e\left[UpperArmMarkerDistance\right] \\
f\left(e\right) &= angle\left(a_e, b_e, c_e\right)
\end{aligned} \tag{15}
$$

where e is an endpoint, $d\left(a,b\right)$ is the euclidean distance between a and b, p_e is the participant that produced the endpoint and cal_e is the calibration produced by the participant. Similar to the elbow flexion, shoulder abduction is computed by finding the angle of a certain vertex in a triangle. The construction of the triangle depends on two observations:

(1) Dimensionality reduction of the vertices. From the definition of shoulder abduction, we know that the abduction angle only is computed on the the x- and y-values of the vertex positions, since it only describes the vertical abduction of the shoulder.
(2) Calibration vertex. By the definition, we know that the abduction should be 0° when the arm is pointing straight down along the body. We can thus use the upper arm position from the first calibration task (figure 5a) as a vertex for the triangle.

Then, to construct the triangle we use the upper arm position from the first calibration task, the position upper arm in the endpoint and the right shoulder position as the three vertices. From this we can compute the angle of the right shoulder vertex, since we can compute the distances between the three vertices.

Figure 22 shows boxplots of the shoulder abduction, grouped by targets. The plots show a similar effect as the index finger verticality feature, which might be expected, since these two features should correlate by the initial idea. We can see that the initial intuition about this feature is right, since the angle is larger when pointing at high targets, compared to lower targets.

5.3.6 Horizontal shoulder angle. The *horizontal shoulder angle* describes the horizontal change in the arm position. The idea is, that humans positions their arm further to the left for targets to the left, compared to right targets and since the arm is attached to the shoulder, should the angle be larger for targets with coordinates `(-1, y, z)` compared to targets with coordinates `(1, y, z)`. The horizontal shoulder angle is computed by the definition shown in figure 21c.

$$\begin{aligned} a_e &= d\left(e\left[UpperArm_x, UpperArm_z\right], cal_e^3\left[UpperArm_x, UpperArm_z\right]\right) \\ b_e &= d\left(e\left[RightShoulder_x, RightShoulder_z\right], cal_e^3\left[UpperArm_x, UpperArm_z\right]\right) \\ c_e &= p_e\left[UpperArmLength\right] - p_e\left[UpperArmMarkerDistance\right] \\ f\left(e\right) &= angle\left(a_e, b_e, c_e\right) \end{aligned} \tag{16}$$

where e is an endpoint, $d\left(a,b\right)$ is the euclidean distance between a and b, p_e is the participant that produced the endpoint and cal_e is the calibration produced by the participant. The horizontal shoulder angle is computed very similarly as the shoulder abduction, but drops the y-values instead of the z-values and the third calibration is used to construct a vertex for the triangle.

In figure 23 boxplots for the horizontal shoulder angle, grouped by target, are shown. We see no correlation between the target position and this feature. We theorize that this might be because the right shoulder marker tends to move, when the shoulder is not moved due to the attachment on the strap of a backpack, such that the marker does not provide reliable horizontal information.

5.3.7 Index finger above head. For the *index finger above head* feature we make use of the binning technique. This means, that we construct a feature that summarizes a range of data into smaller bins. In this case we summarize the vertical position into two bins: (1) if the vertical index finger position is above the participant height and (0) otherwise:

$$f\left(e\right) = \begin{cases} 1, & \text{if } e\left[IndexFinger_y\right] \geq p_e\left[height\right] \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

where e is an endpoint and p_e the participant that produced that endpoint. The plot in figure 24 show a count of this feature, grouped by target. The plot shows a correlation between the vertical values of the target and this feature. Note that the plot also shows that this feature does not separate between middle and lower targets, but is a strong indicator towards whether the target has coordinates `(x, 2.5, z)` or

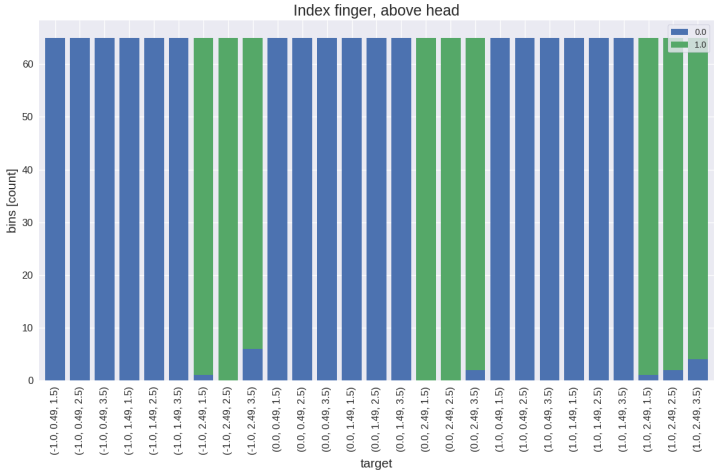


Fig. 24. Bar graph of a binary indicator, that describes whether the index finger is vertically positioned above the participants head. 0 means false, 1 means true.

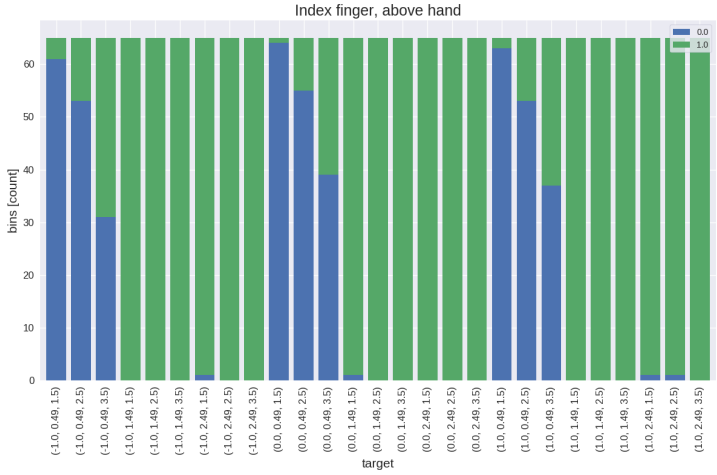


Fig. 25. Bar graph of a binary indicator, that describes whether the index finger is vertically positioned above the hand. 0 means false, 1 means true.

not. We theorize that humans move their index finger above their head, when pointing at targets that are (sufficiently) above their head.

5.3.8 Index finger above hand. Similarly to the mentioned index finger above head feature, is the *index finger above hand* feature constructed by binning. It becomes (1) if the index finger is vertically positioned above the hand and (0) otherwise:

$$f\left(e\right) = \begin{cases} 1, & \text{if } e\left[IndexFinger_y\right] \geq e\left[Hand_y\right] \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

where e is an endpoint and p_e the participant that produced that endpoint. The plot in figure 25 show a count of this feature, grouped by target. The plot shows a some correlation between the vertical values of the target and this binary feature. Similar to the above head feature, does this feature not distinguish well between middle and high targets, but serves as a strong indicator for low targets.

5.3.9 Index finger position relative to body. Figure 9 show a correlation between target x, y and (less so) z values and their corresponding index finger values. For the following features, we want to use binning to put an emphasis on these correlations. As discussed before, will we loose some information, e.g. the correlation between target y value and index finger z value, by doing this binning.

- Index finger horizontal position relative to body. Also the *relative horizontal index finger position* is constructed using binning. This feature categorizes the horizontal position of the index finger into three bins: (2) if the index finger is to the right of the right shoulder, (1) if the index finger is to the left of the left shoulder and (0) if the index finger is between the shoulders:

$$f\left(e\right) = \begin{cases} 2, & \text{if } e\left[IndexFinger_x\right] > p_e\left[RightShoulder_x\right] \\ 1, & \text{if } e\left[IndexFinger_x\right] < p_e\left[LeftShoulder_x\right] \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

where e is an endpoint and p_e the participant that produced that endpoint. Figure 26 shows a count of the bins, grouped by targets. There is a strong correlation between the horizontal target position and the horizontal index finger bins.

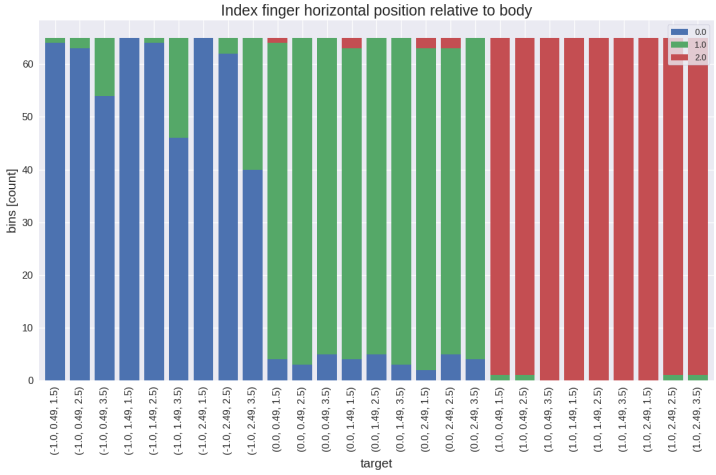


Fig. 26. Bar graph of the horizontal index finger category, grouped by target position.

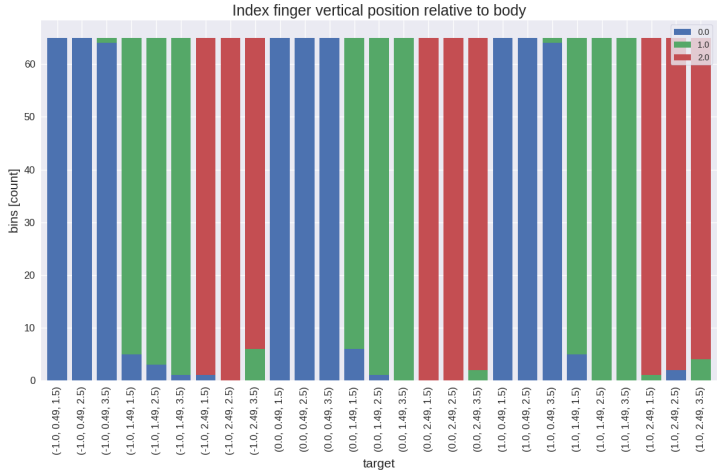


Fig. 27. Bar graph of the vertical index finger category, grouped by target.

- Index finger vertical position relative to body. The *relative vertical index finger position* feature describes the vertical position in terms of bins:

$$f\left(e\right) = \begin{cases} 2, & \text{if } e\left[IndexFinger_y\right] \geq p_e\left[height\right] \\ 1, & \text{if } e\left[IndexFinger_y\right] \geq shoulderheight\left(e\right) \text{ and } e\left[IndexFinger_y\right] < p_e\left[height\right] \\ 0, & \text{if } e\left[IndexFinger_y\right] < shoulderheight\left(e\right) \end{cases} \tag{20}$$

  where e is an endpoint and p_e the participant that produced that endpoint. This feature categorizes the vertical index finger position into three separate bins: (2) if the index finger is above the participants height, (1) if the index finger is positioned between the participants height and shoulder height and (0) if the index finger is below the participants shoulder height. Figure 27 shows a count of the bins, grouped by targets. There is a correlation between the vertical target position and the vertical index finger bins.
- Index finger depth relative to body. To categorize the z value of the index finger position, we want to compare it to some known distance, similar to the categorization of the y value of the index finger. The known length here is the length of the arm.

$$wrist_e^z = e\left[RightShoulder_z\right] + p_e\left[UpperArmLength\right] + p_e\left[ForearmLength\right]$$
$$f\left(e\right) = \begin{cases} 1, & \text{if } e\left[IndexFinger_y\right] \geq \left(wrist_e^z + 10\right) \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

  where e is an endpoint and p_e the participant that produced that endpoint. $wrist_e^z$ computes the maximum reach of the participants wrist, by adding the arm length to the shoulder position of the endpoint e. Also here the idea is similar to the features computed above, since we categorize the depth of the index finger into two categories: (1) if the index finger is more than 10cm in front of the maximal wrist reach of the participant and (0) otherwise.

5.3.10 Distances. The closest we come to automated feature engineering is by applying the euclidean distance operator across all permutations of marker sets. For some pairs of marker sets this intuitively conforms with the idea that the arm is stretched more while pointing at distant targets and less stretched

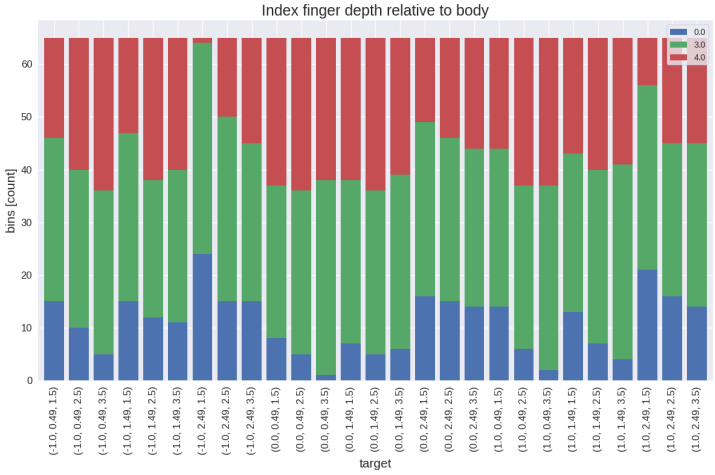


Fig. 28. Bar graph of the index finger depth category, grouped by target position.

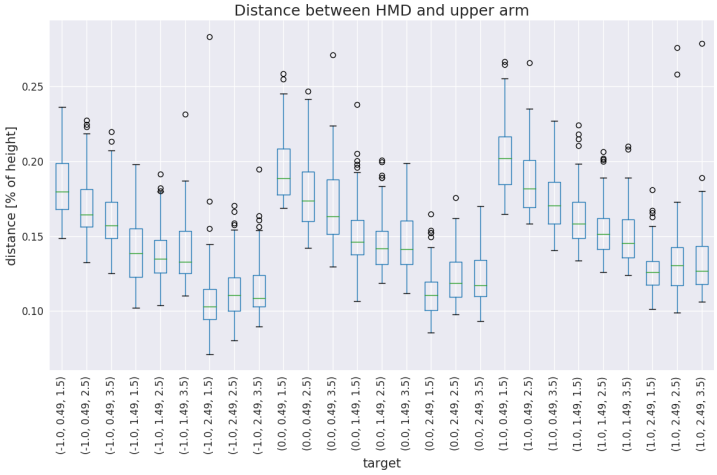


Fig. 29. Boxplot of the distance between HMD and the upper arm, grouped by target.

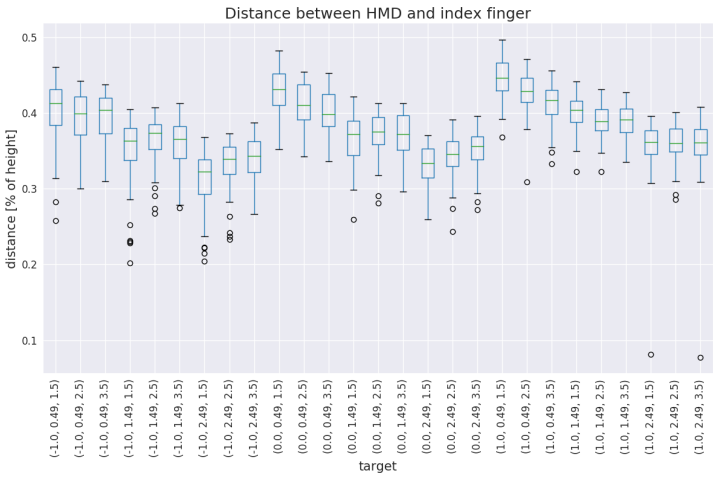


Fig. 30. Boxplot of the distance between HMD and the index finger, grouped by target position.

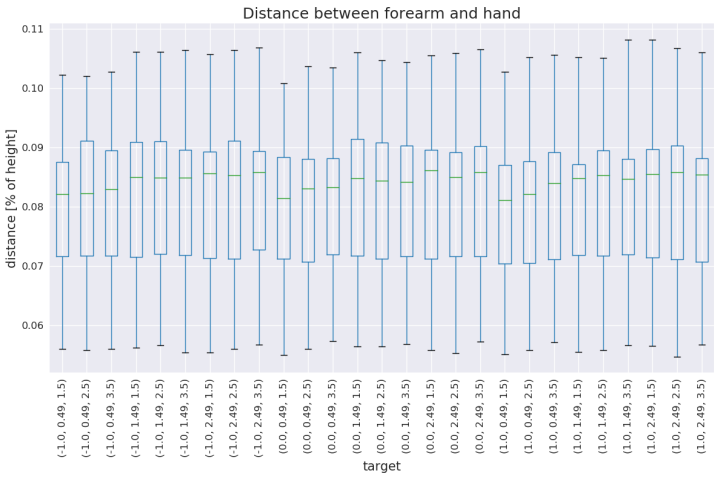


Fig. 31. Boxplot of the distance between the forearm and hand, grouped by target.

when pointing at close targets. In terms of distance between marker sets, this effect would be expected to be observed in the distance between HMD and index finger.

The *distance features* are generated using the formula:

$$f\left(e\right) = d\left(e\left[m\right], e\left[n\right]\right) \tag{22}$$

where e is an endpoint, m and n are two distinct marker sets and $d\left(a, b\right)$ is the euclidean distance between a and b. By applying the distance function on each unique pair of marker set positions we generate $\binom{7}{3} = 21$ features. Figures 29 to 31 show boxplots of the distances between HMD and upper arm, HMD and index finger and forearm and hand respectively, grouped by target.

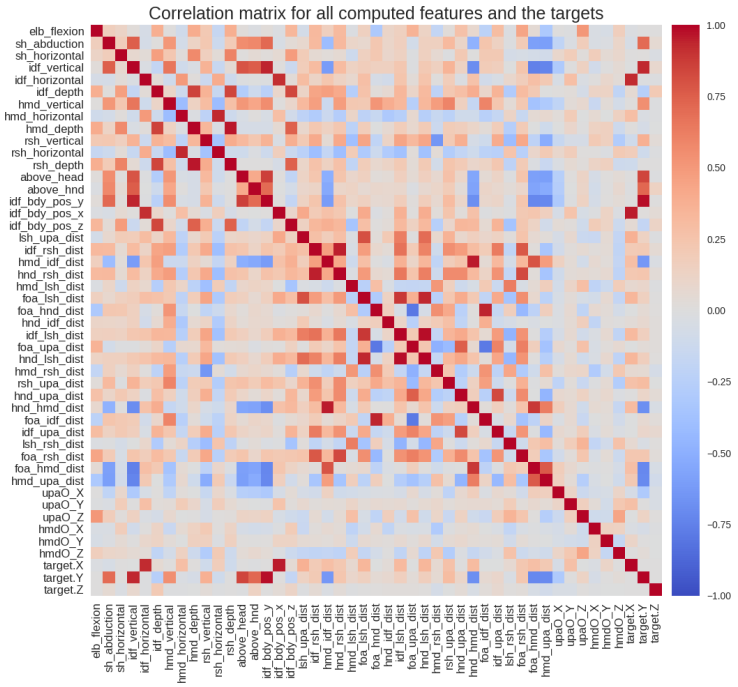


Fig. 32. Correlation matrix, showing the relationship between the constructed features and the target position. In the bottom left corner, we can see that the elbow flexion feature does not correlate to the target position, while the shoulder abduction has some correlation with the vertical target position.

The distances between HMD and both upper arm and index finger grow smaller when the verticality of the target grows larger, which is expected, since when the human arm raises, it naturally gets closer the human head. Interestingly, does the distance between HMD and index finger not, contrary to the theorization before, consistently grow larger when pointing at more distant targets. We can see this effect at targets with coordinates with coordinates $(x,\ 2.49,\ z)$, but for targets with coordinates $(x,\ 0.49,\ z)$ we see the opposite effect. This is quite natural, since the arm is attached to the shoulder and thus the index finger will be physically further away from the head when pointing at close low targets than far low targets and vice versa for high targets.

The last mentioned distance between forearm and hand is less interesting in terms of changes. This distance is not expected to change much during pointing, since the hand is naturally attached to the forearm in a fixed position. The variance in the data most likely is due to changes in wrist angle and due to change in physical placement of the marker sets on the participant.

5.4 Selection

In the selection phase, we support the iterative process of engineering by selecting new features from computed in the transformation phase. This phase consists of selecting features that are interesting, for modelling the human pointing movement. This is done by analysing the features with statistical methods, such as correlation and a χ^2-test. The selection of individual features is also based on the performance of the machine learning model, since that gives an indicator for how well the machine understands a feature. Additionally, when building a model for human pointing we can use this selection phase to determine which properties of the human pointing movement is important for consistently modelling pointing.

Feature selection techniques evaluate the rank of a feature. Based on this rank features can be filtered, such that irrelevant features are not used for machine learning. Benefits of feature selection include better accuracy, less feature redundancy and reduced overfitting.

- Correlation. Pearson's correlation computes the linear correlation between two features, telling us how the first feature behaves when the second grows larger or smaller, and vice versa. Correlation value ranges from 1 to -1, describing the strength of positive and negative correlation.
  For feature selection we can use correlation to determine the rank of a feature. In this case we have three target values (x, y and z positions of the target), for which each feature has some correlation. A correlation matrix for all features and the targets is shown in figure 32. From this matrix, we can see that some features have strong correlation with some positional values (eg. the vertical index finger position), while other features have no correlation at all (eg. elbow flexion).
- Statistical methods. Other statistical methods for feature selection include χ^2-tests and mutual information. These tests show relationships between the features and the target positions. The difference between these two methods is that one is usable only for classification problems, while the other is used for regression problems.
  For the purpose of applying these two methods when selecting features relevant for modelling human pointing, we encode the targets as classes for the χ^2-test and leave them continuous for mutual information, but sum the mutual information value across the three target positions. The strength of the relationship can be used to select features. Plots for these methods, applied to the engineered features are shown in appendix figures B.1 and B.3.
- Feature importance. Tree-based ensemble machine learning models can be used to rank features, since they compute the feature importance, while training. The importance is typically based on how much variance a feature contributes to the relevant true label/value, similar to how PCAs work. A plot of the feature importance can also be found in appendix figure B.2.
- Iterative model evaluation. The simplest way to select features is to iteratively select a subset of features and use a machine learning model to evaluate the set of features. This method is somewhat unreliable, also because it is computationally difficult to try all combinations of features.

5.4.1 Selected features. We select models based on the mentioned ranks. The different methods yield different rankings of the features, but a combination of these rankings is used to construct a list of candidate features. Table 5 shows an overview of all described features and whether they are selected or not. Note for instance the elbow flexion, which intuitively provides a lot of information about the depth of the target, is not selected, since it generally scores very low rankings among the different selection techniques. We can also see that most distance features are included and contribute to the model.

5.5 Modelling

We model human pointing using machine learning techniques. We use machine learning for different reasons:

(1) Evaluation. To construct a model for human pointing, we need to asses constructed features in an unbiased manner. We can use a machine learning model to evaluate a set of features and, based on the models performance, exclude irrelevant features.
(2) Expressive power. Machine learning is based on features that describe the underlying problem. These features need to be well constructed such that the machine learning algorithm performs its prediction on a solid basis. This means that if the machine learning model performs well, we can say that the underlying features are constructed well and can tell us something about the characteristics of human pointing.
(3) Prediction. With the selected features, we can predict the target of some new, previously unseen, pointing movement. This opens up for a wide range of applications, for instance a selection techniques, but also for many new questions. These concerns and opportunities will be discussed in section 7.

Feature	Selected
Horizontal index finger position	✓
Vertical index finger position	✓
Index finger depth	✓
Horizontal HMD position	✓
Vertical HMD position	✓
HMD depth	✓
Upper arm pitch	✗
Upper arm yaw	✗
Upper arm roll	✗
HMD pitch	✗
HMD yaw	✗
HMD roll	✗
Elbow flexion	✗
Shoulder abduction	✓
Horizontal shoulder flexion	✓
Index finger above head	✓
Index finger above hand	✓
Index finger horizontal position relative to body	✓
Index finger vertical position relative to body	✓
Index finger depth relative to body	✗

Feature	Selected
Distance between index finger and upper arm	✓
Distance between index finger and the right shoulder	✓
Distance between index finger and the left shoulder	✓
Distance between the left shoulder and upper arm	✓
Distance between the left shoulder and the right shoulder	✗
Distance between the right shoulder and upper arm	✓
Distance between hand and index finger	✗
Distance between hand and upper arm	✓
Distance between hand and the left shoulder	✓
Distance between hand and the right shoulder	✓
Distance between hand and HMD	✓
Distance between forearm and the left shoulder	✓
Distance between forearm and the right shoulder	✓
Distance between forearm and HMD	✓
Distance between forearm and hand	✗
Distance between forearm and index finger	✓
Distance between forearm and upper arm	✓
Distance between HMD and the left shoulder	✗
Distance between HMD and the right shoulder	✓
Distance between HMD and index finger	✓
Distance between HMD and upper arm	✓

Table 5. Overview of the selected features.

Generally are machine learning algorithms divided in different categories based on their output. There are classification algorithms, which can predict the categories of objects. The input objects can be a wide variety of things, for instance numeric descriptions of flowers and the output is some predicted class of the object, such as the name of the flower. In this application, we can use a classifier to predict which of the 27 different targets a participant pointed at. Another category of machine learning algorithms is called regression algorithms, which output a continuous prediction. This is useful eg. when predicting the outside temperature, based on previous weather data. In this case it is useful for predicting the x, y and z values of the desired target.

5.5.1 Pipeline. The machine learning part of this work is implemented in Python 3.7.4, using the `scikit-learn`[4] machine learning library (version 0.22.1). The general pipelines for classification and regression are the same, apart from the machine learning models.

[4] https://scikit-learn.org/stable/ (*accessed: 27.01.20*)

Model	Cross validation accuracy	Test accuracy	Test F1 score
Naive Bayes	40.10%	40.17%	0.3406
KNN	53.49%	60.97%	0.6157
SVM	**86.18%**	**87.46%**	**0.8680**
Random Forest	71.58%	72.93%	0.7257

Table 6. Overview of classification accuracies for different models. The best performing model and its scores are highlighted in bold.

Before training, we want to tune a machine learning models parameters, in order to construct better results. The tuning is done by gridsearching over some candidate parameters and then selecting the best performing set of parameters.

Since there is no dedicated training and testing dataset, we will have to asses the performance of a given model by splitting the set of constructed dataset into train and test sets. After the split the training set consists of 80% and the test set of 20% of the original dataset. This is done to ensure that the machine learning model has not seen some portion of the data before, which we then can use to asses the model.

The train set is used to train the machine learning model. We test the model twice: first on itself using 5-fold cross validation and secondly against the test set. The validation score it then used to compare to the test score to ensure that the model is not overfitting, which would be the case if the validation score is much better than the test score. The test score is also used to compare the models and then selecting the best among them.

5.5.2 Classification. For classification, the target positions of the training data are encoded as 27 distinct classes. Then four different machine learning models are trained on the data and compared in terms of accuracy and F1 score. We select the four models (1) *Naive Bayes* (NB), for establishing a baseline, (2) *K-Nearest Neighbors* (KNN), since it is a conceptually simple model, (3) *Support Vector Machine* (SVM) and (4) *Random Forest* (RF), for their good performance on other classification tasks. All these models are linear, meaning that they are less complex than for instance a neural network. Neural networks have a lot of parameters to be tuned and also need more data than linear models to perform well. Our dataset is relatively small and thus a linear model is most likely better suited for this task. NB is used as a baseline, since the model itself predicts classes based on a probabilistic guess.

The results of a comparison between the four models is displayed in table 6. Of the four models, the SVM performs best both in terms of validation and test accuracy. With a test accuracy of 87.46%, the model is relatively accurate, which suggests that the selected features work well for as a basis for the human pointing model. The SVM is using a regularization parameter (C) of 20000, an RBF kernel and a kernel coefficient ($gamma$) of 0.001, found by a grid search. See appendix C.1 for a list of all parameters used for comparing models.

The confusion matrix in figure 33 shows us the errors typical this machine learning model makes when predicting targets. A confusion matrix shows how many targets where predicted as some class versus the true class. If all predictions are true, the confusion matrix will be sparse apart from the diagonal and thus all counts not on the diagonal represent an error. Note here that the targets are arranged in a left to right, low to high and close to far manner. When looking the the overall confusion matrix, we see that when the model makes prediction errors, it typically predicts targets to be either closer or further than

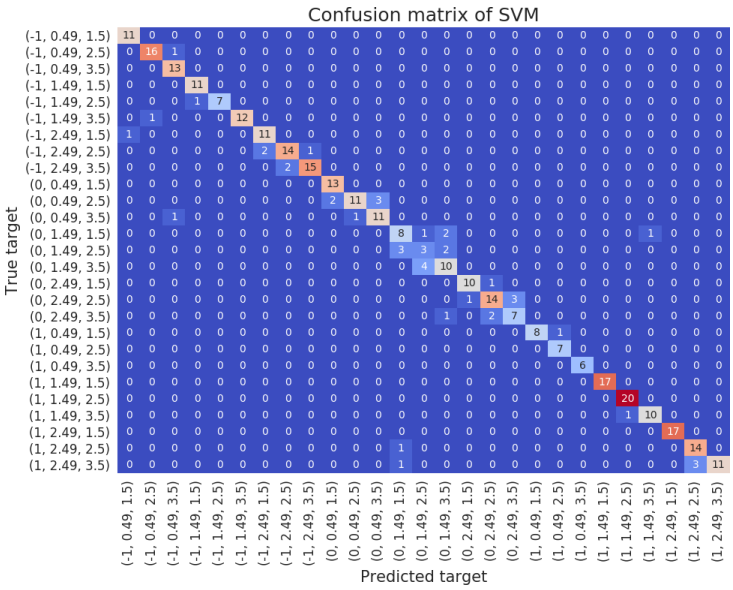


Fig. 33. Confusion matrix of the predictions made on the test set with a SVM. True predictions are counted on the diagonal, while all other counts indicate errors.

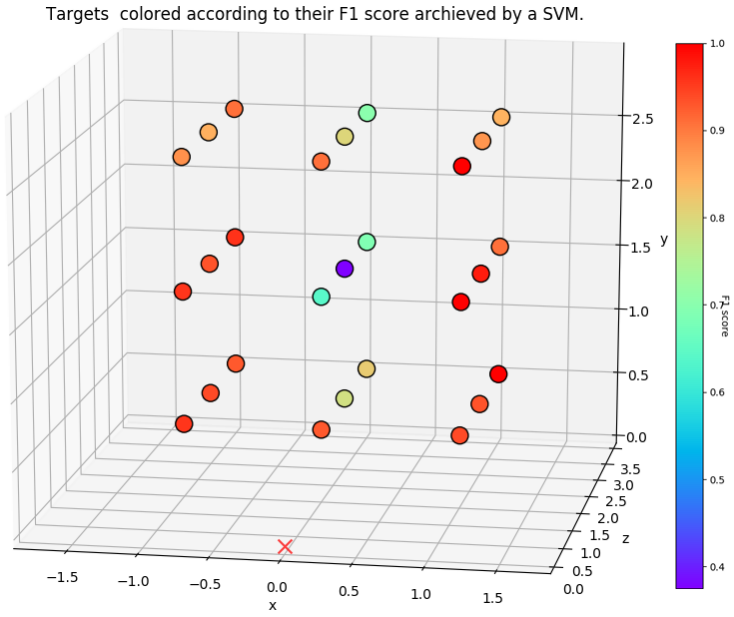


Fig. 34. The target grid, where the colors of the target correspond to their mean F1 score (using SVM), for all folds. The red cross is the participant starting position. Here we can see that the center aisle of targets produces a lower F1 score than other targets.

actual \ predicted	(0, 0.49, 1.5)	(0, 0.49, 2.5)	(0, 0.49, 3.5)
(0, 0.49, 1.5)	13	0	0
(0, 0.49, 2.5)	2	11	3
(0, 0.49, 3.5)	0	1	11

Table 7. Zoom-in on the confusion matrix shown in figure 33 on targets with coodinates `(0, 0.49, z)`.

actual \ predicted	(0, 1.49, 1.5)	(0, 1.49, 2.5)	(0, 1.49, 3.5)
(0, 1.49, 1.5)	8	1	2
(0, 1.49, 2.5)	3	3	2
(0, 1.49, 3.5)	0	4	10

Table 8. Zoom-in on the confusion matrix shown in figure 33 on targets with coodinates `(0, 1.49, z)`.

they should be, meaning that the machine learning model rarely predicts targets outside the right aisle (~15.5% of the times the model is wrong).

The arrangement of targets means that the confusion matrix can be analysed aisle by aisle in the target grid. We can then zoom in on one aisle, for instance the third aisle in the second column (targets with `(0, 0.49, z)`) displayed in table 7. The matrix shows us that the machine learning model predicted `(0, 0.49, 1.5)` correctly 13 times, `(0, 0.49, 2.5)` and `(0, 0.49, 3.5)` both 11 times (the differences in total number of predictions are due to randomness when splitting the original dataset). In the slice we can observe the general trend that the center of an aisle (targets with coordinates $(x, y, 2.5)$) are the hardest to predict right, since they often get predicted either further or closer than the actual target. In this slice one third of the targets are predicted either further or closer in the center of the aisle.

Model	Cross validation RMSE	Test RMSE	Prediction distance [m]
Linear Regression	0.4491	0.4520	0.6715
KNN	0.4311	0.3985	0.5033
SVM	**0.2805**	**0.2543**	**0.2442**
Random Forest	0.2821	0.3107	0.3830

Table 9. Overview of regression scores for different models. The cross validation is obtained by 5-fold cross validation. The best performing model and its score are highlighted in bold.

Distance						
mean	*std*	*min*	*first quartile*	*median*	*third quartile*	*max*
0.2442	0.2436	0.0447	0.1259	0.1559	0.2844	1.2220

Table 10. Metrics of the distance between prediced and true targets. Note that the distance between the maximum value and the 75th percentile is rather large, indicating that the maximum value is an outlier, that impacts the mean. See appendix table C.3 for a full overview of all prediction distances. [m]

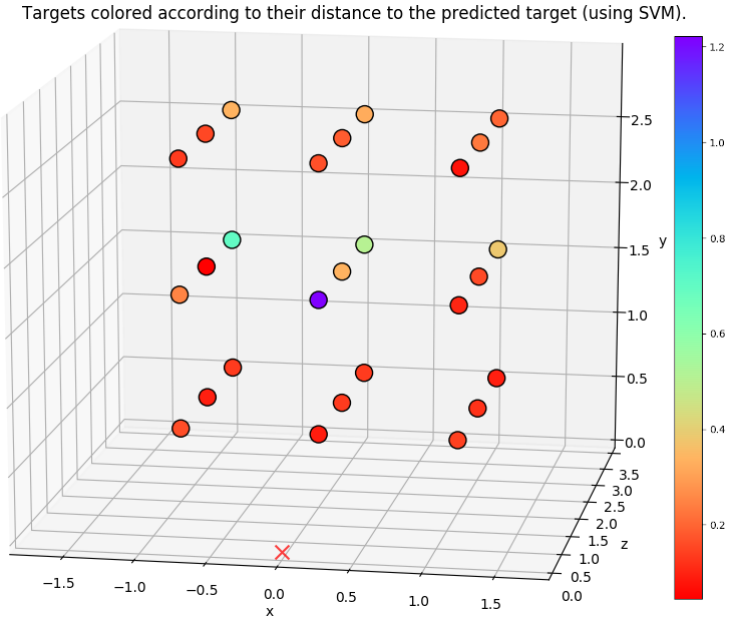


Fig. 35. The target grid, where the colors of the target correspond to their average distance to the predicted target (using SVM), for all folds. The red cross is the participant starting position. Clearly, the target closest to the participant is identified as an outlier. [m]

We can see this general trend across almost all aisles, with one larger outlier: the center aisle (targets with coordinates `(0, 1.49, z)`). The slice zooming in on these targets is displayed in table 8. There we can see that the machine learning model has a hard time to predict the depth right, especially in this center row. This row contributes with ~26.6% of the total false predictions and errors where the center target (`(0, 1.49, 2.5)`) either was falsely predicted or should have been predicted make out ~24.4% of the total error.

This means that we in general have two error sources: The distinction between different medium and far targets and the distinction between targets in the center row. We can see a similar result when looking at the F1 score. In figure 34 all targets are plotted according to their individual F1 score, ie. how good is the machine learning model at predicting this one target. The F1 scores show the same general picture as the confusion matrix: the model has a lower F1 score for deep targets and is especially bad at predicting targets in the center aisle.

The error sources are not unexpected, since there where fewer features that correlated with the depth of the targets in the constructed set of features.

5.5.3 Regression. For regression, we train a model for each target dimension, except when training a RF regressor, since it supports multi-output regression natively. Again, four models are trained: (1) *Linear Regression*, (2) KNN, (3) SVM and (4) RF. Table 9 shows the cross validation RMSE, the test RMSE and the mean distance between true and predicted target for each of the models. The RMSE score describes the standard deviation of the prediction error, meaning that a score of 0 would indicate a perfect fit.

Also for regression the SVM performs best. The SVM is trained using a regularization parameter (C) of 100, an RBF kernel and a kernel coefficient ($gamma$) of 0.01, found by a grid search. See appendix C.1 for all parameters used for comparing models.

The mean distance between the predicted and true targets is 24.42 cm. The distance measure is broke down in table 10. We can see that the 50% of the distances are smaller or equal to 15.59 cm. Already this indicates that the mean is dominated by outlier(s) and we can formalize this by computing outliers based on the interquartile range (IQR), such that if a value is larger than 1.5 times the IQR added to the third quartile, we say the value is an outlier. Thus in this case the maximum value that is not an outlier is $0.2844 + \big(1.5 * \big(0.2844 - 0.1259\big)\big) = 0.5222$, ie. a distance that is larger than 52.22 cm is an outlier. The appendix table C.3 shows that there are two targets, that are marked as outliers: targets with coordinates `(-1, 1.49, 3.5)` (69.89 cm) and `(0, 1.49, 1.5)` (122.20 cm). In the same table we can see that most of the distance is due to predicting the depth position of the targets wrongly. We observe this error especially when predicting the position of the target with coordinates `(0, 1.49, 1.5)` where we can see the that the depth on average is predicted as 2.72, which is closer to the next target in the same aisle, than the true target. Figure 35 shows the target grid, where targets are plotted with a color, that reflects the distance of the target. The before mentioned outliers can be clearly identified here in addition to an additional target that is close to be an outlier (the target with coordinates `(0, 1.49, 3.5)`).

Generally we can say that the model performs well, apart from the mentioned outliers. A median distance of 15.59 cm is acceptable for this simple model, considering a distance between targets of 100 cm and a target diameter of 15 cm. The target grid shows that it is harder to predict the deep targets correctly. We can in general see that the three bottom aisles have low error, while the error grows larger when looking at targets with medium and high height. Apart from the bottom aisles, we can also see that the error grows larger for targets with greater distance to the participant. The center aisle has an overall higher error that other aisles, suggesting that it is the hardest to predict correctly based on the used features.

5.5.4 Summary. We have constructed a model that works reasonably well on classification, but may need more iterations of the process to engineer more feature for regression. We know that both models have problems predicting the depth correct, especially in the center aisle.

These two error sources are very alike in both the classification model and regression model, which suggests that to improve the resulting features set, we needed to introduce some features that describe the differences between targets in the center aisle and some features that can describe target depth.

6 DISCUSSION

Initially we questioned whether we could model human pointing, such that it could be used as a target selection technique. We have shown that the human pointing can be modeled using machine learning and thus made the model usable for selection techniques, although the usability itself is yet to be tested.

6.1 Collecting the pointing movement

The data collection study was conducted in a controlled VR environment. Nearly all participants noted some fatigue after pointing at targets $5 * 27 = 135$ times in a row, leading to the hypothesis that we should not expect participants to point at more targets without introducing a bias. Some fatigue is justifiable, since we do expect the task to be unusual compared to a typical day in the life of a human being, but we should not strain the participant more than necessary. This study design decision is meant to balance between this exact fatigue issue and collecting data, more of which would make the resulting model more reliable. In the end the fatigue issue weighted more, since it is hard to adjust for such a bias, compared to inviting more participants.

Some participants noted different pointing techniques: One aimed the index finger with one eye closed "like a gun", another was rather impatient and went for fast paced pointing and a third noted that they could see two hands and did not know "which index finger to aim with". The first and the second comment is most likely due to the horizontal seperation of the eyes (binocular disparity), which is a natural effect when a human focuses their eyes on a not too distant object (in this case: the index finger). In all cases, the participants where asked to continue pointing as they wanted, since the goal for the data collection study was to collect data on the natural human pointing. Additionally we did observe that participants almost wanted to reach for and touch targets when they appeared within an arms reach from them. From these comments and from observations made during experiments, we can see that humans are not used to point accurately. Kendon [14] describes pointing to be, among others, a referential gesture, that is meant show and refer to objects to other humans, which requires less accuracy than what a computer needs for reliable target selection. We thus assume that an interaction technique, based on the resulting model, needs to be flexible, such that it can account for errors and inaccuracies made by the machine learning model.

6.2 Modelling the pointing movement

From the constructed features, evaluated by the two approaches classification and regression, we build the a model for human pointing. The model consists of these selected properties of human pointing. In table 5 we can see that most selected features (apart from the distance features) are related to either the index finger or HMD position. This is interesting, since it shows that the study setup might be simplified. For instance could the tracking provided by the 24 OptiTrack cameras be simplified by using simpler hand tracking either provided by an external device, such as LeapMotion[5], or by the HMD, for instance does the Oculus Quest support hand tracking[6]. It remains to be tested how much the performance of the model would be affected, when removing information about the arm and the shoulders, but initial testing with the existing data shows promising preliminary results.

Generally when constructing features, linear assumptions about the relationship between target positions and marker set positions do not work. For instance does the index finger height not grow linearly with the target height (see figure 12). This is due to the natural physiology of humans, since for instance the arm is firmly attached to the shoulder, such that the index finger vertical position and depth are on the perimeter of a circle with origin in the shoulder and an approximate radius of the arm length. This makes it hard to purely rely on linear statistical methods, such as correlation, for analysis of features.

[5] https://www.ultraleap.com/product/leap-motion-controller/ *(accessed: 07.02.20)*
[6] https://developer.oculus.com/blog/hand-tracking-sdk-for-oculus-quest-available/ *(accessed: 07.02.20)*

The machine learning models show that the selected features do not express the depth of targets well, especially in the center aisle. But with a classification accuracy of 87.46% and an mean error of 24.42 cm (considering a distance between target of 100 cm and the target diameter of 15 cm), we can say that the constructed model is relatively reliable.

Classification and regression are very different approaches and very much dependent on the desired output. In this case we want to asses whether we can use machine learning to model the human pointing at all. Classification is, in the use case of pointing as an interaction technique, not what we desire, since interactable objects in the virtual or real world will not always appear at a fixed position that are in the presented dataset. But classification is good for figuring out whether we can find some properties of the human pointing that can describe something about which object the human points at and serve as a proof of concept, that we can model human pointing. As we saw earlier, we can train a model that can predict targets based on the human pointing, which gives reason to continue experiments with regression. In this use-case, does regression seem to be the harder task to perform well at. When constructing an input technique based on human pointing, regression would be desired.

Generally is the collected dataset not well suited for the regression task, both because of the study setup and the relatively small number of samples. In the study setup, we use 27 different fixed target positions, which might introduce a bias in regression algorithms, meaning that this model would most likely not work well on samples where humans point at targets, that are not one of those 27. To make a model that can handle these unseen targets well, we need an extension of the existing dataset that includes samples with targets at (relatively) random positions. A challenge for constructing such a dataset is that we need many more datapoints, to make sure that the randomly positioned are evenly distributed. Additionally there are many small optimizations and assumptions for such a dataset, since we work with humans and their individual differences. For instance do we need to limit the space in which a target can appear, for instance since humans normally do not point backwards. But we can use the existing dataset for assessing whether we can observe an effect, which we can. This means that we now know that we can build a model for human pointing based on properties of human movement.

6.3 Pointing movement as interaction technique

With the constructed model we are able to implement hand free pointing, that we assume to feel natural, since it is based on natural movement data. The implementation would have similar applications as existing ray casting techniques [8, 21, 22] and the *Go-Go* technique [26], with the main difference that this implementation would focus on naturalness of input, compared to the other techniques. We speculate that the constructed model can be used outside of VR, ie. in *Augmented Reality* (AR) and in the real world, contrary to the *Go-Go* technique, that bends reality by elongating the arm. Plaumann et al. [25] state the need for free hand pointing in th real world. We use VR to only build a controlled environment in which we can collect data, but Mayer et al. [20] notes that humans point differently in VR, because of differences in eg. field of view in VR compared to the real world. It will require further investigation, whether our model suffers the same problems as Mayer et al. [20] experienced.

7 FUTURE WORK - FUTURE DIRECTION

This work shows that we can model the human pointing, based on the pointing position. The collected data and used machine learning is relatively simple, and thus this work serves the purpose of a proof of concept, for now we know that it is possible to model human pointing. Future work will show further advancements in this topic.

7.1 Next steps

First we will explore the next logical steps, which aim to improve the underlying machine learning model.

7.1.1 Improving features. As discussed in section 5.5, does the machine learning model struggle with predicting the right depth of targets. This can be improved by constructing new features, that describe the depth of target better then current features. The correlation matrices in figures 9 and 10 show that there is little linear correlation between the collected data and the target depth. This could either be corrected by finding some transformation, that computes a depth feature from the existing data, or maybe we need some more datapoints that extend the dataset. This could for instance be eye tracking data.

In addition to this it could be interesting to investigate where the depth prediction breaks. For instance could we align targets in a 1 x 1 x 10 grid with a distance of 50 cm between targets and see whether we can extract meaningful feature from the gathered data.

7.1.2 Exploring boundaries. As a next step it could be interesting to look at how well the regression machine learning model works with new, unseen data where users do not point on targets from the current grid structure. We do not expect great results, since the current model is boxed into the specific data from the 3 x 3 x 3 target grid.

7.2 Near future

Now that we know that we can predict targets based on the human movement, we can extend the functionality of the proposed model.

7.2.1 More realistic data. After improving the current model, we want to extend it with more realistic datapoints. This can be done by collecting the movement of humans that point at random targets. The current data is collected from targets fixed in a 3 x 3 x 3 grid. This means that the current machine learning is biased towards these 27 targets, which might not work well for specific applications. The task would be to extend the dataset and thereby the model, which in turn would become more generalizable.

A challenge with constructing an extended dataset is that the distribution of targets needs to be relatively even in all three dimensions, such that we do not introduce a bias. Additionally will we most likely need many more datapoints, such that the machine learning can separate targets that are potentially very close to each other. But collecting more data will possibly open up for more complex machine learning models, that may be harder to evaluate, but also might yield better prediction results.

7.2.2 Predict targets based on whole movement. In the data collection study did we collect positional and orientational data for the whole human movement pattern while pointing.It would be interesting to explore the whole movement pattern, contrary to only the endpoint of the pattern. We could then build a model that can predict the position of a target before the pointing movement completed. Challenges for such an application could be that if we would model the movement based on a complete collection of movement samples, we limit the generalizability of the application by design. In the collection study did we for instance ask the users to start their pointing movement after they put their hand on their stomach. We could investigate whether we could predict targets based on eg. the last three samples, contrary to the whole movement pattern.

7.3 Far future

After we have worked with and extended the current model, we want to start new projects, that are based on this model. We can for instance aim to build an actual interaction technique using the machine learning model.

7.3.1 Building an interaction technique. With the proposed model we can build an interaction technique, that is based on the natural human movement. Initial studies of this technique could be using a similar setup to what we proposed in our data collection study, but rather than collecting data, we want to use the positions of the marker sets to predict the targets the participants point at. The technique should be evaluated in terms of accuracy (does the participant point at a different target than predicted?) and in terms of usability, to determine the degree of the subjective feeling of naturalness of the interaction.

The machine learning model is also useful in non-VR environments. As long there is a way to track the arm and hand of the user in a similar way we did, it is possible to apply the model in an AR and real world setting, similar to what Plaumann et al. [25] describes, but we need to keep in mind *"that participants point differently while they are in VR."* [20]

When using this interaction technique we can take advantage of the fact that we, as developers, in many applications know the position of interactable objects within the space. We can thus correct for imprecision in the regression machine learning model, by highlighting the interactable object that is closest to the predicted target position.

7.3.2 Using other tracking technologies. As we briefly discussed in section 6, are most of the selected features based around the index finger and HMD. A small test with this subset of features showed promising results and thus it could be interesting to use a simpler tracking technique than OptiTrack, which requires much setup and is expensive. We mentioned LeapMotion or a build-in hand tracking solution as candidates for a consumer grade solution for tracking the hand. Also the Kinect[7] could be imagined to be useful in this task. The first two tracking approaches have the disadvantage that they are used in connection with a HMD, while the Kinect can be used outside of VR (and possibly AR). Additionally have Nickel and Stiefelhagen [22] proposed a gesture recognition application, that can recognize the pointing movement. This is useful for implementing a larger application, that not assumes every hand movement is a pointing movement.

[7] https://developer.microsoft.com/en-us/windows/kinect/ (accessed 12.02.20)

8 CONCLUSION

This work focuses on natural human pointing movement. Current selection techniques do not allow for both naturalness and selection of distant targets at the same time. Both properties are desirable in VR. We have thus build a model for a selection technique, that both is natural and supports selection of distant targets.

We have collected, analysed and modelled the natural human pointing movement. The resulting model is based on a well performing machine learning model, that predicts target positions based on the collected human pointing movement. The machine learning model encapsulates different factors important to the human pointing, but is limited to the described setup. We theorize that, since we can construct a well performing model, we can extend it to work with a more general setup. The collected data can be used for extending the current model.

With the constructed model we can build a selection technique, that can be used as an alternative to current selection technique and that additionally supports natural human capabilities and knowledge. The model shows the feasibility of modelling human movements and thus opens for new selection techniques, as well as other techniques and applications that are based on human movement.

REFERENCES

[1] Adam W. Anz, Brandon D. Bushnell, Leah Passmore Griffin, Thomas J. Noonan, Michael R. Torry, and Richard J. Hawkins. 2010. Correlation of torque and elbow injury in professional baseball pitchers. *American Journal of Sports Medicine* 38, 7 (2010), 1368–1374. https://doi.org/10.1177/0363546510363402

[2] Ferran Argelaguet, Carlos Andujar, and Ramon Trueba. 2008. Overcoming eye-hand visibility mismatch in 3D pointing selection. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST* (2008), 43–46. https://doi.org/10.1145/1450579.1450588

[3] Shelly M. Asbee, Todd R. Jenkins, Jennifer R. Butler, John White, Mollie Elliot, and Allyson Rutledge. 2015. Feature Synthesis Automation. *Obstetrics and Gynecology* 113, 2 PART 1 (2015), 305–312. https://doi.org/10.1097/AOG.0b013e318195baef

[4] Anil Ufuk Batmaz, Mayra Donaji Barrera Machuca, Duc Minh Pham, and Wolfgang Stuerzlinger. 2019. *Do Head-Mounted Display Stereo Deficiencies Affect 3D Pointing Tasks in AR and VR?* Technical Report. 585–592 pages. https://doi.org/10.1109/vr.2019.8797975

[5] Richard A. Bolt. 1980. "Put-that-there": Voice and gesture at the graphics interface. *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1980* (1980), 262–270. https://doi.org/10.1145/800250.807503

[6] Juan Sebastian Casallas, James H. Oliver, Jonathan W. Kelly, Frederic Merienne, and Samir Garbaya. 2014. Using relative head and hand-target features to predict intention in 3D moving-target selection. *Proceedings - IEEE Virtual Reality* (2014), 51–56. https://doi.org/10.1109/VR.2014.6802050

[7] Hélène Cochet and Jacques Vauclair. 2010. Pointing gesture in young children. *Gesture* 10, 2-3 (2010), 129–149. https://doi.org/10.1075/gest.10.2-3.02coc

[8] Andrea Corradini and Philip R. Cohen. 2002. Multimodal speech-gesture interface for handfree painting on a virtual paper using partial recurrent neural networks as gesture recognizer. *Proceedings of the International Joint Conference on Neural Networks* 3, August 2002 (2002), 2293–2298. https://doi.org/10.1109/IJCNN.2002.1007499

[9] Yashar Deldjoo and Reza Ebrahimi Atani. 2016. A low-cost infrared-optical head tracking solution for virtual 3D audio environment using the Nintendo Wii-remote. *Entertainment Computing* 12, January (2016), 9–27. https://doi.org/10.1016/j.entcom.2015.10.005

[10] Tiare Feuchtner and Jörg Müller. 2018. Ownershift: Facilitating overhead interaction in virtual reality with an ownership-preserving hand space shift. *UIST 2018 - Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (2018), 31–43. https://doi.org/10.1145/3242587.3242594

[11] J. M. Foley and Richard Held. 1972. Visually directed pointing as a function of target distance, direction, and available cues. *Perception & Psychophysics* 12, 3 (1972), 263–268. https://doi.org/10.3758/BF03207201

[12] L. Gallo and A. Minutolo. 2012. Design and comparative evaluation of Smoothed Pointing: A velocity-oriented remote pointing enhancement technique. *International Journal of Human Computer Studies* 70, 4 (2012), 287–300. https://doi.org/10.1016/j.ijhcs.2011.12.001

[13] Claire C. Gordon, Cynthia L. Blackwell, Bruce Bradtmiller, Joseph L. Parham, Patricia Barrientos, Stephen P. Paquette, Brian D. Corner, Jeremy M. Carson, Joseph C. Venezia, Belva M. Rockwell, Michael Mucher, and Shirley Kristensen. 2014. 2012 Anthropometric Survey of U.S. Army Personnel: Methods and Summary Statistics. *Security* December 2014 (2014), 640.

[14] Adam Kendon. 2015. *Gesture: Visible action as utterance*. Cambridge University Press. 1–388 pages.

[15] Udayan Khurana, Fatemeh Nargesian, Horst Samulowitz, Elias Khalil, and Deepak Turaga. 2016. Automating Feature Engineering. *30th Conference on Neural Information Processing Systems* Nips (2016), 2016–2017.

[16] Udayan Khurana, Deepak Turaga, Horst Samulowitz, and Srinivasan Parthasrathy. 2016. Cognito: Automated Feature Engineering for Supervised Learning. *IEEE International Conference on Data Mining Workshops, ICDMW* 0 (2016), 1304–1307. https://doi.org/10.1109/ICDMW.2016.0190

[17] Konstantina Kilteni, Jean Marie Normand, Maria V. Sanchez-Vives, and Mel Slater. 2012. Extending body space in immersive virtual reality: A very long arm illusion. *PLoS ONE* 7, 7 (2012). https://doi.org/10.1371/journal.pone.0040867

[18] Ágnes Melinda Kovács, Tibor Tauzin, Erno Téglás, György Gergely, and Gergely Csibra. 2014. Pointing as epistemic request: 12-month-olds point to receive new information. *Infancy* 19, 6 (2014), 543–557. https://doi.org/10.1111/infa.12060

[19] Shaul Markovitch and Dan Rosenstein. 2002. Feature generation using general constructor functions. *Machine Learning* 49, 1 (2002), 59–98. https://doi.org/10.1023/A:1014046307775

[20] Sven Mayer, Valentin Schwind, Robin Schweigert, and Niels Henze. 2018. The effect of offset correction and cursor on mid-air Pointing in real and virtual environments. *Conference on Human Factors in Computing Systems - Proceedings* 2018-April (2018). https://doi.org/10.1145/3173574.3174227

[21] Sven Mayer, Katrin Wolf, Stefan Schneegass, and Niels Henze. 2015. Modeling distant pointing for compensating systematic displacements. *Conference on Human Factors in Computing Systems - Proceedings* 2015-April (2015), 4165–4168. https://doi.org/10.1145/2702123.2702332

[22] Kai Nickel and Rainer Stiefelhagen. 2003. Pointing gesture recognition based on 3D-tracking of face, hands and head orientation. *ICMI'03: Fifth International Conference on Multimodal Interfaces* (2003), 140–146.

[23] David Noy. 2001. Predicting user intentions in graphical user interfaces using implicit disambiguation. *Conference*

on Human Factors in Computing Systems - Proceedings (2001), 455–456. https://doi.org/10.1145/634067.634330

[24] Tomi Nukarinen, Jari Kangas, Jussi Rantala, Olli Koskinen, and Roope Raisamo. 2018. Evaluating ray casting and two gaze-based pointing techniques for object selection in virtual reality. (nov 2018). https://doi.org/10.1145/3281505.3283382

[25] Katrin Plaumann, Matthias Weing, Christian Winkler, Michael Müller, and Enrico Rukzio. 2018. Towards accurate cursorless pointing: the effects of ocular dominance and handedness. *Personal and Ubiquitous Computing* 22, 4 (2018), 633–646. https://doi.org/10.1007/s00779-017-1100-7

[26] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 1996. The go-go interaction technique. In *UIST (User Interface Software and Technology): Proceedings of the ACM Symposium*. ACM, 79–80. https://doi.org/10.1145/237091.237102

[27] Tara Rawat. 2017. Feature Engineering (FE) Tools and Techniques for Better Classification Performance. *International Journal of Innovations in Engineering and Technology* 8, 2 (2017), 169–179. https://doi.org/10.21172/ijiet.82.024

[28] Valentin Schwind, Pascal Knierim, Cagri Tasci, Patrick Franczak, Nico Haas, and Niels Henze. 2017. "These are not my hands!": Effect of gender on the perception of avatar hands in virtual reality. *Conference on Human Factors in Computing Systems - Proceedings* 2017-May (2017), 1577–1582. https://doi.org/10.1145/3025453.3025602

[29] Valentin Schwind, Sven Mayer, Alexandre Comeau-Vermeersch, Robin Schweigert, and Niels Henze. 2018. Up to the fingertip: The effect of avatars on mid-air pointing accuracy in virtual reality. In *CHI PLAY 2018 - Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*. Association for Computing Machinery, Inc, 489–502. https://doi.org/10.1145/3242671.3242675

[30] Mel Slater, Martin Usoh, and Anthony Steed. 1994. Depth of Presence in Virtual Environments. *Presence: Teleoperators and Virtual Environments* 3, 2 (1994), 130–144. https://doi.org/10.1162/pres.1994.3.2.130

[31] Robert J. Teather and Wolfgang Stuerzlinger. 2014. Visual aids in 3D point selection experiments. In *SUI 2014 - Proceedings of the 2nd ACM Symposium on Spatial User Interaction (SUI '14)*. Association for Computing Machinery, New York, NY, USA, 127–136. https://doi.org/10.1145/2659766.2659770

[32] Sarah van der Land, Alexander P. Schouten, Frans Feldberg, Bart van den Hooff, and Marleen Huysman. 2013. Lost in space? Cognitive fit and cognitive load in 3D virtual environments. *Computers in Human Behavior* 29, 3 (2013), 1054–1064. https://doi.org/10.1016/j.chb.2012.09.006

[33] Difeng Yu, Hai Ning Liang, Feiyu Lu, Vijayakumar Nanjappan, Konstantinos Papangelis, and Wei Wang. 2018. *Target selection in head-mounted display virtual reality environments*. Technical Report 9. 1217–1243 pages.

LIST OF FIGURES

1 Close-ups on the 3D models of the marker sets used for tracking with OptiTrack. Designed by Mayer et al. [20]. 10
2 A user wearing the OptiTrack marker sets. 10
3 Each target (green) labeled with its positon arranged in the 3 x 3 x 3 grid. The red cross is the position users stand on. [m] 11
4 The virtual hand pointing. 11
5 A depiction of the calibration tasks. 14
6 Example of a visualization of the collected movement data. Each line represents a sample of a timestamp, while the slightly thicker red line is the final position of the arm. Each point on the line is the position of a rigidbody tracked. This movement took 1.8 seconds to complete. 14
7 The process-diagram of modelling the human pointing movement. 17
8 Visualization of all endpoints that where produced by all participants pointing at a particular target. The data is adjusted such that the HMD-position is at `(0, 0, 0)`, in order to be able compare the endpoints in a meaningful way. Additionally, the left shoulder marker position is not shown on these visualizations for clarity. 18
9 Correlation matrix, showing the relationship between the marker set positions and the target position. Note that the labels are shortened, see appendix A for an overview. The lower left corner shows a strong correlation between the index finger position and the target position. 21
10 Correlation matrix, showing the relationship between the marker set orientations and the target position. On this matrix we see some correlation between the upper arm orientation and the target positions in the lower center. 21
11 Boxplot of the horizontal index finger position, grouped by target position. 0 is the vertical center of the participant. 23
12 Boxplot of the vertical index finger position, grouped by target position. 0 is the floor level, at the feet of the participant. 23
13 Boxplot of the depth of index finger position, grouped by target position. 0 is the horizontal center of the participant. 23
14 Boxplot of the elbow flexion, grouped by target. 23
15 Boxplot of the horizontal HMD position, grouped by target position. 0 is the vertical center of the participant. 24
16 Boxplot of the vertical HMD position, grouped by target position. 0 is the horizontal center of the participant. . 24
17 Density plot of the upper arm pitch, grouped by target position. 25
18 Density plot of the upper arm yaw, grouped by target. 25
19 Density plot of the HMD pitch, grouped by target. 26
20 Density plot of the HMD yaw, grouped by target position. 26
21 Definition of elbow and shoulder angles. Figure and definition inspired by Anz et al. [1]. 26
22 Boxplot of the depth of the shoulder abduction, grouped by target position. 27
23 Boxplot of the shoulder horizontal angle, grouped by target. 27
24 Bar graph of a binary indicator, that describes whether the index finger is vertically positioned above the participants head. 0 means false, 1 means true. 29
25 Bar graph of a binary indicator, that describes whether the index finger is vertically positioned above the hand. 0 means false, 1 means true. 29
26 Bar graph of the horizontal index finger category, grouped by target position. 30
27 Bar graph of the vertical index finger category, grouped by target. 30

28 Bar graph of the index finger depth category, grouped by target position. 31
29 Boxplot of the distance between HMD and the upper arm, grouped by target. 31
30 Boxplot of the distance between HMD and the index finger, grouped by target position. 31
31 Boxplot of the distance between the forearm and hand, grouped by target. 31
32 Correlation matrix, showing the relationship between the constructed features and the target position. In the bottom left corner, we can see that the elbow flexion feature does not correlate to the target position, while the shoulder abduction has some correlation with the vertical target position. 32
33 Confusion matrix of the predictions made on the test set with a SVM. True predictions are counted on the diagonal, while all other counts indicate errors. 36
34 The target grid, where the colors of the target correspond to their mean F1 score (using SVM), for all folds. The red cross is the participant starting position. Here we can see that the center aisle of targets produces a lower F1 score than other targets. . . 36
35 The target grid, where the colors of the target correspond to their average distance to the predicted target (using SVM), for all folds. The red cross is the participant starting position. Clearly, the target closest to the participant is identified as an outlier. [m] 37
B.1 Plot of the χ^2-test results. Features scoring below the red line at $p = 0.05$, can be seen as significant. 50
B.2 A bar graph showing all features importance, sorted from least important to most important. 50
B.3 Plot of the mutual information. The values are grouped by target positional value and sorted by least summed mutual information to most. 50

LIST OF TABLES

1 Number of papers and articles found and selected in search 6
2 Transformation and rotation vectors for the rigidbodies. The unit of the transformation vectors is millimeter and degrees in the case of the orientation vectors. The coordinate system is in y-up fashion. 12
3 Overview of the participant measures. [m] 16
4 Overview of the selected calibration data points. [cm] 19
5 Overview of the selected features. . . 34
6 Overview of classification accuracies for different models. The best performing model and its scores are highlighted in bold. 35
7 Zoom-in on the confusion matrix shown in figure 33 on targets with coodinates `(0, 0.49, z)`. 36
8 Zoom-in on the confusion matrix shown in figure 33 on targets with coodinates `(0, 1.49, z)`. 36
9 Overview of regression scores for different models. The cross validation is obtained by 5-fold cross validation. The best performing model and its score are highlighted in bold. 37
10 Metrics of the distance between prediced and true targets. Note that the distance between the maximum value and the 75th percentile is rather large, indicating that the maximum value is an outlier, that impacts the mean. See appendix table C.3 for a full overview of all prediction distances. [m] 37
A.1 Translation between key words and feature names used in figures. 49
C.1 Overview of classification models with their used parameters. The parameters where found by grid search. 51
C.2 Overview of regression models with their used parameters. The parameters where found by grid search. . . . 51
C.3 Distances between true target positions and predicted target positions. Note here that the predicted x and y values are very close to the actual target values, while most of the error (distance) is stacked up in the z value prediction. We can see that the predictions for target with coordinates `(-1.0, 1.49, 3.5)` and `(0.0, 1.49, 1.5)` contribute to much of the overall average error. 52

Long	Short	Long	Short	Long	Short
index finger	idf	right shoulder	rsh	distance	dist
hand	hnd	left shoulder	lsh	position	pos
forearm	foa	HMD	hmd	body	bdy
upper arm	upa	elbow	elb	shoulder	sh

Table A.1. Translation between key words and feature names used in figures.

A SHORT NAMES FOR DATA POINTS AND BODY FIELDS

In some figure labels the feature or body field name is shortened, since else the readability of the figure would be worse. Table A.1 shows the translation between short and long names.

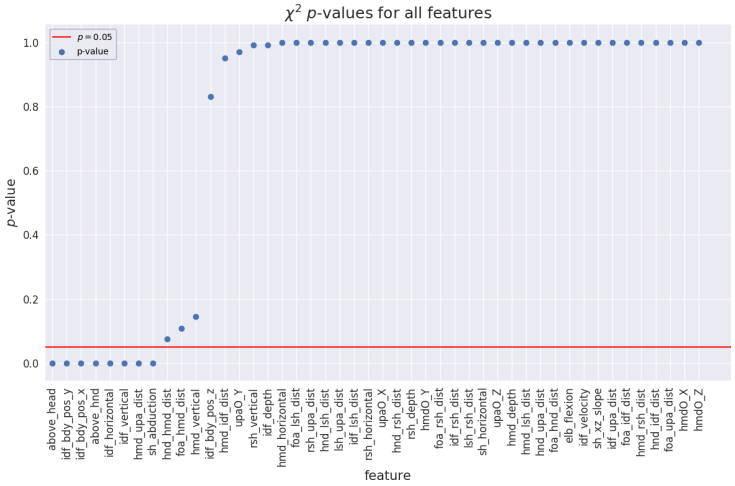


Fig. B.1. Plot of the χ^2-test results. Features scoring below the red line at $p = 0.05$, can be seen as significant.

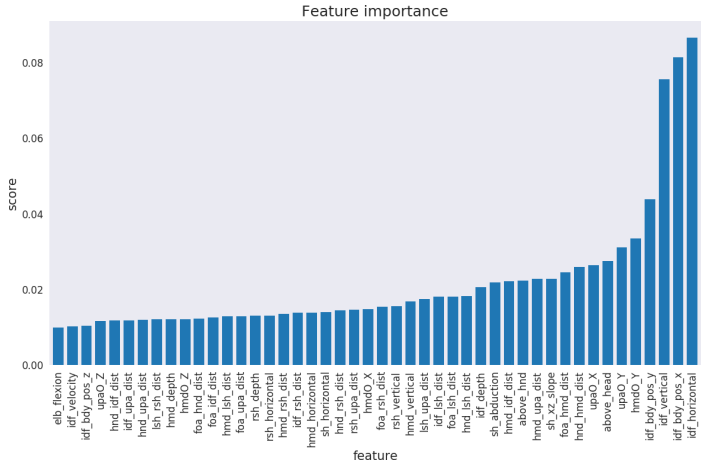


Fig. B.2. A bar graph showing all features importance, sorted from least important to most important.

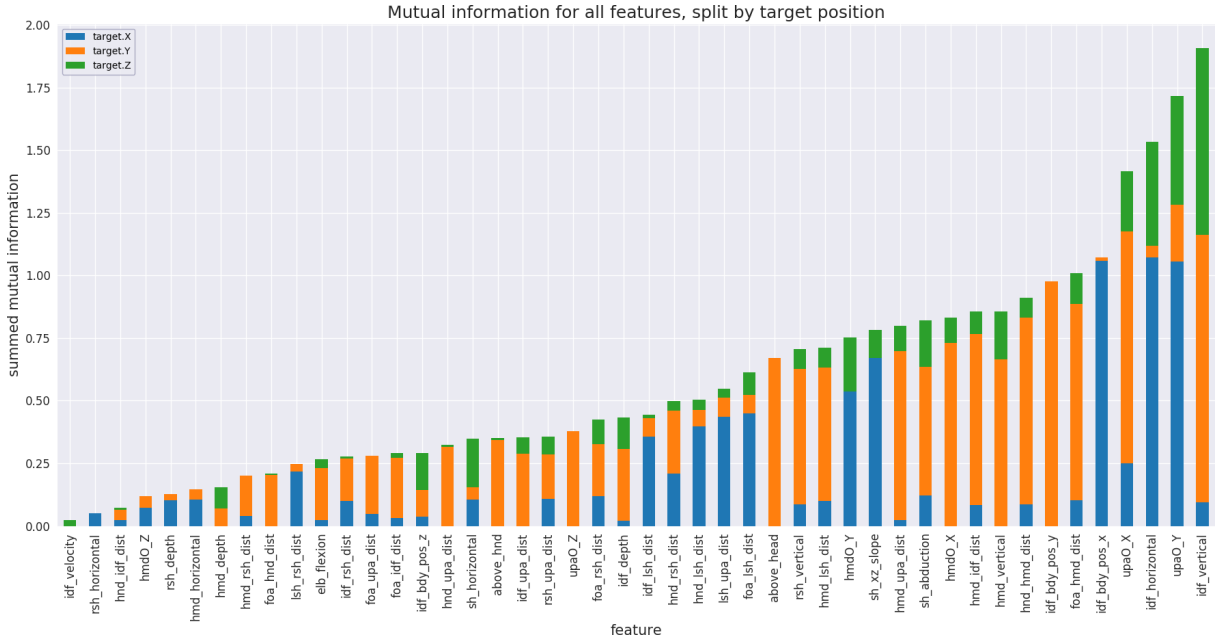


Fig. B.3. Plot of the mutual information. The values are grouped by target positional value and sorted by least summed mutual information to most.

B FEATURE SELECTION

We use different methods for feature selection, which we describe in section 5.4. These techniques aim to rank the features, typically on some statistical measure. Figures B.1 to B.3 show the results of three of these techniques: a χ^2-test, mutual information and feature importance respectively. From these figures we can see an overall trend towards features that have a basis in the index finger and HMD positional values.

Model	`scikit-learn` parameters
Naive Bayes[8]	`N/A`
KNN[9]	`{n_neighbors: 1}`
SVM[10]	`{C: 20000, gamma: 0.001, kernel: rbf}`
Random Forest[11]	`{n_estimators: 150, max_features: auto, max_depth: 17, criterion: entropy}`

Table C.1. Overview of classification models with their used parameters. The parameters where found by grid search.

Model	`scikit-learn` parameters
Linear Regression[12]	`{C: 1, tol: 0.0001, penalty: l2}`
KNN[13]	`{n_neighbors: 2}`
SVM[14]	`{C: 100, gamma: 0.01, kernel: rbf}`
Random Forest[15]	`{n_estimators: 200, max_features: auto, max_depth: 13, criterion: mse}`

Table C.2. Overview of regression models with their used parameters. The parameters where found by grid search.

C MACHINE LEARNING

C.1 Model parameters

For completeness do we here attach the used machine learning parameters, that where used to compare different machine learning models. All these where found through grid search. Table C.1 shows the parameters used for classification model, while C.2 shows the parameters for regression models.

[8] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html *(accessed: 29.01.20)*
[9] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html *(accessed: 29.01.20)*
[10] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html *(accessed: 29.01.20)*
[11] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html *(accessed: 29.01.20)*

[12] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html *(accessed: 29.01.20)*
[13] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html *(accessed: 29.01.20)*
[14] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html *(accessed: 29.01.20)*
[15] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html *(accessed: 29.01.20)*

True x	**True** y	**True** z	**Predicted** x (mean)	**Predicted** y (mean)	**Predicted** z (mean)	**Distance** [m]
-1.0	0.49	1.5	-1.01	0.48	1.34	0.1606
-1.0	0.49	2.5	-1.00	0.44	2.58	0.0943
-1.0	0.49	3.5	-0.89	0.55	3.45	0.1349
-1.0	1.49	1.5	-1.09	1.48	1.73	0.2472
-1.0	1.49	2.5	-0.98	1.49	2.46	0.0447
-1.0	1.49	3.5	-0.84	1.47	2.82	0.6989
-1.0	2.49	1.5	-0.94	2.39	1.57	0.1360
-1.0	2.49	2.5	-0.95	2.46	2.64	0.1517
-1.0	2.49	3.5	-0.84	2.39	3.22	0.3376
0.0	0.49	1.5	0.03	0.52	1.42	0.0906
0.0	0.49	2.5	-0.02	0.47	2.63	0.1330
0.0	0.49	3.5	-0.05	0.53	3.38	0.1360
0.0	1.49	1.5	-0.01	1.42	2.72	1.2220
0.0	1.49	2.5	-0.06	1.48	2.83	0.3356
0.0	1.49	3.5	-0.05	1.56	3.00	0.5073
0.0	2.49	1.5	0.02	2.58	1.64	0.1676
0.0	2.49	2.5	-0.03	2.50	2.68	0.1828
0.0	2.49	3.5	-0.05	2.42	3.19	0.3217
1.0	0.49	1.5	1.02	0.49	1.36	0.1414
1.0	0.49	2.5	1.05	0.45	2.60	0.1187
1.0	0.49	3.5	0.98	0.53	3.58	0.0917
1.0	1.49	1.5	1.06	1.49	1.58	0.1000
1.0	1.49	2.5	1.03	1.52	2.65	0.1559
1.0	1.49	3.5	0.91	1.53	3.13	0.3829
1.0	2.49	1.5	1.00	2.51	1.43	0.0728
1.0	2.49	2.5	0.94	2.36	2.68	0.2300
1.0	2.49	3.5	0.86	2.42	3.38	0.1972
					distance mean	*0.2442*
					distance median	*0.1559*

Table C.3. Distances between true target positions and predicted target positions. Note here that the predicted x and y values are very close to the actual target values, while most of the error (distance) is stacked up in the z value prediction. We can see that the predictions for target with coordinates `(-1.0, 1.49, 3.5)` and `(0.0, 1.49, 1.5)` contribute to much of the overall average error.

C.2 Regression: predicted targets

Table C.3 shows the true targets, the predicted targets and the distance between the two positions. In section 5.5.3 we describe the findings of this table.