Aalto University
School of Electrical Engineering
Master's Programme in Autonomous Systems

Thomas Weikert

# Predicting User's Head Motion in XR with a Gated Recurrent Unit based Model

Master's Thesis
Espoo, December 30, 2021

Supervisors: Professor Mario Di Francesco, Aalto University
Advisor: Matti Siekkinen, PhD, Aalto University
Teemu Kämäräinen, PhD, Streamed Reality Oy

Aalto University
School of Electrical Engineering
Master's Programme in Autonomous Systems

ABSTRACT OF
MASTER'S THESIS

**Author:** Thomas Weikert

**Title:**
Predicting User's Head Motion in XR with a Gated Recurrent Unit based Model

**Date:** December 30, 2021 **Pages:** 62

**Major:** Autonomous Systems **Code:** ELEC3055

**Supervisors:** Professor Mario Di Francesco, Aalto University
**Advisor:** Matti Siekkinen, PhD, Aalto University
Teemu Kämäräinen, PhD, Streamed Reality Oy

Extended Reality enables 3D content to be experienced with six degrees of freedom with head-mounted displays (HMD). Since rendering XR experiences with head-mounted displays (HMD) requires prohibitively high computational power, the use of remote rendering (RR) can outsource the required computational power. One drawback of this solution is the increased interaction latency (motion-to-photon), which can lead to latency errors in XR applications. One possible approach to mitigate this problem is to predict the viewer's head pose in order to pre-load the appropriate content. A suitable time frame for predicting future head motion is 100 milliseconds, as this is approximately the round-trip time of the data in an average remote rendering environment. To predict the immediate future (100 milliseconds) head motion, a method based on Gated Recurrent Units is presented to model the position and rotation of the user's head. The model is compared to a state-of-the-art method, Double Exponential Smoothing Prediction (DESP), and a baseline, the time-lagged time series by 100 milliseconds. he developed models meet the goals of the work: the models outperform the state-of-the-art method DESP and the selected baselines in head position and rotation.

**Keywords:** Extended Reality, Gated Recurrent Unit, Head Motion Prediction, Double Exponential Smoothing, Time Series Analysis

**Language:** English

# Acknowledgements

This report contains the results of my work as a research assistant that serves as the final project of my graduate studies in my Master's in Autonomous Systems at KTH Royal Institute of Technology and Aalto University through EIT Digital Master School. In this thesis project, I was given the chance to dive into an interesting academic machine learning topic with relevance for the industry.

I am grateful to Professor Mario Di Francesco and Matti Siekkinen from Aalto University for giving me the opportunity to join this project and their supervision. I also would like to thank Teemu Kämäräinen at Streamed Reality Oy. Thank you all for your contentious guidance and feedback.

To Gazi Illahi and Ashutosh Vaishnav at Aalto University, who were kind enough to provide me with continuous support and help throughout the project. For me, the internship was a great chance for learning and personal development.

And lastly, to my family and friends back home, who directly and indirectly contributed to this project: Herzlichen Dank fuer eure Unterstuetung und aufmunternden Worte in den letzten beiden Jahren. Thank you all for bringing light to these times, in spite of the pandemic and everything else going on around us, and making this an unforgettable experience.

Espoo, 30.12.2021

Thomas Weikert

# Abbreviations and Acronyms

| | |
|---|---|
| **AR** | Augmented Reality |
| **CNN** | Convolutional Neural Network |
| **DESP** | Double Exponential Smoothing Prediction |
| **DoF** | Degree of freedom |
| **EKF** | Extended Kalman Filter |
| **GRU** | Gated Recurrent Unit |
| **HMD** | Head-Mounted-Display |
| **KF** | Kalman Filter |
| **LAT** | Look ahead time |
| **LSTM** | Long-Short-Term Memory |
| **M2P** | Motion-to-Photon |
| **MAE** | Mean Absolut Error |
| **MEC** | Mobile Edge Computing |
| **ML** | Machine Learning |
| **MR** | Mixed Reality |
| **NLP** | Natural Language Processing |
| **ReLu** | Rectified Linear Unit |
| **RNN** | Recurrent Neural Network |
| **RR** | Remote Rendering |
| **SDG** | Stochastic Gradient Descent |
| **UKF** | Unscented Kalman filter |
| **VR** | Virtual Reality |
| **XR** | Extended Reality |

# Contents

# Chapter 1

# Introduction

The topic of this Master's thesis is the process of predicting human head motion with a 6-dimensional degree of freedom (DoF) for Extended Reality (XR) applications. A wide range of XR applications has gained popularity in recent years. Among them are all relevant domains covering advertisement, cultural heritage, video gameplay, education, industry 4.0, and considerably more. People most frequently experience XR with a head-mounted display (HMD), which consists of two displays to show content to the user's eyes [26]. Meshes and point clouds represent the geometry of 3D objects. The density of data in polygons and point clouds is always very high. [44, 45]. In spite of the considerable computing power of modern mobile devices, rendering large-scale data is still a tedious task. [7]. In addition, there is still no powerful hardware for mesh/point cloud decoders. The software-based decoding process is possibly too expensive concerning power usage and may not meet the requirements for XR applications. [47]. A solution to this technical barrier offers remote rendering. Remote rendering represents a technology that renders 3D graphics on a computing device and shows the results on another computing device connected through a network. In the context of XR applications remote rendering is used to render a 2D view from the HMD of the user's interest based on the user's current head position [47]. The final step is to reduce the rendered texture into a 2D video stream and transmit it over a network to the client. The client can then efficiently extract the video stream using its device decoder and display the dynamically updated content to the viewer. Figure Figure 1.1 displays such a rendering system.

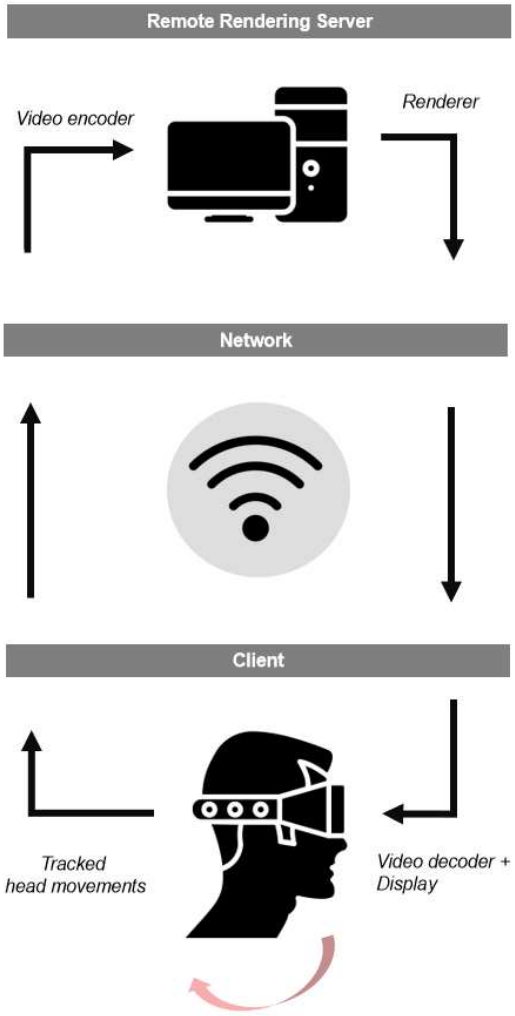



Figure 1.1: Illustrative remote rendering system adapted by [16]

On top of that, remote rendering enables the use of efficient 2D video coding techniques and can consequently decrease the network bandwidth requirements by avoiding the transmission of content[40]. However, one disadvantage of remote rendering is the increased interaction latency, also known as motion-to-photon (M2P) latency [46]. M2P latency is the time interval between the user's head movement (action) and the matching reflection of the display on the HMD (reaction). Due to network runtime and additional processing time, M2P latency is higher than a system that performs rendering locally. In addition, a number of research efforts have indicated that increased interaction latency can degrade the user experience and foster motion sickness. [5, 35, 38].

An option to mitigate latency is to predict the user's future head pose on the remote server and render the matching view of the HMD content in advance. The M2P latency can thus be significantly reduced if not eliminated if the user pose is successfully predicted for a look-ahead time (LAT) equal to or greater than the M2P latency of the systemM2P latency can therefore be greatly mitigated by predicting a head pose look-ahead time (LAT) that is equal to or greater than the system M2P latency. Nevertheless, erroneous predictions of head motion can increase the error rate and degrade the user experience in XR environments. Nevertheless, incorrect predictions of head motion can increase errors and affect the user experience in XR environments [35]. Consequently, the development of accurate head motion prediction algorithms is critical for the high-quality streaming of XR applications.

The main contributions of this thesis are the development of a Recurrent Neural Network-based predictor for head motion prediction in 6 DoF space and analyzing its performance compared to a double-exponential-smoothing model and a baseline (no prediction) model using recorded head motion datasets.

## 1.1 Problem statement

Low latency is critical for XR applications, so this thesis addresses the main problem of reducing latency by predicting head motions in the immediate future (100 milliseconds). The corresponding prediction problem is defined as predicting the future user's head position between t and t+H (H is equal to 100 milliseconds in this case) with the only available knowledge of this user's past position. A simplified overview of the prediction problem can be seen in Figure 1.2.

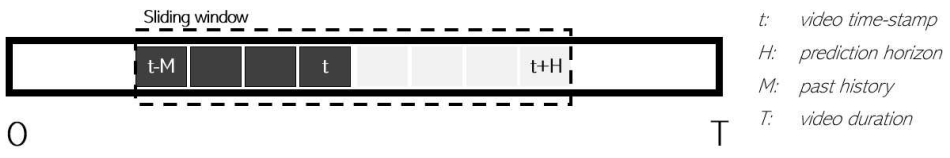



Figure 1.2: Simplified prediction problem adapted by [16]

Let $P_t = [\theta_t, \phi_t]$ denote the vector coordinates of the head position ($\theta_t$) and

head orientation ($\phi_t$) at time $t$. The problem exists at each time-stamp $t$ having the only knowledge of this user's past input data at this particular session. If $P_t$ defines the prediction at time $t$, $T$ the video duration, $H$ the prediction horizon, then the trajectory for all predictions is finding the most suitable model, which minimizes the expected distance, $D(.)$, between the ground truth series of the future positions and the series of predicted positions. Equation 1.1 formulates the problem to find the best model $F*_H$.

$$F_H^* = arg\,min\, E_t[D([P_{t+1}, ..., P_{t+H}])] \tag{1.1}$$

## 1.2 Structure of the Thesis

The organization of this thesis is as follows. The literature review chapter introduces the concepts of XR technologies and principles of motion prediction algorithms. It follows an overview of previous research. The following chapter outlines the pre-processing pipeline, architecture, and evaluation process used to develop ahead motion prediction algorithm. Next, the results are presented, followed by a discussion regarding its limitations and improvements. The conclusion is provided in the last chapter.

# Chapter 2

# Background

This chapter introduces the background of the thesis to investigate head motion prediction. First, the concept of extended reality (XR) related to the research problem is presented, followed by introducing different head motion prediction approaches.

## 2.1 Extended Reality

This thesis refers to extended reality (XR) as a technology that evolves around the volumetric capturing of 3D scenes and objects, such as in virtual reality (VR), augmented reality (AR), and mixed reality (MR). VR touches on a system, where a virtual world is rendered and displayed to the user, typically wearing a head-mounted display (HMD). AR overlays natural world environments with computer-generated imagery. Mixed reality also overlays virtual imagery over the real world, but the real world can interact with the virtual content somehow. The visual differences can be better described by seeing Figure 2.1


Figure 2.1: Differences between VR,AR and MR adopted by [11]

Volumetric video technologies have been developed since the 1960s [8] and have gained increasing popularity in recent years as it is offering opportunities for various entertainment, research, and industrial applications. The focus of this thesis is set on entertainment applications. In particular interactive real-time graphical applications are used within the experimental set up. In the context of XR, the experience of 3D technologies is mostly carried out via an HMD. All HMDs are equipped with sensors such as gyroscopes, motion sensors, and cameras to capture user information. The combined data allow insights about the user's orientation and location information to render different viewing angles based on the user head's movement. Nonetheless, streaming high density data in XR applications remains a significant challenge. Because of the required amount of computational resources in HMD's, which is considerably higher than in a regular desktop personal computer environment. Todays HMD devices have remarkable computing power [7]. Nonetheless, HMD devices are not capable to meet the requirements for real-time rendering [40]. Various technologies, including remote rendering, are promising approaches to avoid rendering on the HMD, as discussed in the following chapter.

## 2.2 Remote Rendering

Remote rendering emerged as an idea back in the 1990s, when computers did not have enough computing power for intensive graphics tasks [46]. Complex graphics are rendered on a capable server in a remote rendering system, and the result is broadcast over a network to a remote client device that is less powerful. The advent of cloud computing and mobile edge computing (MEC) has made real-time applications through RR servers possible [36, 40, 46]. In a remote rendering process for HMD devices, a powerful remote server is processing the streaming content and computes the streaming content based on the head pose, transfers the content into a 2D video format and delivers a compressed version back to the client over a network. In addition, remote rendering (utilizes cloud technology) can further reduce the network bandwidth requirements by applying highly efficient 2D video coding techniques, thus avoiding the transmission of volumetric content [40]. The main drawback of running an RR server is its high computational resources since the server has to serve many clients simultaneously. In addition, an RR server can have high bandwidth usage as it sends numerous high-resolution images over the network to RR clients. These two aspects can lead to the motion-to-photon latency phenomenon that will be discussed in the next chapter.

## 2.3 Motion-to-photon latency

Due to too few available computational resources, motion-to-photon latency occurs, which is a critical parameter for user experience in XR. Motion-to-photon latency describes the time gap between a user's physical motion and the resulting displayed content of an HMD [52]. At a minimum, a latency of 60ms is required in order to provide an acceptable user experience [28]. Any latency beyond this might cause cybersickness and will impact performance [1]. Multiple factors contribute to the latency. First, acquired tracking data needs to be processed, which must be passed on to the software that renders the content. Following this, the content is transferred to the display, which has pixel switching latency. Figure 3 illustrates a simplified motion-to-photon pipeline.

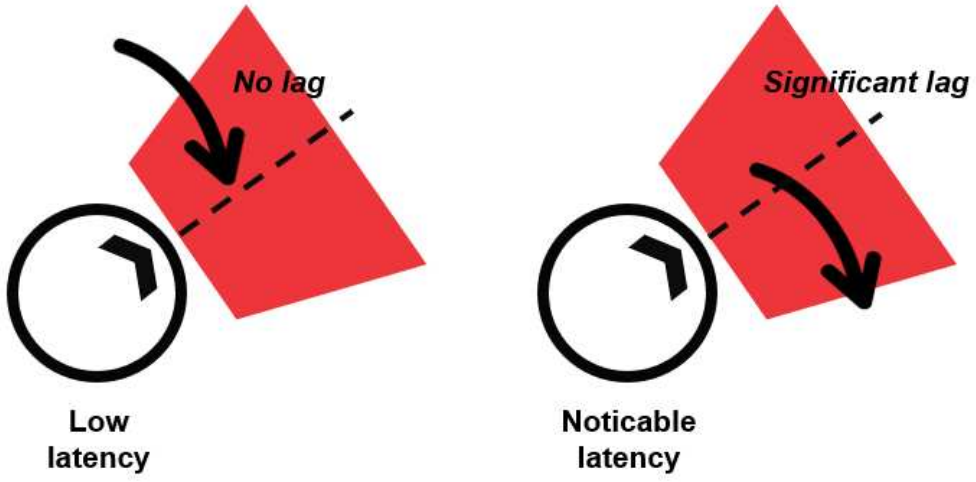



Figure 2.2: Illustration of the motion-to-photon latency adapted by [21]

Most XR systems aim to mitigate the risk of high motion-to-photon latency by predicting user intentions. Observed head tracking data is used as input, for example, the head's angular velocity. Such an approach usually succeeds when no unexpected direction changes and sudden acceleration happen. Nevertheless, the prediction fails for unpredictable movements, which turns into a high latency. Therefore designing an accurate head motion prediction algorithm is crucial for high-quality XR streaming. Head motion prediction relies on data from motion capture sensors in the HMD. This technology captures sequences of 3-dimensional joint positions, particularly head position and head orientation, at high frame rates (120 Hz - 1 kHz). In this context, the task of generating head motion sequences

has been approached with different strategies ranging from non-unit variate time series, classical machine learning, and the application of advanced deep learning algorithms.

An overview of different head motion modeling algorithms is given in the following chapters. When using non-uni variate time series models and classical machine learning algorithms, a careful selection of algorithms is needed to express the mutual dependence of the joints. For deep learning algorithms, the hope is to achieve a universal approximation through automatic feature learning. The formulation of the pose prediction task is as important as the choice of neural architecture. The input and output variables' definition, representation, and loss function used for training are of particular relevance, as the experiments in this thesis will show. Regarding the prediction of joint orientation, previous work has relied on various representations, such as quaternions, Euler angles, and rotation matrices.

## 2.4 Smoothing Algorithms for head motion prediction

The simplest methods for predicting head motion are smoothing methods. These are relatively simple methods characterized by robustness and accuracy. This makes them a desirable and widely used method. Simple methods have the advantage of being more transparent and therefore enjoy more trust than complex methods, even if the latter shows a higher accuracy. Smoothing methods are based on the idea of distinguishing between random fluctuations and the underlying pattern by averaging historical values. Random fluctuations are evenly distributed in the positive and negative directions, and averaging compensates for errors. Popular choices of smoothing methods are the Kalman Filter (KF), extended Kalman Filter (EKF) as well as the Double Exponential Smoothing Prediction (DESP). The KF/EKF predictors have received significant attention in previous research [2, 13, 24, 33] and appear to be the prediction method of choice for many researchers. Furthermore, previous researchers who used these algorithms have demonstrated that the KF/EKF is relatively fast and accurate for various user motion dynamics. However, this thesis regards DESP as a viable alternative over KF/EKF predictor counterparts because it has proven to generate similar performances, is simpler to understand and implement than KF/EKF predictors [29]. Therefore, DESP approach will be introduced in detail for predicting user position and orientation.

## Double Exponential Smoothing

The underlying idea of double exponential smoothing prediction (DESP) is that when setting weights for a moving average, recent observations are prioritized by an additional set weight. DESP does exactly this by automatically defaulting to a weighting of recent observations and decreasing the weighting of previous observations exponentially the longer they are in the past. The degree of exponential decay is determined by the parameter $\alpha \in [0, 1)$. The basic exponential smoothing equation to estimate a vector $p_t$ representing the x, y, and z components of position is given by 2.1 to smooth the original position sequence and 2.2 to smooth the $\vec{Sp}_t$ values. These equations follow a simple linear regression approach, where the y-intercept $\beta_0$ and slope $\beta_1$ vary slowly over time.

$$\vec{Sp}_t = \alpha\vec{p_t} + (1-\alpha)\vec{Sp}_{t-1} \tag{2.1}$$

$$\vec{Sp}_t^{\,[2]} = \alpha\vec{p_t} + (1-\alpha)\vec{Sp}_{t-1}^{\,[2]} \tag{2.2}$$

Based on these two equations, it is possible to calculate the estimates $\vec{b_0}$ and $\vec{b_1}$ for corresponding ground truth $\beta_0$ and $\beta_1$ respectively by using 2.3 and 2.4.

$$\vec{b_0}(t) = \frac{\alpha}{1-\alpha}(\vec{Sp}_t - \vec{Sp}_t^{\,[2]}) \tag{2.3}$$

$$\vec{b_1}(t) = 2\vec{S_{p_t}} - \vec{S_{p_t}}^{\,[2]} - t\vec{b_1}(t) \tag{2.4}$$

Following these estimations, the user's position prediction at time $\tau$ into the future is given by 2.5.

$$\vec{p_{t+\tau}} = \vec{b_0}(t) + \vec{b_1}(t+\tau) \tag{2.5}$$

The position prediction equation is defined by 2.6.

$$\vec{p_{t+\tau}} = (2 + \frac{\alpha\tau}{(1-\alpha)})\vec{S_{p_t}} - (1 + \frac{\alpha\tau}{(1-\alpha)}\vec{S_{p_t}}^{\,[2]}) \tag{2.6}$$

To predict user orientation, the same formulation prediction can be applied using orientation values instead of position values. Thus, $p$ is substituted by $q$ in equation 2.1, 2.2, 2.3, 2.4, 2.5 and 2.6, defined as each individual quaternion component. DESP will always follow a trend in the actual data, since it will adjust the next forecast by at most a certain percentage of the last error of the last error (Makridakis et al, 1983, p. 87).

## 2.5 Deep Learning Algorithms for head motion prediction

This chapter presents the main elements of deep neural networks and the methods for training large neural networks. Recurrent neural networks are described and fundamental techniques for training and regularizing deep neural networks are outlined.

In the last decade, Recurrent Neural Network (RNN) algorithms have been adopted for motion prediction of 3D sequences as in [12, 37]. To understand RNN's, it is necessary to understand the broader concept of machine learning (ML). ML refers to a range of techniques that aim to allow statistical models and algorithms to obtain information from given data [18]. The data sets typically involved are large, making computations very computationally intensive. Thus, manual computation are not a viable option, giving ML its name. There are different types of machine learning, but this thesis is only about supervised learning instead of unsupervised learning. Supervised learning teaches the algorithm to learn the dependencies of the data based on a set of labeled data points, while in unsupervised learning, the data is unlabeled and the algorithm must learn its structure independently without a preconceived definition [30]. The goal of a machine learning algorithm is to find anomalies. However, there are many different types of anomalies and it would be difficult to represent them all without having a definition of an anomaly. Therefore, the algorithm used in this thesis learns the structure and dependencies from the input data that best includes all anomalies. The networks used in this thesis belong to the category of ANNs that is a set of statistical models inspired by neurological processes [30]. These networks consists of artificial neurons, of which each receives inputs and weights, which are then combined into a weighted sum and passed to the activation function that determines the output. Formally,

$$y = f(\sum w_n x_n + b) \tag{2.7}$$

where x is the input, n is the number of inputs, w is the weight and b is the bias term. Each neuron produces an output that fires on to the next neuron if an activation function f(x) activates the output. Examples for activation functions are Rectified Linear Unit (ReLU) and tanh, all used in this thesis. Artificial neurons consist of different layers. Each network has at least one input, one hidden and one output layer. Each layer can have different numbers of neurons, which increases the number of parameters to be trained

and thus the complexity of the network. The term "deep neural network" intends multiple hidden layers piled one after another [27]. A popular activation function used in the output layer in prediction tasks, is a linear function that converts results between zero and infinity. Neural networks can learn the dependencies between inputs and outputs with high accuracy by changing the weights in the equation above. Optimal weighting allows the results to approximate the desired output [30]. Backpropagation is responsible for weight adjustments during the training process, which aims to increase the accuracy of the neural network by computing the gradient of a loss function. To evaluate predictions of a neural network a loss function compares the output generated to the expected output. The algorithm then uses the output to calculate the loss function's gradient regarding the neurons' parameters in the neural network's layers. The gradient specifies the function's fastest direction and magnitude of increase. The calculated value of the gradient at a given point, allows conclusions on how to change the neuron weights to increase the accuracy of the model [19]. The weights change slowly through the training process and approximate values enhance the model's accuracy. The iteration of weight changes by calculating the gradient of the loss function over all layers gives backpropagation its name. The order of the inputs is important in time series prediction, such as in modeling head motion predictions. RNN's have been developed to model situations like these, where the sequence of the events is important.

## 2.6 Training Methods

The training process of a neural network requires to use an optimization algorithm to find a set of weights to best map inputs to outputs. An objective function or a loss function, allows to find the error between the real data dimensions and the values obtained by the model. Once the loss is computed, the backpropagation algorithm efficiently computes the objective function's gradient or derivatives. A gradient-based optimizer then utilizes the gradient information to adjust the internal weights to minimize the loss of the objective function. This subchapter introduces fundamental concepts in relation to training a neural network.

### Loss function

The objective function $\mathscr{L}(y,\hat{y})$ captures the error of the network by comparing the real values y and the predicted $\hat{y}$. Regression tasks often use the mean absolute error (MAE) loss or the mean squared error (MSE) as

evaluation metrics, but a regression for pose estimation requires a geometric loss function to learn the position and rotation of the joints from each other. An algebraic-based loss function fails to consider that a small error in a crucial joint can drastically affect the pose error. Previous work [22, 51] uses various approaches, such as the Euclidean distance between each predicted joint position and the reference pose.

$$\mathscr{D}_{euc} = \sqrt{\sum (y_i - \hat{y}_i)^2} \tag{2.8}$$

Euclidean distance is well suited for position estimation, but it is no exact distance function between two rotations. Instead, the distance between two rotations is geometrically interpreted as the geodesic distance between two points on the unit sphere. The geodesic distance (or shortest path) is the arc angle between two viewpoints, exponential form.

$$\mathscr{D}_{geo} = ||log(\hat{y_i}^t * y_i)|| \tag{2.9}$$

,where $||.||$ presents the L1-norm.

## Gradient based optimization

Training a machine learning model seeks to minimize the loss in order to obtain the optimal parameters. Among these optimal parameters are weights and biases. The loss refers to an objective function. The loss function is usually optimized using a gradient-based optimization algorithm. Given training data D and a set of trainable parameters $\theta$, the final loss of the model can be defined as:

$$L(\theta; \mathscr{D}) = \sum \mathscr{L}(\hat{y}, y) \tag{2.10}$$

Two of the most popular learning algorithms at this time are the Stochastic Gradient Descent [42] (SDG), and Adaptive Moment Estimation [23] (Adam), which uses both momentum and adaptive learning rate.

## Back-Propagation Algorithm

Concerning the model parameters, gradient-based algorithms are being used to update the model parameters by optimizing the loss function. In 1986, one of the most efficient techniques for computing gradients was the back-propagation algorithm [54]. The algorithm is designed to effectively train a neural network following a method called the chain rule. In simple terms, back-propagation performs a backward pass after each forward pass

across that network and thereby modifies the parameters of the model. Applying the back-propagation algorithm, it is possible to transfer the gradients from the last to the first layer. As a result, the derivatives of the neuron's trainable parameters are calculated for the loss function.

### Dropout regularization

Among the most popular strategies to enhance the performance of the model in the presence of unseen data is the concept of dropout [3]. It is based on the premise of avoiding training all neurons in the network at each epoch, and thereby reducing over-fitting and increasing speed. In more technical terms, at any given training stage, single nodes are either dropped from the network with probability 1-p or kept with probability p, so that a reduced network remains; incoming and outgoing edges to a dropped node are also removed. As the weights of the network increase due to the failures, so if a neuron with probability p is retained, the weights of a neuron are multiplied by p, the final predictions can be interpreted as the mean of remaining network that were previously trained and re-scaled.

## 2.7 Recurrent Neural Networks

RNN's are a type of deep neural network specifically designed for use with time series data. RNN's are a popular choice in the field of Natural Language Processing (NLP) or time-dependent movements, that also includes head movements. It exists a temporal correlation between the previous value in the sequence and the next value in the sequence, when using temporal data. An RNN is recurrent by design because it performs the same operation for each data input, while the output of the last input is based on the previous operation of the computation. Unlike feedforward neural networks, RNNs are able to leverage their internal state (memory) to deal with sequences of inputs. With this ability, RNNs are suitable for tasks such as seizure detection, languange processing or head motion prediction. In other neural networks, all inputs are independent. In an RNN, however, all inputs are linked. Different sub-types exist, which use different methods, but most have some sort of internal memory or state to remember the information and reuse it when the next input value in the sequence is received [34].

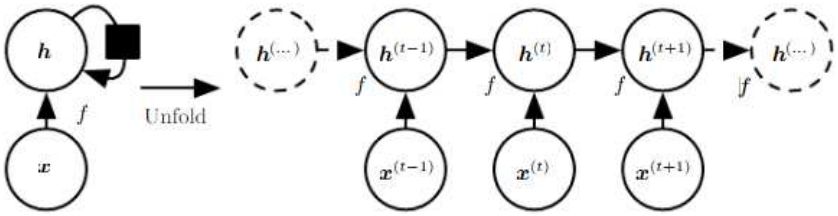



Figure 2.3: A recurrent network with no outputs adopted by [18]

RNNs are particularly sensitive to vanishing gradients, which means that the neuronal activation will saturate after several steps, and the magnitude of the gradient vector ends up very close to or equal to zero. Thus, no further learning can occur. The following subsections describe extensions of the basic RNN that are robust to this problem.

## Gated Recurrent Unit

The main difference between plain RNN's and Gated Recurrent Unit's (GRU) is the use of the latest state supports the gating of the hidden state. This infers unique processes for determining whether a hidden state should be updated and at what step it should be set back. For example, if the first data point is important, it will not revise the hidden state after the initial data point. Similarly, irrelevant temporary observations will be skipped. It has proven to have comparable performance to its more complex variant, the LSTM, but is simpler to implement and faster to compute [6].
To begin with, the reset gate and the update gate must be implemented. Both gates are vectors with entries in the range (0,1) to perform a convex combination. For example, with a reset gate in place, it is possible to control how much the previous state is left to remember. An update gate can control how much of the new state is just a copy of the old state.

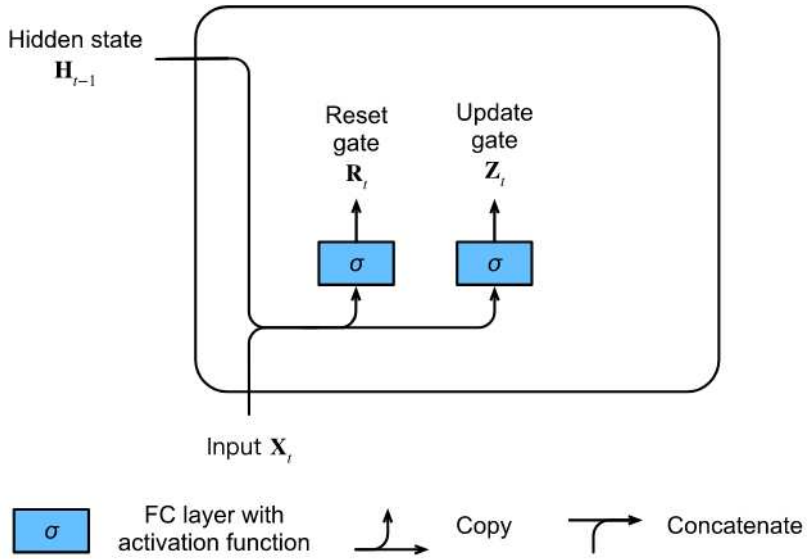



Figure 2.4: Reset gate and update gate in a GRU model adopbted by [18]

Figure 2.6 illustrates the input to the reset and update gates in a GRU. The input gate considers the current data input, while the hidden state controls the effects of the previous data input. Two fully connected layers with the sigmoid activation function have control over the output of both gates. Formally, for a timestep t, assume that the input is a minibatch $X_t \in R^{nxd}$ (number of examples: n , number of inputs: d) and the hidden state of the previous time step is $H_{t-1} \in R^{nxh}$ (number of hidden units: h). Thereafter, the reset gate $R_t \in R^{nxh}$ and the update gate $Z_t \in R^{nxh}$ are defined as follows.

$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_h r + b_r) \tag{2.11}$$

$$Z_t = \sigma(X_t W_{zr} + H_{t-1} W_h z + b_z) \tag{2.12}$$

Here $W_{xr}, W_{xz} \in R^{dxh} and W_{hr}, W_{hz} \in R^{hxh}$ are defined as weight paramters, whilst biases are defined as $b_r, b_z \in R^{1,h}$. Next, let's integrate the reset gate Rt, which leads to the following candidate hidden state $H_t \in R^{nxh}$ at time step t.

$$\widetilde{H_t} = tanh(X_t W_{xh} + R_t \oplus H_{t-1}) W_{hh} + b_h) \tag{2.13}$$

Similiar as in previous equations $W_{xh} \in R^{dxd} and W_{hh} \in R^{dxh}$ are defined as weight parameters, whilst biases are defined as $b_h \in R^{1xh}$, and $\oplus$ is the element wise product operator symbol. In this case a tanh function has been used in the hidden state in order to remain in the interval (-1,1). The

result is a candidate, but the action of the update gate remains open yet. The procedure after implementing the reset gate can be seen in Figure 2.6.

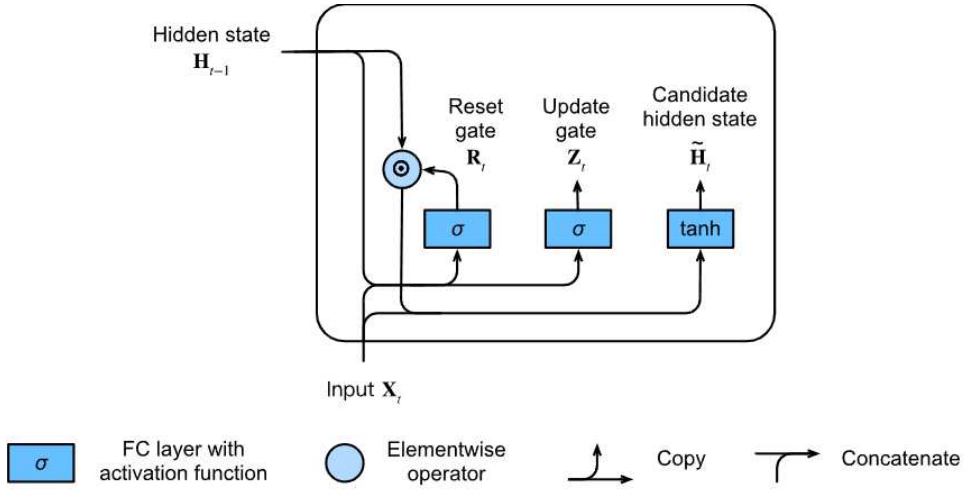



Figure 2.5: Computational flow after reset gate application in a GRU model adopted by [18]

Within the last step, the state of the update gate $Z_t$ is included. It determines to what extent the new hidden state $H_t \in R^{nxh}$ is just the old state $H_{t-1}$ and to what extent the new candidate state $H\ t$ is used. The update gate $Z_t$ can be used for this purpose by simply forming element-wise convex combinations between $H_{t-1}$ and $H\ t$. A final update equation for the GRU is obtained as a result:

$$H_t = Z_t \oplus H_{t-1} + (1 - Z_t \oplus \widetilde{H_t} \tag{2.14}$$

At any point that the update gate $Z_t$ is approaching 1, the old state is simply retained. In this case, the information from $X_t$ is essentially discarded, allowing the time step t in the dependency ladder to be skipped. On the other extreme, when $Z_t$ is close to 0, the new latent state $H_t$ becomes closer to the candidate latent state $H\ t$. These designs can help provide a better way of dealing with the vanishing gradient problem in RNN's and of understanding dependencies for sequences with large timestep intervals. For example, if the update gate was nearby 1 for all time steps of an entire subsequence, the old hidden state is easily retained in the time step of its beginning and propagated to its end, no matter the length of the subsequence.

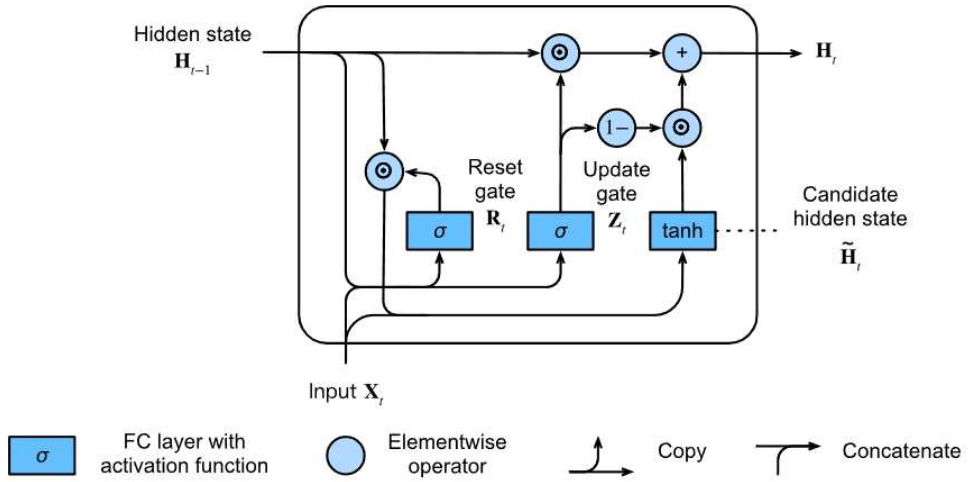



Figure 2.6: Computation of the hidden state in a GRU model adopted by [18]

To sum up, the Gated RNN's can more effectively detect dependencies in sequences with long time step intervals. Reset gates help detect short-term dependencies in sequences. Update gates provide the ability to capture long-term dependencies in sequences. GRUs include base RNNs as an edge case when the reset gate is enabled. In addition, GRUs can skip partial sequences by switching on the update gate.

## 2.8 Rotation formalism's

The human head movement is represented as a sequence of human poses. Each pose can be characterized by joint body positions or characterized by joint rotations. For motion prediction, it is possible to use different formalism's to predict rotations with alternative advantages and trade-offs.
There has been substantial research to enhance state of the art, and several rotational formalisms are to address various challenges. For example, the widely used Euler angle representation does not appear to be well adapted to regression tasks in general, as this representation is fraught with singularities. Instead, some research has used quaternions for regression that are free of singularities but have the problem of antipodes. Also, any rotation representation in 3D with less than five dimensions is discontinuous, complicating learning. This chapter will briefly discuss different rotation formalism's and their problems that might affect the regression performance of deep learning models.

### Euler Angles

Euler angles can describe each rotation by three successive elementary rotations about three independent axes. In accordance with the coordinate axis around which the Euler angle rotates, there are three different functions for calculating the elementary rotation matrices. The process of identifying the correct order becomes tedious and error-prone when working with multiple data sources that contain angular information, especially when working with small rotations since, in this case, all rotation sequences look similar.

### Quaternions

Quaternions are a compact representation method of rotations. They can be interpreted as a scalar component and a three-dimensional vector component with four parameters. Quaternions are common to be preferred in computer graphics for computationally efficient rotation and interpolation. Unlike Euler angles, quaternions are free of the gimbal lock problem but still have an ambiguity caused by their antipodal symmetry: q and -q correspond to the same rotation. An attempt could avoid this ambiguity and restrict the quaternions to one hemisphere by restricting one scalar component to a positive value.

### Rotation Metrices

Rotation matrices are continuous representations. However, it is necessary to enforce their unique orthogonal properties for using rotation matrices for regression. Using the Gram-Schmidt method, an orthonormal basis of two vectors can be formed. Saxena et al. [43] noted that the Euler angles and Quaternions create learning problems due to discontinuities. However, there were no general rotational representations other than direct regression of 3x3 matrices in their study.

# Chapter 3

# Previous research

To re-call, the purpose of this thesis is to mitigate latency in XR applications by predicting head motions in the immediate future (100 millieseconds). It is important that the design of the predictor is in a resource efficient manner. Resource efficiency in this context is defined as user specific data that is simply recorded during the gaming session. Head motion prediction is a research field that has multiple overlaps with other research areas, such as body motion prediction. Being of the data type time series, traditional machine learning problems such as classification tasks diverge significantly as compared to the problem statement of this thesis. Nevertheless, a substantial effort of research has been done on topics that revolve around the problem statement of this thesis. This chapter briefly reviews previous work on predicting future head motion in XR.

Earlier head motion prediction techniques were developed mainly to compensate for the rendering and display delays of the first AR systems. Azuma [2] designed an AR system using head tracking data to reduce the systematic response time of M2P latency. The findings of his work show that the prediction is most effective for short prediction intervals, which are less than 80 ms. Azuma followed this up, and Bishop [53] also presented a frequency-domain analysis of head motion prediction and concluded that the error in predicted position increases with higher prediction intervals and frequencies of the head motion signals increase rapidly. Van Rhijn et al. [49] suggested a framework for Bayesian predictive filtering algorithms and evaluated the influence of the filter parameters on the prediction performance. To this end, a comparison was made between the achievements of several predictive techniques using artificial created and experimental data. Kraft [25] suggested a quaternion-based UKF that extends the original UKF formulation to account for the properties of unit quaternions that

are inherent in them. The designed filter was employed to predict the head orientation. However, the assessment is restricted to simulated motions. Himberg and Motai [20] suggested the adoption of an EKF on the basis of quaternion modification between adjacent time steps (delta-quaternion), demonstrating that in fact their solution has equivalent forecasting ability to a quaternion-based EKF, while incurring lower computational costs. La Viola [14] gave a direct comparison with the unscented Kalman filter (UKF) and the extended Kalman filter (EKF) for the prediction of head and hand orientation by using quaternions.

With the rising interest in XR in recent years and 360 degree videos, head motion prediction gained new importance for predicting the user's future field of view . Bao et al. [4] suggested regressions-based prediction to forecast the field of view of a user. The same models were also applied to forecast the size of the borders around the field of view for more efficient transmission of 360 degree videos. Sanchez et al. [9] suggested prediction of angular velocity and angular acceleration to overcome the difficulty of delay streaming. With the advances of HMD's, new sensor data were also available. This made sensor fusion between two data types possible such as gaze and head tracking data. Xu et al. [50] explored the integration of gaze for head motion prediction. By proposing saliency maps of gaze history and images of VR content into a Convolutional Neural Network network (CNN) for feature extraction. To do so, they use a long short term memory (LSTM) to encode the scan path history. Subsequently, they merge the CNN features and LSTM features for predicting the gaze shift between gaze point at a current time and the gaze point at a future time. Qian et al. [41] evaluated the accuracy of different head motion prediction techniques based on machine learning with the use of head motion traces from 130 different users, depending on the prediction interval. The resulting prediction method was made part of a streaming system for 360 degree videos to boost bandwidth utilization and increase video quality while maintaining bandwidth.

The concepts of head motion prediction for 360 degree videos and XR volumetric videos are based on similarities in terms of algorithms and methodologies. It is, however, volumetric videos that are known to be more complex and enable motion in more degrees of freedom. Thus making prediction more difficult. This thesis focuses on joint prediction of translational and rotational head motion, prediction of head motion in 6DoF space. There are also some studies in this area, but the studies are still very scarce. In [16], Gul et al. introduced a Kalman filter for making head motion predic-

tions suitable for cloud-based volumetric video streaming. Despite delivering high-resolution 3D Content to any device with satisfactory Internet access speed, server-side rendering systems have suffered from interaction latency. While the research results were encouraging, the authors pointed out that additional research is essential to investigate shortcomings that evolve around spherical size prediction and more accurate head orientations.

Head orientation predictions are a broader part of body motion prediction, which is an ongoing research interest itself and goes beyond the focus of XR applications. Current state-of-the-art techniques for body motion generation are generally data-driven and based on Deep Learning [32, 39, 51]. Zhou et al. [32] suggested a modified training procedure for recurrent neural networks to generate a human motion with more excellent long-term stability, while Pavlo et al. [39] proposed separate short-term and long-term recurrent motion predictors using quaternions to more adequately express body rotations. Sequencing head rotations and head positions are done using an RNN, as in [14, 37]. However, GRU architecture has proven to be superior[10].

**Key-take-aways**

The review of previous research indicates that head motion prediction was used to leverage early VR applications. Classic machine learning models and smoothing techniques are used for this purpose. With the increasing interest in 360 degree videos and the technical advances of HMD's, Deep Learning and Sensor Fusion are also being adopted. The most recent research efforts are in the area of 6DoF head motion prediction to enable XR applications. Previous research is still in its early stages in this domain and lacks, for that matter, comparisons on the performance of Deep Learning models and Sensor Fusion for 6 DoF to classical Machine Learning and Smoothing algorithms.

# Chapter 4

# Data and Methods

To predict head motions of the user's head pose data is a complex task, which can be divided into three major steps. The first step is to construct a dataset. When training machine learning algorithms, it is essential to have training data from which the model can be learned. It is in the interest of the training process to be successful that the training data correspond as closely as possible to that of the target of interest. This work investigates head motion prediction in real-time interactive graphics applications. To this end, instead of using open-source datasets that are not fully indicative of interactive VR applications, this thesis collected user data from a real-time graphics applications. The second step is to train a machine learning model that uses position as well as rotation data of the user's head. The acquired data then can be used to develop, implement and train an algorithm. Such process is iteratively and will be determined by a converging loss. The final step evolves around model evaluation and includes to test and discuss the performance of the model. Figure 4.1 illustrates the entire process, which will be presented in detail in this chapter.

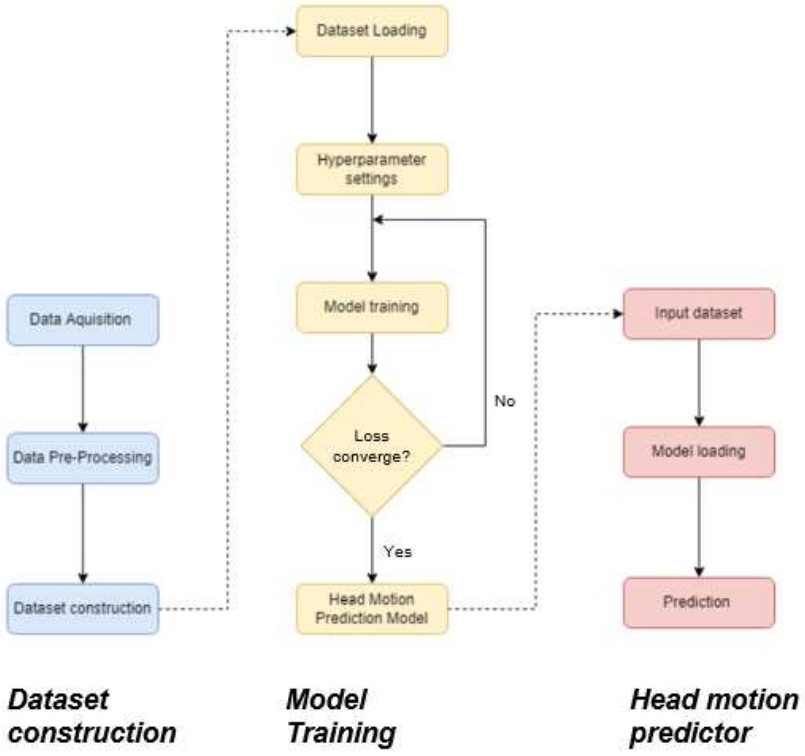



Figure 4.1: Solution overview

Section 4.1 outlines the data collection and experimental setup. Section 4.2 provides the steps to be followed to convert the collected data into a structure that can be used to train the machine learning models. Section 4.3 introduces the model architecture used and justifies its choice, it also explains the training process to find the optimal parameters used to build a predictor.

## 4.1 Data Collection and Experimental Setup

By utilizing Unity it is possible to export data of the user's head movement during gaming sessions. Most commonly among these are the user's head orientation and position. The objective of this thesis is to predict in a resource efficient manner head motions in the near future. Resource efficiency is given when the corresponding ML model only use input data that comes from consumer-grade hardware. Besides, the models should be able to produce reliable head motion predictions in commercial VR applications. Therefore, the decision was made to use training data acquired

during a real-world gaming session, instead of synthetic data. As such, a set of mini-games with different types of user interaction called Nvidia VR FunHouse was used. A typical scene during data collection can be seen in Figure 4.2.

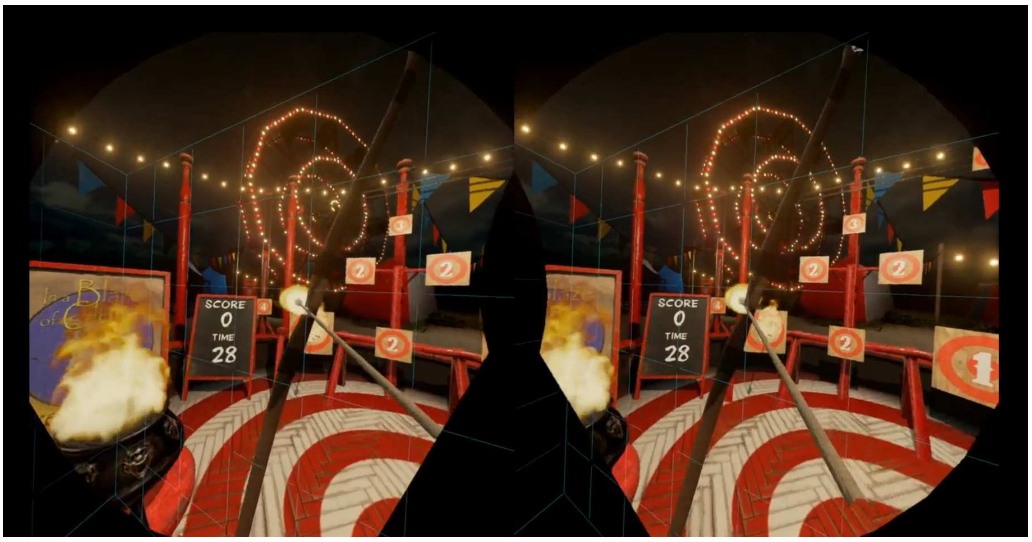



Figure 4.2: Sreenshot

In addition, the recording device is a commercially available HMD, and a commercially available application is used for stimulus presentation. An application for data acquisition that can be utilized in Unity was developed in previous work by Lehtiranta, K. (2021) [31]. The HMD used was a HTC Vive [10], which has two with a resolution of 1080x1200 pixels each and a refresh rate of 90 Hz. The data acquisition happens at a frequency of 60 occurrences per second, which corresponds to 60 rendered and displayed frames per second. Such frequency rate is typically the default refresh rate of screens and thus a good balance between resource requirements and providing accurate training data. The entire procedure and set-up aims to replicate a typical use case of interactive VR content and minimize the collection of biased data.

The data collection is limited to the user's pose and head orientation data. Considering that virtual reality content is usually experienced using an HMD, the acquisition of pose data does not have hardware requirements that are beyond the reach of the average consumer if the developed models are to be used in a commercial remote VR solution. As such, the output the raw data is defined as a pose p given by a HMD position x and orientation

represented by a quaternion q:

$$p_t = [x_t, q_t] \tag{4.1}$$

Each pose p is defined at each timestep t of the corresponding data set and relative to an arbitrary global reference frame which is given by the data set used. The choice to collect quaternions is motivated by the fact that its arbitrary 4-D values can be conveniently represented to legitimate rotations by normalizing them to a unit length. Furthermore, quaternions are not bounded to discontinuities and singularities, are more numerically stable and computationally more efficient than other representations [39], such as Euler angles. Euler angles have been used widely for the representation of joint rotations [17]. Their advantage is that an angle can be defined for each degree of freedom, so that they can be easily fit to the degrees of freedom of real human joints. Nevertheless, Euler angles also have the problem of not being unique ( $\alpha = \alpha + 2\pi$, both representing the same angle), discontinuity in the representation space, and singularities (gimbal locking) [15]. It can be shown that all representations in 3D are subject to these problems, including the popular exponential maps [15].

A number of 5 participants in total took part in the data collection experiment. Participants were familiarized with the objective of the data collection and made comfortable with the HMD, controllers, and the controller and eye tracker. Each participant was given time to practice the game mechanics, the VR hardware, and the calibration routine of the eye tracker. When ready, participants were again briefed on the calibration procedure and instructed to stop the experiment if they felt uncomfortable. The HMD was then placed on the head and the game and data acquisition application were initiated. The participant then completed the calibration routine of the data collection application before starting the game. A session was ended either by the participant's decision to end it or when a participant had played a full session of the game. A full overview of the data collected can be seen in Table 4.1

Table 4.1: Data acquired during experiment

| Participant | Participant Duration in h |
|---|---|
| 1 | 00:10:39 |
| 2 | 00:04:58 |
| 3 | 00:13:07 |
| 4 | 00:19:49 |
| 5 | 00:05:55 |
| **Total** | **00:54:28** |

The raw data captured during data acquisition is not suitable for use with current machine learning technologies. For example, by looking at the raw data given in Figure 4.3 it is getting obvious that the rotation trace is discontinuous, making it hard for a predictor to learn.

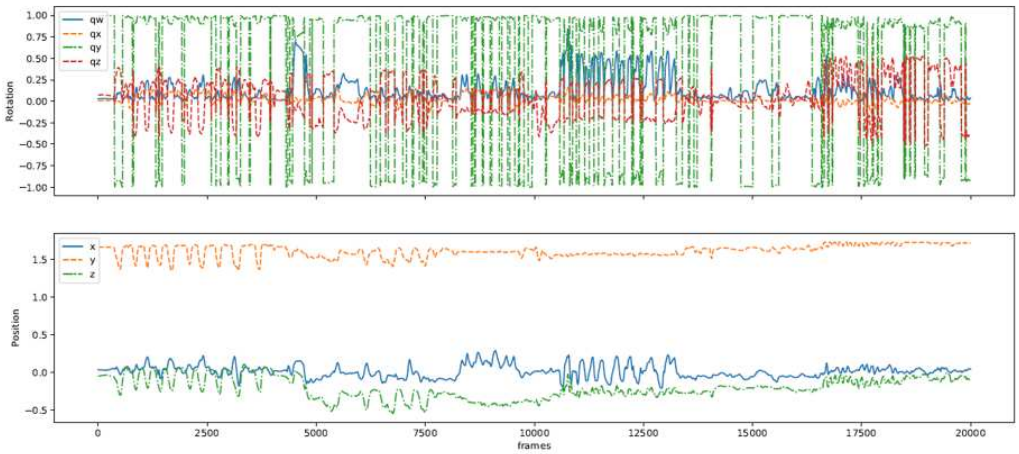



Figure 4.3: Example Trace, illustrating the position and rotation of a participant during data acquisition

Thus, certain transformations and pre-processing steps must be carried out to ensure that the captured data can be used in corresponding machine learning models. These pre-processing transformations are described in detail in the following section.

## 4.2 Data Pre-Processing

The entire pre-processing pipeline is illustrated in Figure 4.4 and will be further explained within this section.

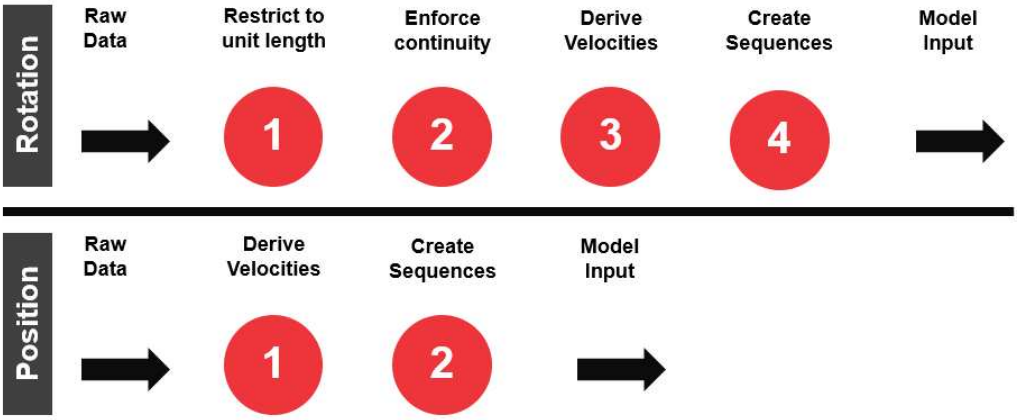



Figure 4.4: Pre-Processing Pipeline for rotation and position

First, the application of quaternions requires a few necessary steps: in order to represent valid rotations, rotations must be normalized to a unit length. An additional important point to consider is that if q denotes a particular orientation, then -q (antipodal representation) represents the same orientation. The two representations are blended into one data set, leading to discontinuities in the time series as can be seen in Figure 4.3. For each orientation at time t, there is continuity enforced by using the representation with the smallest geodesic distance from the representation in the previous frame t-1. To illustrate the effect of enforced continuity, Figure 4.5 shows the raw and continuous data for a time window of 3000 frames. After the transformations, the trace is smoother and eases statistical learning.

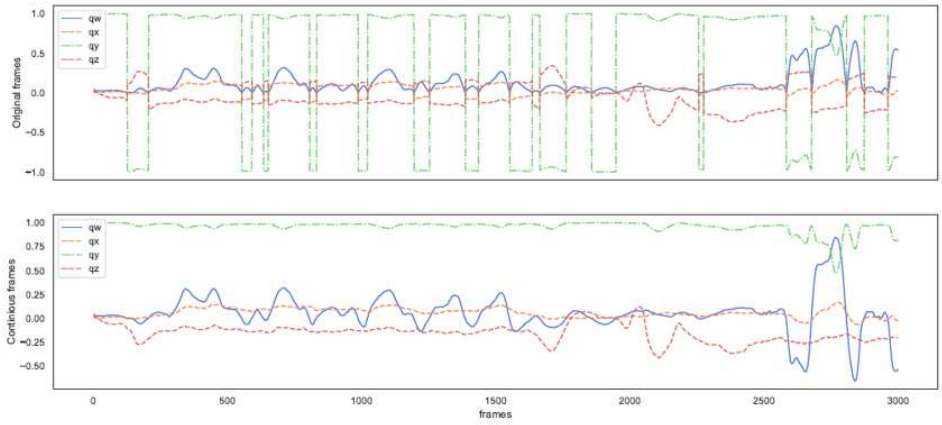



Figure 4.5: Raw and continuous trace for a time window of 3000 frames

For both positions and rotations, one can predict velocities (referred to as deltas, with respect to time, in the following) instead of absolute values [37, 48]. The density of velocities is centered on a smaller scale of values, which eases statistical learning. Nonetheless, in practice, velocities are prone to be unstable in long-term tasks and perform worse in generalization due to accumulation errors. Noise in the training data is also a concern with velocities, because invalid poses cause large errors that can result in unstable models. As such deltas with respect to time for position and rotation were obtained by using the following equation.

$$\Delta q = q_i * q_{i+1}^{-1} \tag{4.2}$$

$$\Delta x = x_i - x_{i+1} \tag{4.3}$$

In all experiments, a sequence architecture is being used. Therefore the raw data has to be split into sequences. As can be seen in Figure 4.6, the raw data is being transformed into sequences with 30 time steps each. However, for training purpose, rotations and positions were processed separately from each other in sequences and fed individually into the ML model.

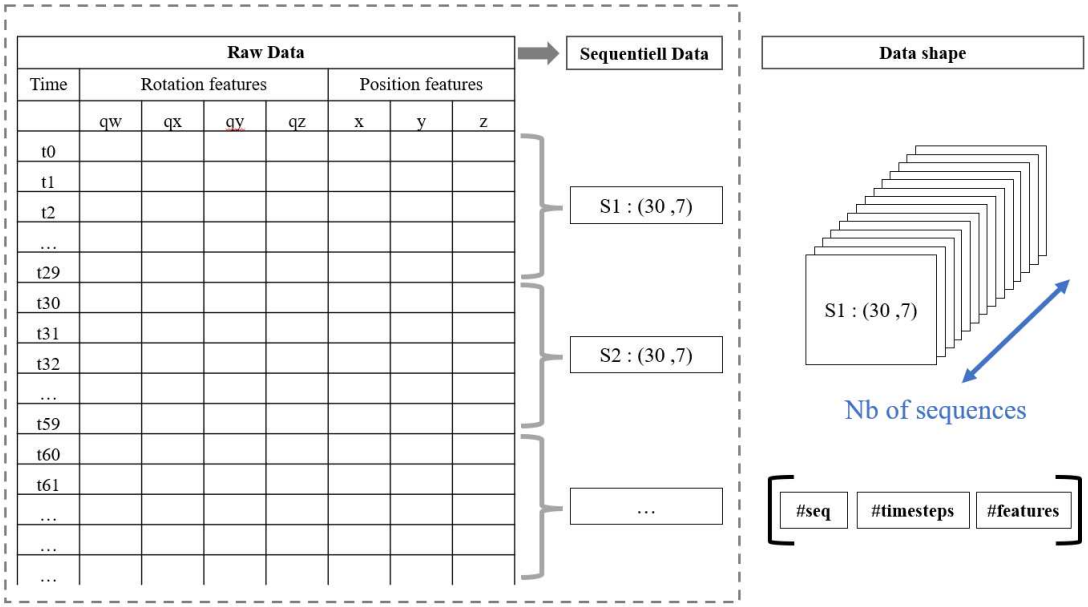



Figure 4.6: Illustration of sequence split

For training and validation purpose, the available data was split into training, validation and testing datasets. A total of five traces with different people were recorded, giving a total duration of 54:28 minutes. All participants played the virtual reality game Nvidia VR. One trace was obtained for training and validation dataset, whilst the other traces were used for testing. It is important to note that the quality of the trace had an impact on the performance of the ML model. Quality in this context is defined as sufficient coverage of movements in all directions. For further information about the recordings, see Table 4.7.

| | Data | | Duration (h) |
|---|---|---|---|
| **Participant 1** | **70% Training Data/** 24.464 sequences | **30% Validation Data/** 10.464 sequences | 00:10:39 |
| **Participant 2** | **Test Data/** 14.847 samples | | 00:04:58 |
| **Participant 3** | **Test Data/** 44.543 samples | | 00:13:07 |
| **Participant 4** | **Test Data/** 69.631 samples | | 00:19:49 |
| **Participant 5** | **Test Data/** 19.964 samples | | 00:05:55 |

Figure 4.7: Illustration of data split

## 4.3 Model Architecture and Training

As pointed out in chapter 3 the use of RNN has proven to show good performance in modeling human motions. In this thesis, a GRU architecture has been chosen. A GRU is an autoregressive model, thus at each time step, the model uses the previous recurrent state and features used to characterize the previous pose as input to predict the following pose. Similar to Martinez et al. in [37] , the choice of a GRU architecture is motivated by its simplicity and efficiency. Consistent with the results of Chung et al in [6], there have been no advantages in using long short memory (LSTMs) that require additional gates to be learned. Unlike Martinez et al [37], adding a second recurrent layer was found to be of advantage, yet not for a third one. The two GRU layers each consist of 32 hidden units whose initial states h0 are learned from the data. Figure 4.8 shows the architecture of the pose network, which is used for both position prediction and rotation prediction.

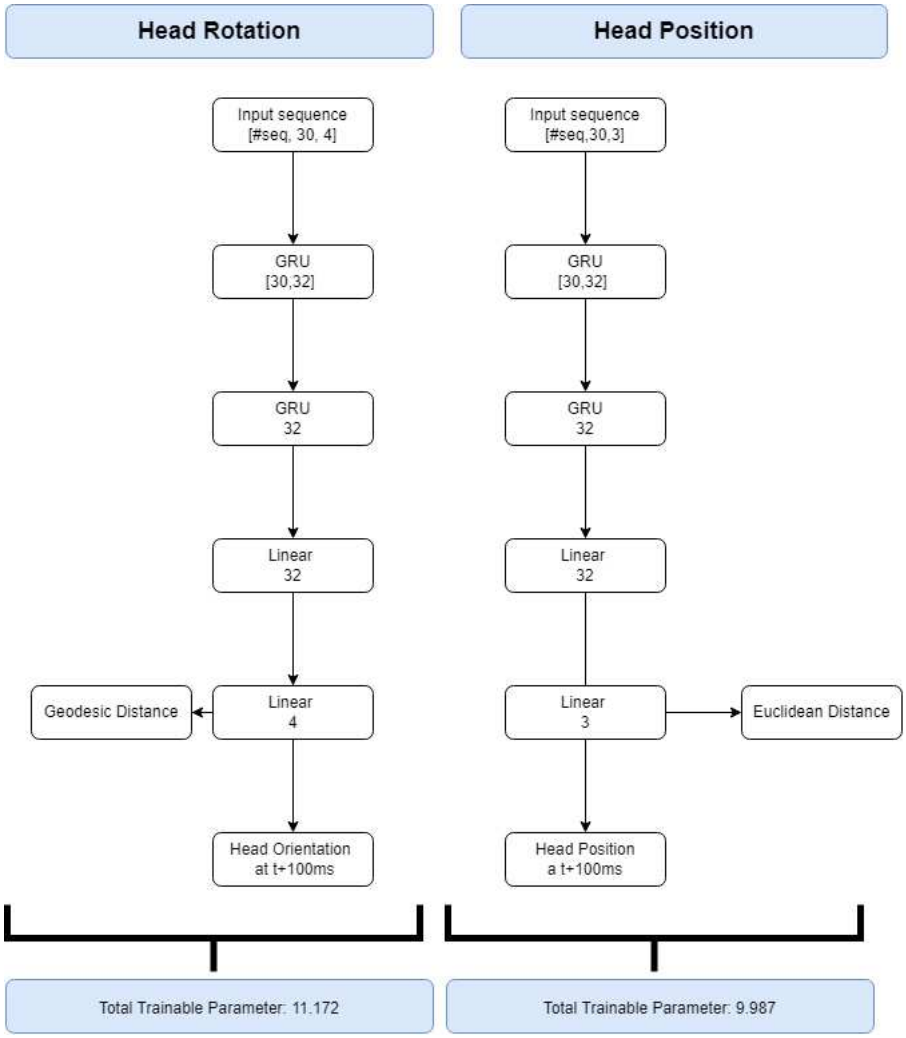



Figure 4.8: Architectures for Head Position and Head Rotation Prediction

The network takes as input the rotations of the head motions (encoded as unit quaternions, a choice justified previously, and is trained to predict the future states of the head over k time steps. . The position predictor is trained by the Euclidean Distance loss function, while the rotation predictor is trained by the geodesic distance loss function.

$$d_{euc} = \sqrt{(\hat{x} - x)^2 + (\hat{y} - y)^2 + (\hat{z} - z)^2} \tag{4.4}$$

$$d_{geo} = ||ln(q^{-1} * \hat{q}|| \tag{4.5}$$

The Adam optimizer [23], which prunes the gradient norm to 0.1 and decays the learning rate exponentially with a factor alpha = 0.999 per epoch,

is used for learning. The entire implementation is done using Tensorflow. In order to find the optimal parameters, hyperparameter testing has been performed.

Table 4.2: Training parameter

| **Parameter** | **Rotation** | **Position** |
|---|---|---|
| Neurons | 32 | 32 |
| Batch size | 128 | 128 |
| Initial learning rate | 0.001 | 0.001 |
| Dropout rate | 0.5 | 0.1 |
| Loss function | Geodesic Distance | Euclidean Distance |
| Optimization algorithm | Adam | Adam |
| Beta1 | 0.9 | 0.9 |
| Beta2 | 0.999 | 0.999 |
| Decay rate | 0.01 | 0.01 |
| Epsilon | 0.00000001 | 0.00000001 |
| Epochs | 20 | 20 |

# Chapter 5

# Evaluation and Results

The method explained in the previous section are tested to assess the ability to predict head motions. In this chapter, the evaluation criteria used to determine the performance of the underlying machine learning model and the results of the machine learning model are presented as well as discussed.

## 5.1 Evaluation Metrics

To asses the performance of the machine learning model, two objective error metrics were employed - the position error and rotation error. The position error is the Euclidean distance (in meters) between the actual and the predicted position. Rotation error is the spherical distance (in rad) between the actual and the predicted rotation. After computing the position and rotation error for each time step, the mean absolute error (MAE) over a dataset is computed as:

$$Position - MAE(d_{euc}) = \frac{1}{N}\sum d_{euc_i} \tag{5.1}$$

$$Rotation - MAE(d_{geo}) = \frac{1}{N}\sum d_{geo_i} \tag{5.2}$$

## 5.2 Benchmark Performance

To analyze the performance of the developed models against existing solutions, they are compared to a state-of-the-art head motion predictor and a baseline. The thesis considers an agnostic zero-velocity baseline that

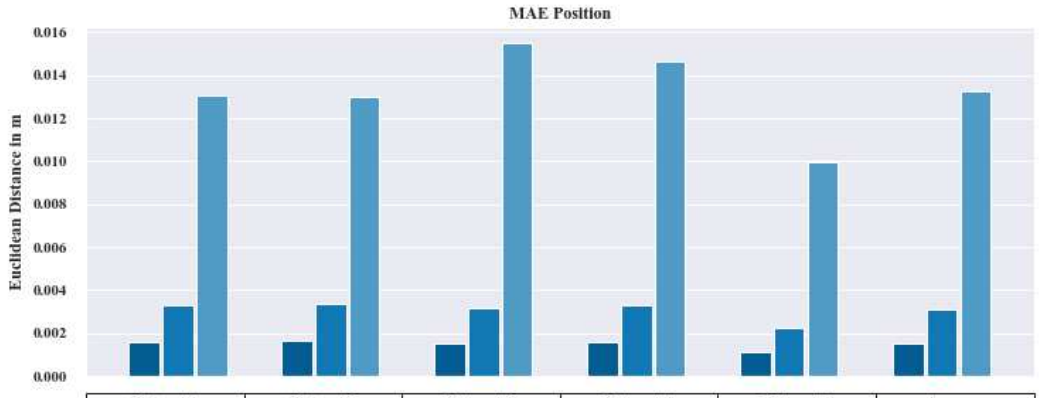
in each instance predicts the most recently observed value of the previous sequence. Next, a comparison is made with the Double-Exponential-Smoothing Method (DESP) method [29], which is well established and popular in the industry. The choice of DESP is motivated based on the fact that it is comparatively easy to implement and provides similar results as the state of the art method of a Kalman filter [29].

## 5.3 Results

The models developed in this thesis perform better than the two baseline models, with the GRU model performing the best for position and rotation. As the result show, the proposed position and rotation model outperform both the naive baseline model, which uses the most recent available head pose, and the current state-of-the-art method DESP in predicting future head pose's. The position model achieves a mean absolute error of 0.00149m, which is a significant improvement compared to the DESP model and to the baseline model in terms of mean absolute error delta. The rotation model achieves an mean absolute error of 0.00428 rad and thereby also outperforms the DESP model and the baseline.

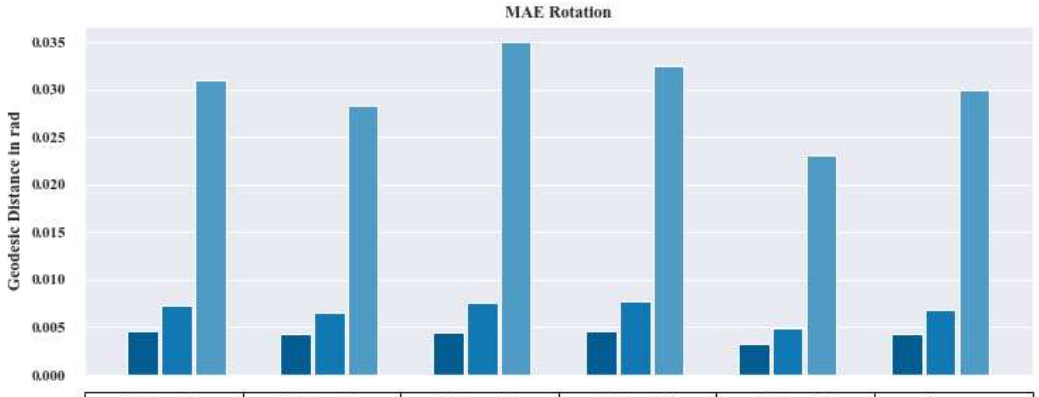
Table 5.1: Average Position and Rotation Error - Comparison

| | **Position Error in m** | **Rotation Error in rad** |
|---|---|---|
| **GRU** | **0,00149** | **0,00428** |
| DESP | 0,00307 | 0,00684 |
| Baseline | 0,01324 | 0,03004 |




| | Dataset 1 | Dataset 2 | Dataset 3 | Dataset 4 | Dataset 5 | Average |
|---|---|---|---|---|---|---|
| GRU | 0.00156 | 0.00167 | 0.00152 | 0.00161 | 0.00109 | 0.00149 |
| DESP | 0.00327 | 0.00338 | 0.00314 | 0.00331 | 0.00225 | 0.00307 |
| Baseline | 0.01309 | 0.01297 | 0.01547 | 0.01467 | 0.00999 | 0.01324 |

Figure 5.1: Mean Absolute Position Error




| | Dataset 1 | Dataset 2 | Dataset 3 | Dataset 4 | Dataset 5 | Average |
|---|---|---|---|---|---|---|
| GRU | 0.0046 | 0.0044 | 0.0045 | 0.0046 | 0.0033 | 0.00428 |
| DESP | 0.00724 | 0.00658 | 0.00768 | 0.00775 | 0.00497 | 0.00684 |
| Baseline | 0.0311 | 0.0284 | 0.035 | 0.0326 | 0.0231 | 0.03004 |

Figure 5.2: Mean Absolute Rotation Error

Using the position and rotation error to measure the accuracy of the models does not provide a sufficient basis for claiming that the developed GRU position and rotation model outperform the DESP and Baseline. For this reason, the boxplots with whiskers and cumulative distribution functions (CDFs) were used to visualize the statistical properties of the position and rotation error.

The statistical properties of the error are better described by a boxplot. Boxplots for each of the models and the baselines are shown in shown in Figure 5.3 for position error and in Figure Figure 5.4 for rotation error. The orange line represents the median of the error, while the box surrounding the median line extends from the first quartile value (Q1) to the third quartile value (Q3) of the data. This means that 75% of the data are below the third quartile line. The interquartile range (IQR) is defined as the distance between the first and third quartiles. The upper whisker will extend to the last datapoint less than 99% of the data. The lower whisker will extend to the first datapoint greater 1% of the data. Beyond the whiskers, data are considered outliers. For the position error as well as for the rotation error, the GRU model has a lower spread of outliers. The DESP approach performs generally better then the baseline, but is unable to outperform the GRU across all datasets - for rotation and position. Looking at the boxplots illustrates very convincingly the superiority of the GRU model compared to DESP and baseline. Nonetheless, furhter insights are missing about the error distribution. A CDF plot will further allow insights on the distribution of the error, that will improve a better understanding on how the GRU model is capable to predict headmotions.

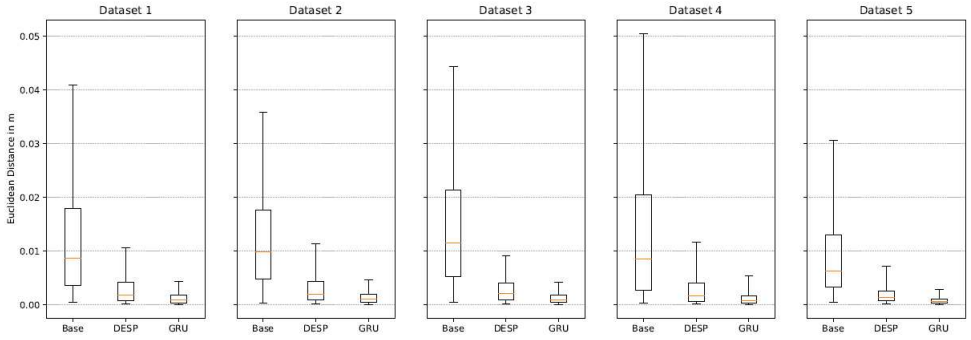



Figure 5.3: Boxplot Position Error. Whiskers represent the 1-99% range.

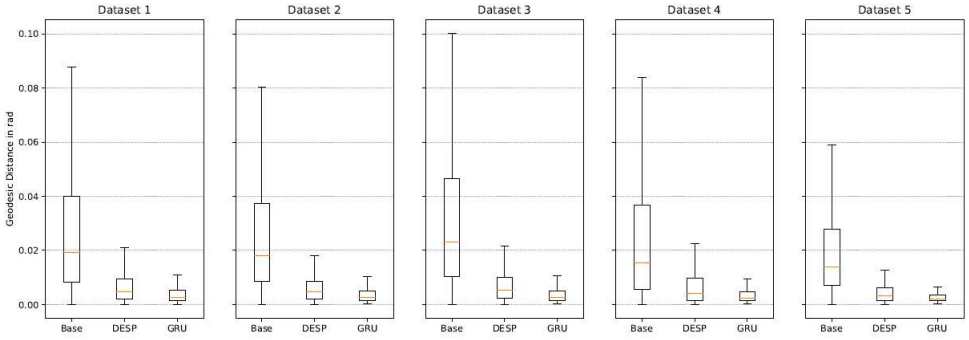



Figure 5.4: Boxplot Rotation Error. Whiskers represent the 1-99% range.

CDF plots for all datasets and models are shown in Figure 5.5 with plotting CDF plots for rotation on the left and CDF plots for position on the right. A CDF illustrates the proportion of position and rotation errors that are less than a specified value. Figure 5.5 reveals that over 80% of the head position predictions have an error of less than 0.002 m and over 80% of the head rotation predictions have an error of less than 0.005 rad. Even the Mean Average for the DESP and Baseline across all datasets (Table 5.1) are above these values. This again shows how much better the GRU model is able to predict head movements.

Nonetheless, the CDF plots reveal an interesting insight about the performance of the GRU model. By comparing the GRU and DESP CDF curves, it shows that the DESP achieves a lower minimum error rate for rotations across all datasets, resulting in an intersection of both curves. For this reason, the marginal cases are examined in more detail below to better understand the performance of the GRU model.

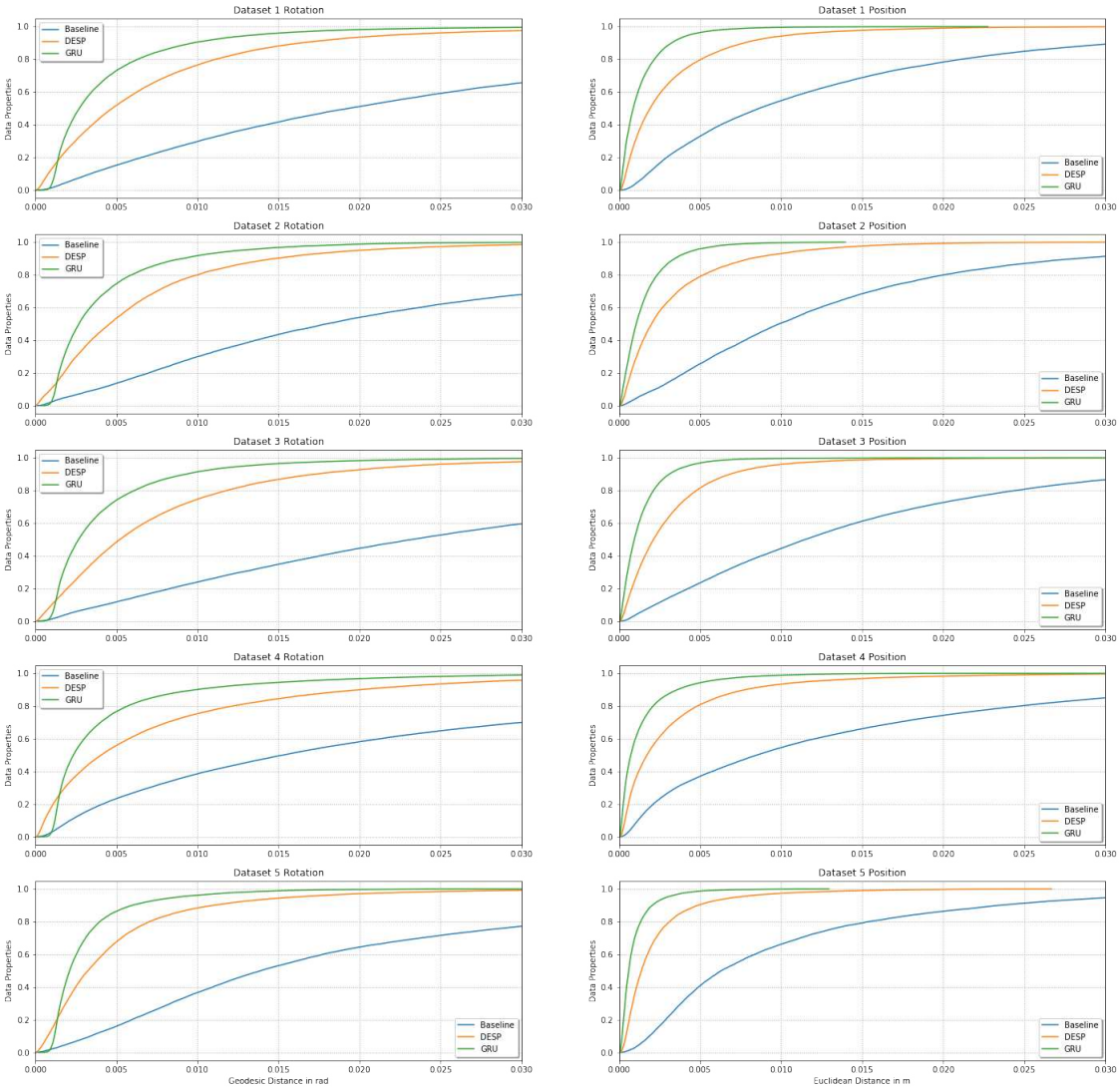



Figure 5.5: Cumulative distribution functions of the position and rotation errors

Table 5.3 and Table 5.2 shows the minimum and maximum error for position and rotation across all models for an example trace. Interestingly the DESP and Base model have both a smaller minimum error compared to the GRU. This indicates room for improvement on the loss function, which is responsible for the learning rate. A possible solution could be to penalize values higher then one and lower then minus one in order to ensure that prediction remain a unit quaternion. Moreover, it is interesting to observe that the index for rotation as well as for position minimums diverges across all models. In the contrary for the rotation and position maximums, which

are all located around a similar index.

Table 5.2: Three largest and smallest rotation errors with index

| | | Rotation Maximum | | |
|---|---|---|---|---|
| | | **GRU** | **DESP** | **Base** |
| **1** | **value** | **0.043755** | **0.065234** | **0.231098** |
| | *frame index* | *6749* | *6774* | *6749* |
| **2** | **value** | **0.042894** | **0.063435** | **0.227044** |
| | *frame index* | *6746* | *6773* | *6752* |
| **3** | **value** | **0.040660** | **0.060693** | **0.226985** |
| | *frame index* | *6753* | *6772* | *6748* |

| | | Rotation Minimum | | |
|---|---|---|---|---|
| | | **GRU** | **DESP** | **Base** |
| **1** | **value** | **0.000360** | **0.000064** | **0.000278** |
| | *frame index* | *7614* | *4952* | *8327* |
| **2** | **value** | **0.000499** | **0.000081** | **0.000511** |
| | *frame index* | *11066* | *5660* | *8328* |
| **3** | **value** | **0.000508** | **0.000084** | **0.000517** |
| | *frame index* | *1855* | *3930* | *8326* |

Table 5.3: Three largest and smallest position errors with index

| | | Position Maximum | | |
|---|---|---|---|---|
| | | **GRU** | **DESP** | **Base** |
| **1** | **value** | **0.013955** | **0.030863** | **0.075220** |
| | *frame index* | *3318* | *3352* | *3321* |
| **2** | **value** | **0.013899** | **0.029536** | **0.074853** |
| | *frame index* | *9911* | *9940* | *3322* |
| **3** | **value** | **0.013899** | **0.029331** | **0.074824** |
| | *frame index* | *3314* | *3348* | *3320* |

| | | Position Minimum | | |
|---|---|---|---|---|
| | | **GRU** | **DESP** | **Base** |
| **1** | **value** | **0.000018** | **0.000062** | **0.000088** |
| | *frame index* | *5623* | *11949* | *11911* |
| **2** | **value** | **0.000020** | **0.000064** | **0.000106** |
| | *frame index* | *11894* | *5009* | *2494* |
| **3** | **value** | **0.000026** | **0.000064** | **0.000152** |
| | *frame index* | *4747* | *5059* | *4990* |

In order to draw further conclusions from the results, the following section compares the individual components of rotation and position with the corresponding error rates. The hope is to understand if there are certain movement patterns that make it difficult for the GRU model to learn. As the format of this thesis does not allow to plot larger charts, the following charts can be found within the appendix:

- Figure 8.1 : Entire example trace: Position components and error comparison
- Figure 8.2 : Entire example trace: Rotation components and error comparison
- Figure 8.3 : Window of example trace that shows maximum position error
- Figure 8.4 : Window of example trace that shows maximum rotation error

Figure 8.1 and Figure 8.2 reveal that position and rotation errors are the smallest, when little to no movement occur. It is also interesting to observe

that the qz component of the quaternion seems to be stronger correlated to the rotation error, as both curves are similarly spiking from frame index 8.000 - 10.000 with qz having the strongest magnitude from 0.4 to -0.6 compared to qw,qx and qy component. Interestingly similar patterns can be seen within the position, the y component seems to be stronger correlated to the position error, as both curves are similarly spiking from frame index 8.000 - 10.000 with y having the strongest magnitude from 1.3 to 1,7 compared to x and z component. In order to further investigate the behaviour of the maximum position and rotation error, that occur often within the same frame index, Figure 8.3 and Figure 8.4 visualize the maximum position and rotation error across all models. The baseline and the DESP are performing in similar manner. Both models follow the trace. Whilst the baseline is overshooting the ground truth, DESP makes serpentines around the original data series and thus has many intersections. One possible reason for this is that the DESP is updated from time step to time step always following the actual trend of the ground-truth. The GRU has less intersections with the ground-truth, but is always closer to the ground-truth compared to the DESP and Baseline, which aligns with the lowest MAE.

To conclude, the GRU models outperform DESP in predicting future head movements by a significant margin. The GRU models also outperform the baseline significantly, demonstrating the effectiveness of the GRU models. The presented models are constructed in a way that they can be extended in the future. The current models consist of different pipelines that process different data modalities, which are then combined in the later parts of the models. If another data type correlates with head pose, a pipeline that processes the input data of that data type can be added to one of the models of the models and then combined with the output data from the pipelines that processed the other data modalities.

# Chapter 6

# Discussion

This chapter discusses some limitations of the current research project, presents future research opportunities and evaluates the contribution of this work.

## 6.1 Limitations

**Data:** Training was performed on a limited dataset from VR Mini games. Since the differences between these datasets may not be significant to capture all and enough movements, this may have hindered the learning of the model. For example, in these VR Mini games, there are limited changes in body position, whereas in actual XR applications this would be the case. Therefore, a more robust model could be trained on positional training data if it were available.

**Model and Algorithm:** In this study, very simple model architectures were used, therefore the number of layers and the number of filters per layer were kept minimal. While this has the effect of making the model very robust, the architecture is also one of the most important factors for good performance. Choosing more complex architectures in conjunction with adding more layers, more filters per layer, might have improved performance. For example, each component could be trained in a model and the results merged in the last layer. Also, testing the models for different time windows has not been done, which would improve the understanding of the GRU performance.

**Evaluation:** Another limitation of this thesis is the quality of the evaluation. The evaluation was mainly based on Rotation Error and Position

Error, as it is sometimes used in the literature. However, the use of this metric does not allow for comparability to results from other publications using the same metrics. There is a lack of comparability that can only be achieved by evaluating across common datasets.

## 6.2 Suggestions for future work

**Data:** Based on the first limitation, using larger datasets would most likely improve the models. All models were trained and tested with a fairly limited amount of training and validation data trained and tested. The variance of content in real-time interactive graphical applications is quite large and may contain head pose patterns that the proposed models have not learned and therefore cannot well be able to predict either. Future work could include the collection of additional data data consisting of different types of content, and use this data to train and test the ML models. In addition, open data sets for head motion prediction could be be used for additional training and validation. Transfer-learning could also improve the learning of the models. An other interesting insight would be to explore on which types of virtual reality content the models perform best.

**Model and Algorithm:** The models are currently only predicting head pose positions in the immediate future (100 milliseconds). Potential future work could train re-learning of the models for lengths of time greater than 100 milliseconds and evaluate the performance of the models under these conditions. Utilizing the models to predict gaze positions of the models to predict gaze positions in the future might require reevaluation of the design and implementation of the models as the conditions for the predictions changed. Furthermore, the model architecture could be designed in a way to handle multiple signal. Signal Fusion is an upcoming topic within the research community and gains more and more popularity. For example, gaze data is currently excluded from the learning process. Intuitively, gaze is a good indicator of future head pose because its correlation has already been demonstrated. Future work could integrate gaze data into the prediction of head posture. Moreover, future work could also incorporate various other types of data such as EMG/EEG to further improve the prediction horizon and the corresponding effectiveness of the predictor in this context.

**Evaluation:** It is recommended that ways be found to compare the results from this paper to compare with other publications. This could be done

by training other models with the same data set. However, this may not be desirable. Instead, the current models could be be trained on a task for which benchmark datasets or competitions are available. Evaluation in a fully integrated XR system would be useful, too. Evaluating models in a remote rendering system will likely lead to more relevant evaluation results, e.g., in terms of which LAT and frame rate are acceptable and which are problematic. Also, the performance of the model will be tested when using the model as part of a larger pipeline.

## 6.3 Contribution

Since most current research focuses only on 3DoF head motion prediction, the 6DoF pipeline used in this project, is scientifically relevant. The comparison performed in this work showed how much better the use of Deep Learning models is to conventional smoothing methods for 6DoF. However, it showed that the method for predicting rotations is much more comprehensive and non-homogeneous than originally thought. Thus, there is a clear need for improvement to establish the best possible approach. In addition, this project contributes to the literature by focusing on prediction of head motion in 6DoF, a task that has rarely been studied in the literature as it is comparably new. Since XR applications have considerable and growing relevance, it is necessary to examine the extent to which methods that have proven successful for VR applications also provide sufficient results in this context. For the prediction of head movements, only rotation and position of the head were used in this work. Unfortunately, no further signals (gaze) could be added as originally planned, which shows that the prediction of head movements is not as easy as often (implicitly) assumed. The pre-processing pipeline for rotations is very extensive. The Euclidean distance is often used in the literature to predict rotations. In this work, however, the geodetic distance as a loss function in the context of new training methods was continued.

# Chapter 7

# Conclusion

The objective of this work was to develop and implement a machine learning model capable of predicting future head pose in extended reality content. Another goal was to outperform the current state of the art while reducing resource requirements to the point where the machine learning model could be used as part of a remote rendering server. As described in the task statement, one of the main focuses was the universality of the ML models, i.e., the ability to use the machine learning models in currently available extended reality applications. For this reason, the models were trained using data from such applications, as opposed to data from self-generated, limited content. To achieve this goal of the ML models, the data collection application was designed to collect data when a user experiences off-the-shelf VR content.

As a contribution to this work, two machine learning models were developed and implemented using head position and head rotation. To support the development process of both models a set of pre-processing scripts were developed and implemented to collect raw data from virtual reality users and convert the raw data into a format usable by the implemented models.

As shown in the evaluation results, the developed models meet the goals of the work: the models outperform the state-of-the-art method DESP and the selected baselines in head position and rotation. The models succeed in this while reducing the cost of computational resources and achieve a higher level of general-purpose usability by not using internal data from the virtual reality content experienced by the users.

While the ML models were able to achieve good performance in predicting

head motion, the performance could not be tested in an end-2-end application. This is also the case for the DESP method. It remains to be noted that the baseline method is very competitive and not easy for a model to beat. However, to reduce M2P latency, a model must be able to beat the baseline. Furthermore, it remains to be seen that the Deep Learning methods used are many times more complex and nontransparent in their functionality, compared to simple methods like DESP. Even if the results speak for themselves, it must be investigated in a fully integrated application whether the use of such complex models is justified in this context, or whether the use of simple models leads to the same result. In any case, the results show great potential for further investigation of Deep Learning methods for head motion prediction in the future.

# Chapter 8

# Appendix


Figure 8.1: Entire example trace: Position Components and Error Comparison


Figure 8.2: Entire example trace: Position Components and Error Comparison

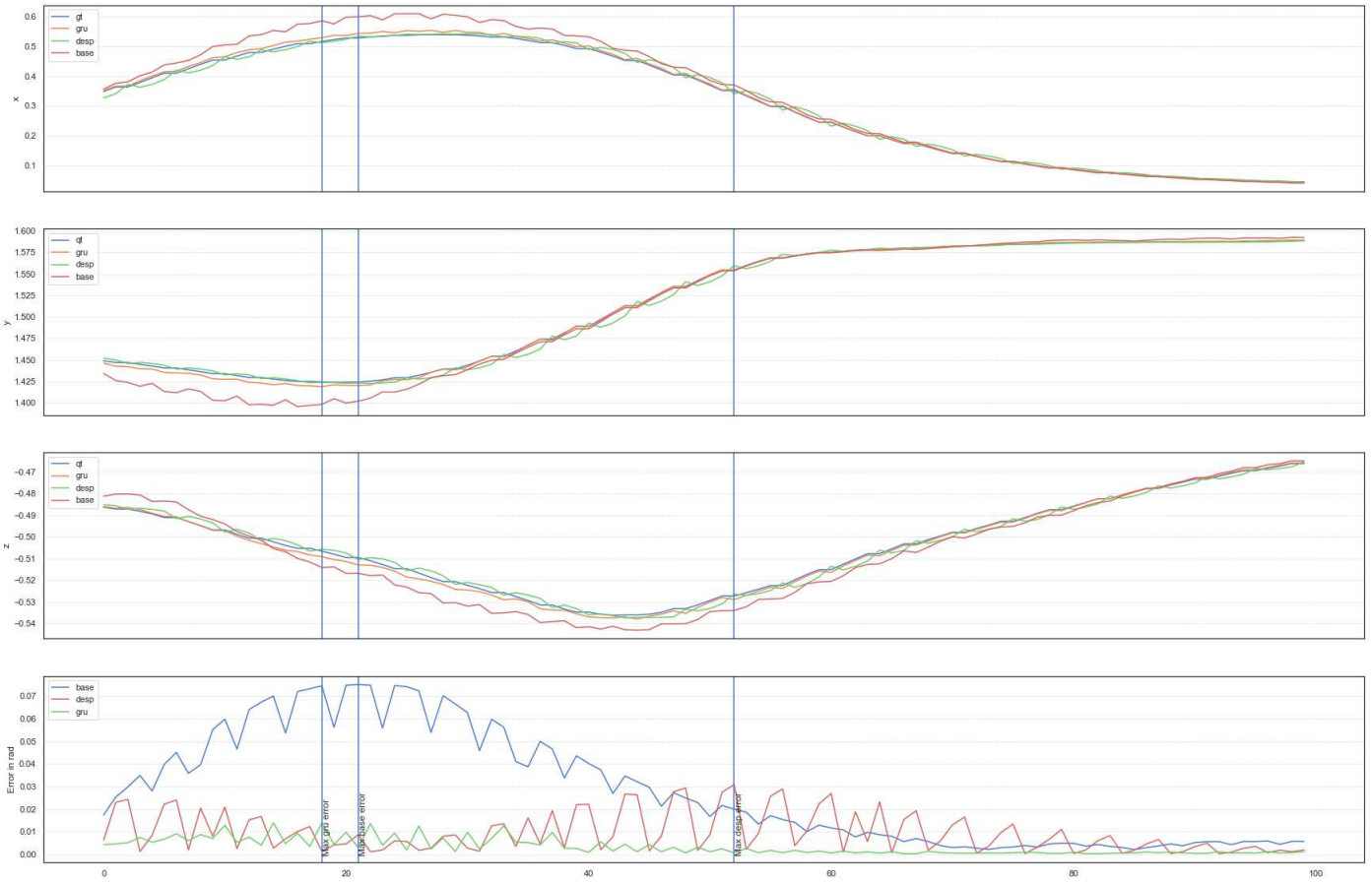



Figure 8.3: Window example trace for maximum position error: Position Components and Error Comparison

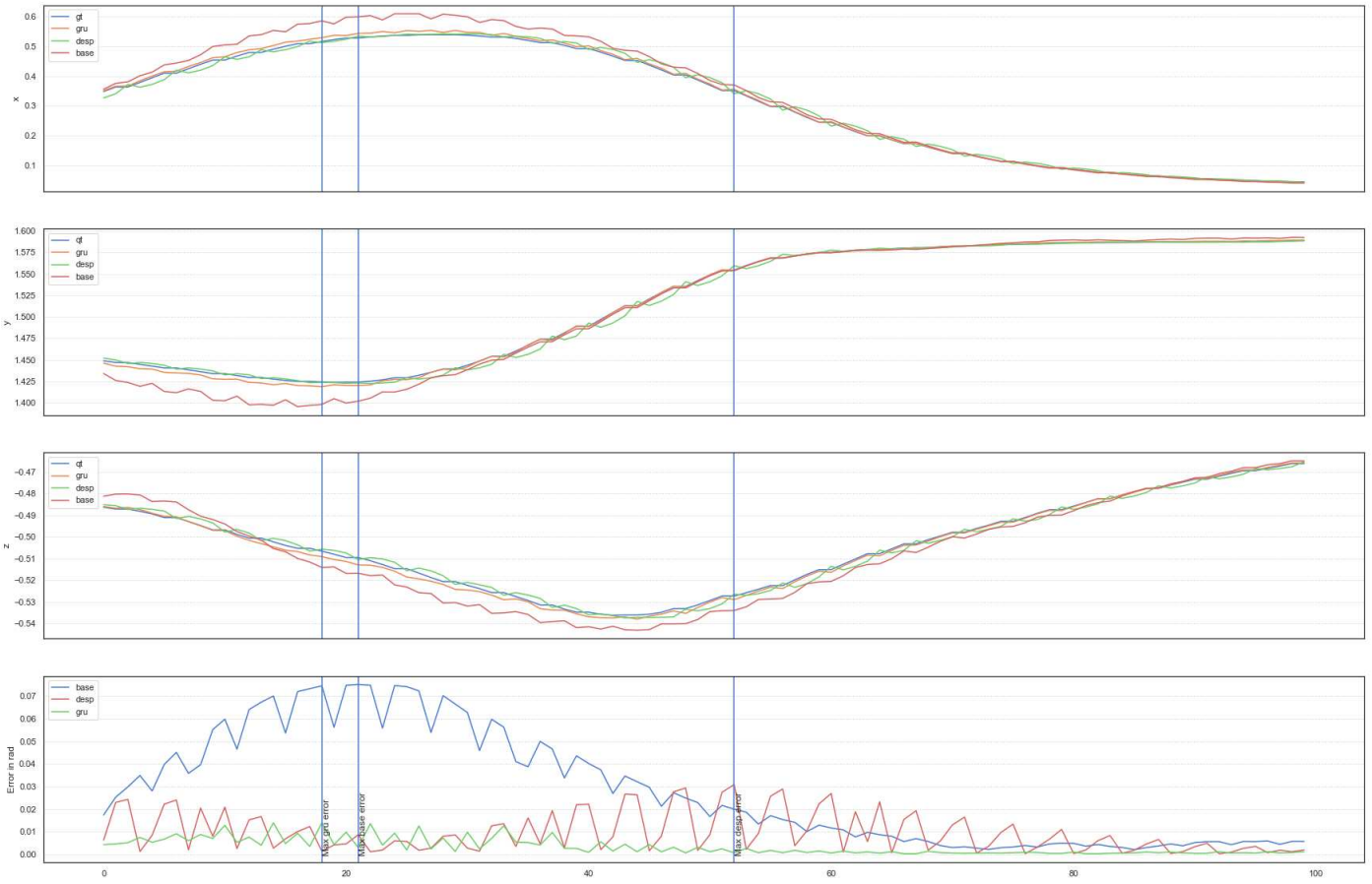



Figure 8.4: Window example trace for maximum rotation error: Rotation Components and Error Comparison

# Bibliography


[1] Allison, R. S., Harris, L. R., Jenkin, M., Jasiobedzka, U., and Zacher, J. E. Tolerance of temporal delay in virtual environments. In *Proceedings IEEE Virtual Reality 2001* (2001), IEEE, pp. 247–254.

[2] Azuma, R., and Bishop, G. Improving static and dynamic registration in an optical see-through hmd. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), pp. 197–204.

[3] Baldi, P., and Sadowski, P. J. Understanding dropout. *Advances in neural information processing systems 26* (2013), 2814–2822.

[4] Bao, Y., Wu, H., Zhang, T., Ramli, A. A., and Liu, X. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *2016 IEEE International Conference on Big Data (Big Data)* (2016), IEEE, pp. 1161–1170.

[5] Beigbeder, T., Coughlan, R., Lusher, C., Plunkett, J., Agu, E., and Claypool, M. The effects of loss and latency on user performance in unreal tournament 2003®. In *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games* (2004), pp. 144–151.

[6] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[7] Clemm, A., Vega, M. T., Ravuri, H. K., Wauters, T., and De Turck, F. Toward truly immersive holographic-type communication: Challenges and solutions. *IEEE Communications Magazine 58*, 1 (2020), 93–99.

[8] Davis, S., Nesbitt, K., and Nalivaiko, E. A systematic review of cybersickness. In *Proceedings of the 2014 conference on interactive entertainment* (2014), pp. 1–9.

[9] de la Fuente, Y. S., Bhullar, G. S., Skupin, R., Hellge, C., and Schierl, T. Delay impact on mpeg omafs tile-based viewport-dependent 360 video streaming. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems 9*, 1 (2019), 18–28.

[10] Dempsey, P. The teardown: Htc vive vr headset. *Engineering & Technology 11*, 7-8 (2016), 80–81.

[11] for Digital Business, H. D. I. Eintauchen in die welt von vr, ar and mr, 2019. WWW page of HWC Digital: `https://www.hwzdigital.ch/eintauchen-in-die-welt-von-vr-ar-mr/`. Accessed 18 Oct 2021.

[12] Fragkiadaki, K., Levine, S., Felsen, P., and Malik, J. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 4346–4354.

[13] Friedmann, M., Starner, T., and Pentland, A. Device synchronization using an optimal linear filter. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics* (1992), pp. 57–62.

[14] Gelfi, S., Stefanopoulou, A. G., Pukrushpan, J. T., and Peng, H. Dynamics of low-pressure and high-pressure fuel cell air supply systems. In *Proceedings of the 2003 American Control Conference, 2003.* (2003), vol. 3, IEEE, pp. 2049–2054.

[15] Grassia, F. S. Practical parameterization of rotations using the exponential map. *Journal of graphics tools 3*, 3 (1998), 29–48.

[16] Gül, S., Bosse, S., Podborski, D., Schierl, T., and Hellge, C. Kalman filter-based head motion prediction for cloud-based mixed reality. In *Proceedings of the 28th ACM International Conference on Multimedia* (2020), pp. 3632–3641.

[17] Han, F., Reily, B., Hoff, W., and Zhang, H. Space-time representation of people based on 3d skeletal data: A review. *Computer Vision and Image Understanding 158* (2017), 85–105.

[18] Heaton, J. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning, 2018.

[19] Hecht-Nielsen, R. Theory of the backpropagation neural network. In *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[20] Himberg, H., and Motai, Y. Head orientation prediction: delta quaternions versus quaternions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39*, 6 (2009), 1382–1392.

[21] Kapanen, S. Optical measurement of virtual reality headset performance.

[22] Kendall, A., and Cipolla, R. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 5974–5983.

[23] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[24] Kiruluta, A., Eizenman, M., and Pasupathy, S. Predictive head movement tracking using a kalman filter. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 27*, 2 (1997), 326–331.

[25] Kraft, E. A quaternion-based unscented kalman filter for orientation tracking. In *Proceedings of the Sixth International Conference of Information Fusion* (2003), vol. 1, IEEE Cairns, Queensland, Australia, pp. 47–54.

[26] Kwok, A. O., and Koh, S. G. Covid-19 and extended reality (xr). *Current Issues in Tourism 24*, 14 (2021), 1935–1940.

[27] LaValle, S., and Giokaris, P. Perception based predictive tracking for head mounted displays, May 24 2016. US Patent 9,348,410.

[28] LaValle, S. M., Yershova, A., Katsev, M., and Antonov, M. Head tracking for the oculus rift. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), IEEE, pp. 187–194.

[29] LaViola, J. J. Double exponential smoothing: an alternative to kalman filter-based predictive tracking. In *Proceedings of the workshop on Virtual environments 2003* (2003), pp. 199–206.

[30] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature 521*, 7553 (2015), 436–444.

[31] Lehtiranta, K., et al. Gaze prediction in vr.

[32] Li, Z., Zhou, Y., Xiao, S., He, C., Huang, Z., and Li, H. Auto-conditioned recurrent networks for extended complex human motion synthesis. *arXiv preprint arXiv:1707.05363* (2017).

[33] Liang, J., Shaw, C., and Green, M. On temporal-spatial realism in the virtual reality environment. In *Proceedings of the 4th annual ACM symposium on User interface software and technology* (1991), pp. 19–25.

[34] Lipton, Z. C., Berkowitz, J., and Elkan, C. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019* (2015).

[35] Livingston, M. A., and Ai, Z. The effect of registration error on tracking distant augmented objects. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality* (2008), IEEE, pp. 77–86.

[36] Mangiante, S., Klas, G., Navon, A., GuanHua, Z., Ran, J., and Silva, M. D. Vr is on the edge: How to deliver 360 videos in mobile networks. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network* (2017), pp. 30–35.

[37] Martinez, J., Black, M. J., and Romero, J. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 2891–2900.

[38] McCandless, J. W., Ellis, S. R., and Adelstein, B. D. Localization of a time-delayed, monocular virtual object superimposed on a real environment. *Presence 9*, 1 (2000), 15–24.

[39] Pavllo, D., Grangier, D., and Auli, M. Quaternet: A quaternion-based recurrent model for human motion. *arXiv preprint arXiv:1805.06485* (2018).

[40] Qian, F., Han, B., Pair, J., and Gopalakrishnan, V. Toward practical volumetric video streaming on commodity smartphones. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications* (2019), pp. 135–140.

[41] Qian, F., Han, B., Xiao, Q., and Gopalakrishnan, V. Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (2018), pp. 99–114.

[42] Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).

[43] Saxena, M., Kumar, R., Jhingan, A., Mandal, S., Stolarz, A., Banerjee, A., Bhowmik, R., Dutt, S., Kaur, J., Kumar, V., et al. Rotational behavior of te 120, 122, 124. *Physical Review C 90*, 2 (2014), 024316.

[44] Schreer, O., Feldmann, I., Kauff, P., Eisert, P., Tatzelt, D., Hellge, C., Müller, K., Bliedung, S., and Ebner, T. Lessons learned during one year of commercial volumetric video production. *SMPTE Motion Imaging Journal 129*, 9 (2020), 31–37.

[45] Schwarz, S., Preda, M., Baroncini, V., Budagavi, M., Cesar, P., Chou, P. A., Cohen, R. A., Krivokuća, M., Lasserre, S., Li, Z., et al. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems 9*, 1 (2018), 133–148.

[46] Shi, S., and Hsu, C.-H. A survey of interactive remote rendering systems. *ACM Computing Surveys (CSUR) 47*, 4 (2015), 1–29.

[47] Shi, S., Nahrstedt, K., and Campbell, R. A real-time remote rendering system for interactive mobile graphics. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 8*, 3s (2012), 1–20.

[48] Toyer, S., Cherian, A., Han, T., and Gould, S. Human pose forecasting via deep markov models. In *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)* (2017), IEEE, pp. 1–8.

[49] Van Rhijn, A., Van Liere, R., and Mulder, J. D. An analysis of orientation prediction and filtering methods for vr/ar. In *IEEE Proceedings. VR 2005. Virtual Reality, 2005.* (2005), IEEE, pp. 67–74.

[50] Xu, Y., Dong, Y., Wu, J., Sun, Z., Shi, Z., Yu, J., and Gao, S. Gaze prediction in dynamic 360 immersive videos. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 5333–5342.

[51] Zhang, H., Starke, S., Komura, T., and Saito, J. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–11.

[52] Zhao, J., Allison, R. S., Vinnikov, M., and Jennings, S. Estimating the motion-to-photon latency in head mounted displays. In *2017 IEEE Virtual Reality (VR)* (2017), IEEE, pp. 313–314.

[53] Zorin, D., and Barr, A. H. Correction of geometric perceptual distortions in pictures. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), pp. 257–264.