Freie Universität Berlin
Fachbereich Mathematik und Informatik
Takustraße 9, 14195 Berlin

MASTER THESIS

USER POSITION PREDICTION IN 6-DOF MIXED REALITY APPLICATIONS USING RECURRENT NEURAL NETWORKS

Oleksandra Baga

Freie Universität Berlin
Matrikelnummer 5480722
Master Computer Science
E-Mail: oleksandra.baga@gmail.com

Prof. Dr. Daniel Göhring
Fachbereich Mathematik und Informatik
Freie Universität Berlin

Prof. Dr. Tim Landgraf
Fachbereich Mathematik und Informatik
Freie Universität Berlin

Statutory Declaration

I herewith formally declare that I have written the submitted master thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper.

I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content.

I am aware that the violation of this regulation will lead to failure of the thesis.

10.10.2022..... Oleksandra Baga

Acknowledgments

This thesis was created in cooperation with the Fraunhofer Heinrich Hertz Institute.

I would like to thank Prof. Dr. Daniel Göhring, who consulted me during the work on a thesis. Also I would like to thank Prof. Dr. Tim Landgraf, who made it possible for me to choose a topic in the field of ML as my master thesis topic.

A special thanks goes to the Dr.-Ing. Cornelius Hellge, heading the Multimedia Communications Group at Fraunhofer Heinrich Hertz Institute, and Serhan Gül, researcher of Fraunhofer Heinrich Hertz Institute, who suggested an exciting topic for a research, which I was allowed to choose for my master thesis.

Contents

List of Figures

Listings

List of Abbreviations

ANN	Artificial Neural Networks
AR	Augmented Reality
CNN	Convolutional Neural Network
CPU	Central processing unit
DoF	Degree of freedome
DL	Deep Learning
FFN	Feed-forward Neural Network
GRU	Gated Recurrent Unit
HMD	Head-Mounted-Display
IEEE	Institute of Electrical and Electronics Engineers
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
RTT	Round-trip time
SDG	Stochastic Gradient Descent
VR	Virtual Reality
3-DoF	Three degree of freedom
6-DoF	Six degree of freedom

1 Introduction

This thesis is focusing on designing and evaluation of the approach for the prediction of human head position in a 6-dimensional degree of freedom (6-DoF) of Extended Reality (XR) applications for a given look-ahead time (LAT) in order to reduce the Motion-to-Photon (M2P) latency of the network and computational delays. At the beginning of the work the existing head motion prediction methods were analysed, and their similarities differences will be taken into account when a proposed Recurrent Neural Network-based predictor will be developed. Main goal is the systematic analysis of the potential of recurrent neural networks for head motion prediction. The proposed approach was evaluated on a real head motion dataset collected from Microsoft HoloLens. Based on a discussion of the obtained results, suggestions for future work are provided.

1.1 Problem statement

The correct and fast head movement prediction is a key to provide a smooth and comfortable user experience in VR environment during head-mounted display (HDM) usage. The recent improvements in computer graphics, connectivity and the computational power of mobile devices simplified the progress in Virtual Reality (VR) technology. The way users can interact with their devices changed dramatically. With new technologies of VR environment user becomes the main driving force in deciding which portion of media content is being displayed to them at any time of interaction with VR Applications [18]. Until recently the high-quality experiences with modern Augmented Reality (AR) and VR systems were not widely presented in home usage and were mainly used in research labs or commercial setups. The hardware for displaying the VR environment was once extremely expensive but recent years became more broadly accessible and the 6-DoF VR headset designed for the end-user were released[1]. It is possible now to experience virtual reality scenes and watch new type of volumetric media at home and the market interest for development VR and AR applications expected to be huge next years.
In fact, the existing on this moment virtual environments can be divided into two main groups depending on position of the user and their ability to move inside the VR environment. The user motion and prediction within a 3-DoF environment

[1] https://medium.com/@DAQRI/motion-to-photon-latency-in-mobile-ar-and-vr-99f82c480926

has been intensely researched for years. Extending such approaches to a 6-DoF environment is not straightforward, due to the change of the user's viewing point from inward to outward and additional three degrees of freedom [19].
Although all mentioned above improvements, rendering of volumetric content remains very demanding task for existing devices. Thus the improvement of a performance of existing methods, design and implementation of new approaches specially for the 6-DoF environment could be a promising research topic.

1.2 Motivation for the research

Research efforts to reduce the computational load are being already wide attempted. However, these approaches designed for the client side. Recently presented technique of the rendering on a cloud server makes possible to decrease the computational load on the client device by offloading of the task to a server infrastructure and than by sending the rendered 2D content instead of volumetric data [12]. The calculated 2D view must correspond the current position and orientation of a user. However, cloud-based streaming approach adds network latency and processing delays due uploading to a server the user position, rendering a new 2D picture from the 3D data and sending it back to a device. Thus, a rendered 2D image can appear even later on a display than with usage of local rendering system.
The promising research topic is a reducing the Motion-to-Photon (M2P) latency by predicting the future user position and orientation for a look-ahead time (LAT) and sending the corresponding rendered view to a client. The LAT in this approach must be equal or larger to the M2P latency of the network including round-trip time (RTT) and time need for calculation and rendering of a future picture at remote server.

1.3 Structure of the thesis

The organization of this thesis is as follows. The thesis starts from introduction and problem statement, followed by theoretical background related to the research topic. Literature review chapter introduces different approaches and technologies of motion prediction algorithms. The chapters **??**nd 4 show the implementation ff the presented models and evaluation of the results that were obtained during experiments. Last, the discussion regarding method limitations and suggestions for the future work are done.

Chapter 1 - Introduction.
The current chapter shortly introduces a state of development on scientific field achieved at a time of master thesis creation in the context on XR applications. The

necessity of timely action to improve the situation with increasing computational and network latency is shown in problem statement section 1.1. Due to the breadth of the research topic, the section 1.2 focuses and motivates the research topic.

Chapter 2 - Background.
The next chapter includes a review of the area being researched. It starts with a short introduction of the concept of MR applications and presents a 6-DoF environment. The presence and influence of a computational and network latency is covered. In section 2.5 the challenges faced in predicting of the viewer's position are discussed. Last section contains an overview of previous research in the field of prediction of user's head position and orientation and places a master thesis's topic in the context of the existing literature.

Chapter 3 - Implementation.
Chapter describes practical implementation of the approach. The dataset including data collection from head mounted display (HDM) and data understanding and preprocessing are described in section 3.1. Model Inputs, Model architecture and the development steps are covered in section 3.2. The implementation of Unity Application, training and evaluation loop with PyTorch and hyperparameter search described in the section 3.2.3.

Chapter 5 - Evaluation.
A Baseline model, used for comparing the obtained results and tuning the hyper-parameters is described. The goal of evaluation and metrics used in this research are covered. The conducted experiments with a data obtained from HMD for each analysed RNN Model can be found in section 4.4.

Chapter 5 - Conclusion.
The last chapter presents a discussion about the limitation of proposed method and provides a conclusion about the received results including suggestions for potential types of future research.

2 Fundamentals

This chapter introduces theoretical background of the presented research problem. First, the concept of mixed reality (MR) followed by an introduction of six degree of freedom (6-DoF) environment and the difference to the three degree of freedom (3-DoF) are described. The term motion-to-photon latency (M2P) is covered, followed by a short discussion about an influence of M2P latency on the decreasing of user experience. The new developed cloud-based rendering and streaming approach is shortly discussed in this chapter. The last section of this chapter highlights challenges with the prediction of viewer's head pose that arises in modern XR applications in connection especially with the added network latency due the using of remote cloud server for computational offload. The section 2.6 overviews the existing works done in the research field using traditional algorithms and recurrent neural networks.

2.1 Mixed reality

Mixed reality makes possible to to break down the border between the virtual and real world. MR provides today an experience that just a short-time ago we could only imagine when watching the sci-fi movies. Realistic images, sounds, and other sensations can be generated by a powerful HMD and projected on transparent holographic lenses giving a user the feeling that virtual objects have size, density, can be picked up and moved to another place.

2.2 Six degrees of freedom

2.3 Motion-to-photon latency

2.4 Cloud-based volumetric video streaming

2.5 Challenges of head motion prediction

All modern HMD has a position tracker, a device or a system of devices, that is responsible for reporting the position and orientation of HMD to the computational unit that generates the virtual environment images displayed in the HMD. These images represent the view that a wearer of HMD would have seen if user was present in VR at the position and orientation reported by position tracker [5].
While the task of position tracking is performed by HMD hardware, the task of position prediction of the movement of human body in the virtual reality remains challenging, and it is still complicate to achieve high-precision estimation. Recurrent neural networks have recently shown promising results in many machine learning tasks, especially when input and/or output are of variable length and are coming as time series with a sequential order. Unfortunately, the known problem of RNN that was observed many years ago by e.g., *Bengio et al., 1994* that it is difficult to train RNNs to capture long-term dependencies because the gradients tend to either vanish (most of the time) or explode (rarely, but with severe effects) [4]. New approaches are needed to be implemented to reduce the negative impacts of this issue. Since traditional recurrent unit overwrites its content at each time-step, a LSTM unit is able to decide whether to keep the existing memory via the introduced gates. The Long Short-Term Memory (LSTM) has a number of minor modifications [7] since it was initially proposed in work [13]. Another approach called a gated recurrent unit (GRU) can adaptively capture dependencies of different time scales without having a separate memory cells [7]. These two approaches can help to find the long-term dependencies in the data obtained from HMD that are otherwise are hidden by the effect of short-term dependencies from the standard RNN models.
Not only the NN architecture is important for high prediction accuracy. Understanding how users interact and behave in AR or VR is a key for preparing the correct dataset when working with HMD's sensors. The experiment done by *Zerman et al., 2021* found out that users preferred to stay in front of static point clouds and 1-1.5 meter away from them and spent more time looking at the frontal view and faces of human models [24]. The navigation trajectories of users within a 6-Degrees-of-Freedom (DoF) should be additionally investigated. An extra level of interaction between user and content is available in 6-DoF environment. The user has now the freedom to change the viewing direction (rotating and translating the head as in 3DoF) but also to change position inside the VR environment [18]. In a 3-DoF environment, users are viewing a portion of the omnidirectional content all the time being positioned at the centre of the spherical content. Thus it is important to understand that a distance between user and content is constant during the interaction [18]. In a 6-DoF, however, the distance changes over time when user moves due to the added degrees of freedom. Thus viewport's center position is not sufficient for tracking the trajectories, the additional metrics such the spatial coordinates and

user orientation are needed to obtain the point of origin. Following [18] *Rossi et al, 2021* authored same year another work [19] dedicated 6-DoF metrics. Researchers proposed to change the video detailing based on user distance to the volumetric object. The closer users are to the volumetric content, the smaller and more detailed is the portion of the displayed content; the farther they are, the bigger but with fewer details becomes the displayed portion [19]. *Rossi et al, 2021* experimented with different metrics to perform clustering in order to detect group of users with similar behavior in VR. The most promising metric seems to be based on the user position on the virtual floor. Metrics based only on viewport center and distance failed in detecting the group of users, which in the ground-truth case form their own cluster, as similar and divided them instead in different clusters [19]. For the trajectory detection best performed a metrics based on user position on the virtual floor, distance and viewport center. Thus not only the way in which users interact within a 3- and 6-DoF environment is fundamentally different. New physical settings and locomotion functionalities given to users also prevent a straightforward extension of current 3-DoF algorithms to 6-DoF [19]. The analysis above leads to the conclusions that prediction of the user's position and orientation on 6-DoF not only a contemporary but also a challenging task that requires new metrics and approaches to be investigated and implemented.

2.6 Related works

This section presents the overview of previous research in the field of the prediction of user position. It includes both approaches for 3-DoF and 6-DoF environment, focusing on time series methods, and overview of different Recurrent Neural Network architectures such as LSTM and GRU.

2.6.1 Traditional prediction algorithms

HMDs now being more widely available to general users last years and the using of algorithms of head motion prediction has been an indispensable component even in 360-degree video streaming systems. A lot of previous approaches uses basic processing of head movement history to predict the future movement, such as simple average, linear regression, and weighted linear regression [16]. Work of *Corbillon et al., 2017* calculates the region that receives from server the video data with a better quality than the remaining of part the video. With simple average researchers determine the distance to the center of viewpoint. The bit-rate of of the biggest part of the delivered video will be thus lower than the original full-quality video because video parts distant from the center will be encoded at low quality [8].
The work of *Duanmu et al., 2017* proposes prediction of the viewing direction for

segment n + 1 through linear regression based on the past 30 view samples [10]. The 360 video is divided into twelve segments horizontally and into three vertically. Thus it is able to obtain the actual rates of the low-quality, medium-quality and the high-quality versions based on calculated position of a chunk n + 1 [10].

Approach of *Xie et al., 2017* uses user's orientation in Euler angle and leverage Linear Regression model to do prediction [22]. The historical samples in window are used to apply Least Square Method and calculate the trends of head movements. The slope over yaw, pitch and roll is denoted and thus an estimated value of yaw, pitch and roll can be predicted using Linear Regression model [22].

The work [20] proposes an adaptive 360-degree video streaming with data segmented into chunks of fixed duration. A representation set for each time-domain chunk is stored on server and clients receives only the data of covered by user's viewport chunk. At each point of time, the client requests chunks which would be played in the future. *Taghavi et al., 2017* use Weighted Linear Regression to predict the next viewport based on window with the last 10 viewport samples obtained with frequency of 10 samples/sec. Researchers mentioned the advantage of robustness of their approach since client can continue playback of at least a low-quality version of the video when the download time of chunks varies [20].

Analysis done by *Qian et al., 2016* indicates that at least in the short term, viewers' head movement can be predicted with accuracy > 90% by even using simple methods such as linear regression [17]. The different approaches were compared such as computing the average value in the window as the prediction result; using the linear regression with all samples in the window having the same weight; and with weighted linear regression when a more recent sample is considered more important. With weighted linear regression the average prediction accuracy for short-term values was higher than 90% across all users. However in the longer term it is more difficult to achieve the good result and the average accuracy drops to about 70% [17].

A method to apply saliency algorithms to VR video viewings was presented by *Aladagli et al., 2017* in work [1]. Cross-correlation analysis used for measuring the relationship between the predicted fixation sequences and the recorded head movements [1]. Based on works mentioned above, *Nguyen et al., 2018* proposed panoramic saliency algorithm with user's head movement history. The eye gaze point is represented by head orientation because the head tends to follow eye movement to preserve the eye position [16]. A head orientation sample stored as a quaternion was converted to a regular 3D unit vector v ($|v| = 1$). Based on head movement velocity and acceleration derived from timestamped head orientation it is possible to find when user head orientation fixates at a specific region for a short period of time. A collection of fixation from multiple users is stored as fixation map.The points from different users are unlikely to match exactly and thus Gaussian Filter is applied to generalize and smooth these scattered user fixation points to a statistical region [16]. Finally, in order to learn the dependence of head tracking logs of multiple

users and multiple saliency maps from the past video frames, the an LSTM network was adopted. Current section addresses using of traditional prediction algorithms on a new media content. The research of previous work in the field of RNN-based architectures is presented in section 2.6.2.

2.6.2 Recurrent Neuronal Networks

In this section the network architectures of most relevant works in the field head motion prediction with RNNs are explained and discussed whether LSTM or GRU architecture is better for the actual problem presented in this thesis. As was mentioned in section 2.6.1, typical HMD computes user positions in 6-DoF by using its tracker module and data comes as time series with a sequential order. The structure of an input is crucial and needed to be followed in order to predict the next future step for a look-ahead time correctly. A sequence of inputs can be processed with Artificial Neural Network (ANN) called Recurrent Neural Network (RNN). Moreover, RNN can processes input with remembering its state while processing the next sequence of inputs. It is known that standard RNN has difficulties to learn long-term dependencies with gradient descent [4]. Though RNN can robustly store information, it yields a problem of vanishing gradient that make leaning difficult [4]. In the last decade, RNN algorithms have been adopted for motion prediction of 3D sequences with long-term dependencies taken into account. For example, the work of *Crivellari et al., 2020* targets traces of tourists in a foreign country and tries to predict the motion of people in the environment they never seen before. LSTM-based model is used thus for analyzing the tourists' mobility patterns [9]. The work of *Yang et al., 2020* proposes using of flashbacks on hidden states in RNNs to search past hidden states with high predictive power. Authors mentioned that their approach gives an ability to determine a specific Point of Interests (POIs) from user's mobility trace stored as a sequence of check-ins in Based Social Networks (LBSNs). Exploring the dataset researchers found sparsity and incompleteness of input sequences and thus the sequential pattern is difficult to be captured by RNNs. They considered temporal distances between similar POIs by flashing back to the historical hidden states sharing a similar temporal context as the current one [23].
The authors *Aykut et al., 2018* claims their research to be first work that applies deep learning for head motion prediction. The current three rotations in three dimensions the so-called Euler angles as well as past values thereof within a certain time window (W) used as input for the network [2]. The best results delivered when 20 last values for each orientation direction (W = 250 ms) were used [2]. Instead of the absolute orientation values *Aykut et al., 2018* suggest for a goal of generalization for other datasets to subdivide the inputs into their respective orientation groups and compute their normalized differences [2]. The authors experimentally confirmed that Feed-forward Neural Network (FFN) indeed had difficulties to learn for different

delays even after an architecture was extended with the present delay and injected into all NN nodes. The using of LSTM-based architectures *Aykut et al., 2018* reasoned with feedback loop and ability to establish a way of memory and share weights over time [2]. Researchers used Adam optimization algorithm, the maximum number of epochs was set to 1000, early stopping technique (patience = 2, min. delta = 0) was used to avoid overfitting. Additionally, the learning rate was decreased by 70% from initial value of 0.001 every 30 epochs. The batch size B was set to 2^{11}. Rectified Linear Unit (ReLU) as activation function for the FFN layers used with LSTM [2]. Three different architecture were tried. In all variants an input for NN was subdivided into dimensions and normalized within time window W set to 250 ms with $\triangle$t = 25 ms. Thus the length of input vector is equal to 10. Final result obtained from each NN variant is a vector of length of 10 for each pan, tilt and roll dimension separately containing future prediction with LAT 1s and step of 0.1s.
Conducted by researchers experiments showed that the LSTM-based architecture leads to a significant improvement of the MAE and RMSE metrics. The best performance is achieved by the interleaved architecture of LSTM and dense FFN blocks [2]. The LSTM-based methods were compared also to widely used approaches like the Linear Regression and a Kalman Filter based optimal state estimate. Thus *Aykut et al., 2018* demonstrated a substantial improvement of the deep predictor for latencies in the range of 0.1–0.9 s [2].
Next year *Aykut et al., 2019* experimented in their work [3] with GRU model that belongs to the group of recurrent neural networks (RNN). Authors mentioned that they already achieved best performance for headmotion prediction with LSTM as described before. However they tried an usage of dense GRUs combined with convolutional components regarding that fact that GRU is computationally more efficient, as it has fewer parameters and states than LSTM units [3]. The GRU-CNN-based network also receives pan, tilt, and roll present and past values within the same W = 250ms for last 20 values. Network trained on the normalized differences instead of using of the absolute values as it was done in [2]. A convolution layer performs as low-pass filter and reduces the noise. The core of the network consists of an interleaved structure of six dense GRUs each with 30 nodes and subsequent convolution units (kernel size 30x30), which compute the most distinct features. Max pooling layer down-samples the output of the last convolution layer and keeps the most significant features. The last step is passing the results through dense FFN and inverse remapping to the absolute orientation values. The model also predicts whole sequence of orientation values with LAT 1s and step of 0.1s as in [3]. The number of epoch in training remains the same as in [2] but early-stopping technique has higher patience parameter and equal 10. Compared to [2] the batch size was reduced to 2^9. The rest of model parameters remains the same as in previous work of *Aykut et al., 2018*. Authors said that proposed GRU-CNN-based network is able to improve the MAE and RMSE compared to all Kalman Filter and described above LSTM-CNN methods, especially for larger delays [3].

Researchers *Karim et al., 2018* developed long short term memory fully convolutional network (LSTM-FCN). In the proposed models, the fully convolutional block is augmented by an LSTM block followed by dropout. The fully convolutional block consists of three stacked temporal convolutional blocks with filter sizes of 128, 256, and 128 respectively [14]. Each convolutional block is identical to the convolution block in the CNN architecture proposed by *Wang et al., 2018* in their work [21]. This means the basic block is a convolutional layer followed by a batch normalization layer and a ReLU activation layer. The convolution operation is fulfilled by three 1-D kernels with the sizes 8, 5, 3 without striding [21]. *Wang et al., 2018* excluded in their FCN any pooling operation to prevent overfitting. Batch normalization helps to reduce generalization error. Global average pooling layer reduces the number of weights. Softmax layer produces the final labels [21]. The LSTM block, comprising of either a general LSTM layer or an Attention LSTM layer, is followed by a dropout. The output of the global pooling layer from FCN output and the LSTM block is concatenated and passed onto a softmax classification layer [14].

The important detail is that FCN block and LSTM block perceive the same time series input in two different views. If there is a time series of length N, the fully convolutional block will receive the data in N time steps [14]. *Karim et al., 2018* sends additionally the time series input into a dimension shuffle layer. Thus LSTM block receives transformed input having N variables with a single time step. Authors say that without the dimension shuffle, the performance of the LSTM block is significantly reduced due to the rapid overfitting of small short-sequence [14].

In work of *Chang et al., 2020* LSTM-based model is proposed to predict displacement of positions in each timestamp given measured acceleration and Euler angles from sensor signals [6]. In addition to standard LSTM networks, they also use bidirectional LSTM (Bi-LSTM) networks, which is stacked two LSTM networks in forward and backward directions. Standard LSTM networks can only consider the past information and Bi-LSTM networks can capture both past and future information by two opposite temporal order in hidden layers [6]. Authors predicted first displacement $x_{t_2} - x_{t_1}$ over the time interval $[t1, t2]$ and used it to obtain the position. Experimentally, authors found that the basic LSTM performs the best comparing to Bi-LSTM and Temporal Convolutional Network. Thus LSTM network with 3 layers used in this paper as the main structure.

The paper [15] also aims for action recognition using sensor signals from HMD. This approach presents another combination of of gated recurrent unit and additional model. Convolutional Auto-Encoder (CAE) model in a clustering fashion learns discriminative feature representation and helps to conquer difficulty of recognizing similar actions. As in the work [6], a deep sequential model learns the temporal representations and predicts the displacement of positions with acceleration and Euler angle to reduce error accumulation. Similar to [6], researches also used additionally a bidirectional LSTM (Bi-LSTM) network. The usage of CAE is proposed as a feature extraction model for feature representation of action signals

and is composed of an encoder and decoder [15]. The encoder contains two 2D convolutional layers with a kernel of size 3×1 with the stride of 1. The decoder contains two 2D transposed convolutional layers with the same kernel size and stride. The bottleneck is three feed-forward fully-connected layers with neurons of 1024, 128, and 1024, respectively. The tanh function is used as the activation function between the convolution layer and the fully connected layer [15]. Input contains body acceleration, gravity acceleration, gyroscope, and quaternion with dimension equal to 26. The action signal is divided into segments of 25 timestamps by stride-1 sliding windows In this approach an action is performed and recorded separately and ground-truth labels was assigned. Then, the signals of each action with ground-truth labeling are used as input to CAE model for feature extraction by sliding windows [15]. Authors mentioned a representation of a signal data by the latent vector in the low-dimensional space after using the CAE model. The latent vector then will be clustered by K-Means Clustering so that centroids are referred to as motion bases in a model. For action tracking with LSTM, the displacement in each timestamp was predicted first and then added to its previous location, instead of predicting each position directly [15]. Similar as in work [6] the 3-layered LSTM model performed better compared to Bi-LSTM. Authors said that the possible reason could be the short-term correlation of human actions in their dataset and that Bi-LSTM with its complicated model structure is rather suitable for long-term actions [15].

The paper of *Chung et al., 2014* should be noted separately in the end of related works review because it provides an interesting compassion and evaluation of the performance of recurrent units LSTM and GRU on sequence modeling. Authors mentioned the ability of LSTM to keep the existing memory via the introduced gates and thus to detect an important feature from an input sequence at early stage, to easily carry this information (the existence of the feature) over a long distance, hence, capturing potential long-distance dependencies [7]. The GRU takes linear sum between the existing state and the newly computed state similar to the LSTM but does not have any mechanism to control the degree to which its state is exposed, but exposes the whole state each time [7]. *Chung et al., 2014* emphasize the fact that any important feature, decided by either the forget gate of the LSTM unit or the update gate of the GRU, will not be overwritten but be maintained as it is [7]. LSTM unit controls the amount of the new memory content and does not have any separate control of the amount of information flowing from the previous time step. The GRU differs and controls the information flow from the previous activation when computing the new and does not independently control the amount of the candidate activation being added via update gate [7]. The experiments provided in this work clearly indicate the advantages of the gating units over the more traditional recurrent units. *Chung et al., 2014* mentioned that with dataset they used GRU unit outperformed LSTM unit. But they suggest that the choice of the type of gated recurrent unit may depend heavily on the dataset and corresponding task [7]. Thus in section **??** of chapter “Data and Model” the data exploration of the obtained from

HMD Microsoft HoloLend is done in order to understand the dataset before the beginning with a development of model architecture.

3 Implementation

This chapter presents the steps of development and implementation of the proposed approach. The Unity Application for HoloLens was deployed on HMD and a raw data with measures and dimension columns was obtained. This data than was analyzed and preprocessed to ensure that the captured data can be used in corresponding machine learning models. The model architecture was implemented and experimentally improved during training and evaluation steps.

3.1 6-DoF Dataset

This section describes how the dataset was obtained, analyzed and presents the visualization of user's head position and rotation. The real 6-DoF dataset must be used as training data from which the model can learn the spacial and time dependences. This step is crucial for a high accuracy prediction and almost all machine learning approaches requires not only row data collection but also data exploration and data preprocessing steps to be done before training begins.

3.1.1 Data collection from HMD

In this master thesis HoloLens 2, the second iteration of Microsoft's head-mounted mixed reality device, was used for data collection. The user position and orientation were obtained with Unity application developed for this purpose. Main Camera in Unity is automatically configured to track head movements. More details about Unity application can be found in section **??**.
Using the Main Camera, a user position (x, y, z) and orientation in quaternion (qx, qy, qz, qw) were logged in a csv-file. Quaternions obtained from HMD will be used to define a rotation by four numbers. Quaternions representations are very convenient for operations such as composition or rotations and coordinate transformation [**principles_robot_motion_book**]. For these reasons quaternions are chosen for the representation of user head's rotation in three dimensions. Comparing to dataset in [11], the additional parameters were recorded from the Main Camera in order to add more information during training processes. Thus the world-space speed of the camera in meters per second was recorded. Unity velocity has the speed in (x, y, z) defining the direction. The obtained 6-DoF dataset has 10 features used in

training process: position (x, y, z), orientation (qx, qy, qz, qw) and velocity (x, y, z). The datasets were recorded in the laboratory space. HMD was presented to users and the basic functions were explained. During data recording, users freely walked wearing HMD in laboratory space. The Unity application, running on HMD, not only recorded the mentioned before parameters but also had a volumetric animated object placed 3 meters ahead of the user in the Mixed Reality environment. No personal data was recorded during these sessions and all traces are obtained anonymously. Thus, after an Unity application was launched, user could immediately see the animated object. The several traces were recorded at least for 10 minutes each. It allows to have enough data after splitting the dataset into training, test and validation partitions. Table 3.1 show the first 20 rows from raw dataset obtained from HoloLens 2 and used in training. As mentioned above, 6-DoF dataset has 10 columns, thus the table 3.1 presents only $timestamp$ and position (x, y, z) columns.

timestamp	x	y	z
2.649431	0.004954389	0.003402365	0.01010712
2.66943	0.00459053	0.003120769	0.01130438
2.698009	0.003960807	0.002990472	0.01276976
2.719285	0.003730714	0.003037783	0.01305151
2.746641	0.003252693	0.003489003	0.01368421
2.764094	0.003153284	0.003518121	0.01400959
2.780033	0.003087142	0.003409061	0.01435899
2.802086	0.003021815	0.00314023	0.01473305
2.815575	0.002789935	0.003551113	0.01506916
2.832602	0.002527435	0.003542757	0.01534094
2.848514	0.002212256	0.003605011	0.01565307
2.863769	0.001921757	0.003369405	0.01590317
2.879648	0.001668522	0.00348538	0.01607716
2.89686	0.001501704	0.003624826	0.01627397
2.913541	0.001487849	0.00359472	0.01643924
2.930006	0.001501501	0.003769569	0.01669565
2.948201	0.001617525	0.004252479	0.01697758
2.964302	0.001755987	0.004224311	0.01721937
2.97978	0.001838901	0.004487753	0.01747578
2.997117	0.002005509	0.005007531	0.01782864

Table 3.1: Raw data from HoloLens 2

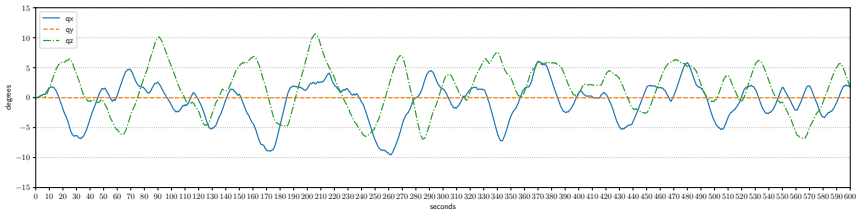
The first column in dataset is $timestamp$. It is obviously, that timestamp appears in row dataset not linearly and comes with different pauses. Even the high-cost HMD, like used in this research HoloLens 2, is sometimes unstable in frame rate during collecting data[1]. In the Unity Application, the frame rate is 60 Hz which means that data will be collected per 0.016(6) second. Unfortunately, HMD could have delays, and the time gap of two samples may be reduced or increased, as we can observe on raw dataset. Data on some expected timestamps may be unavailable in HMS for recording. Those sequences with fewer timesteps may be considered to have missing

[1] https://docs.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/hologram-stability

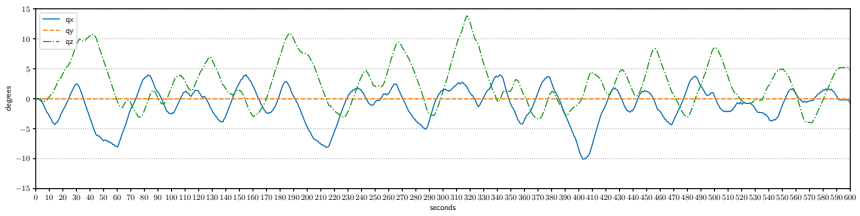
values. To deal with above situation, the preprocessing steps must be done. They are described in a section **??** below.

3.1.2 Data Exploration

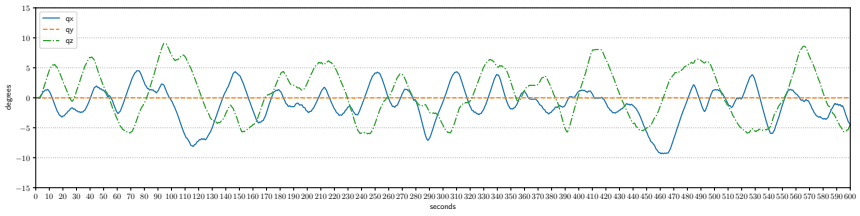
The next step after looking at raw data, gathered for machine learning, is a data exploration. The goal of this initial step in data analysis is a data visualization and an usage of statistical techniques to describe dataset characterizations. As already stated in section **??**, a user position (x, y, z), orientation in quaternion (qx, qy, qz, qw) and the world-space speed of the camera for each direction in (x, y, z) was obtained from Main Camera in Unity application launched on HMD.



(a) Dataset 1556



(b) Dataset 1623



(c) Dataset 1626

Figure 1: User position plots from obtained datasets (a, b, c)

First, let's start with the analysis of user position data. The figures 1a, 1b, 1c show dataset named $1556, 1623$ and 1626 correspondently. As a matter of fact, plotted dataset were already interpolated on the preprocessing step. Although interpolation was done before data exploration after text data analysis, the details about interpolation can be found in section **??**. The names of datasets means only a $timestamp$ in form $HH : MM$ when a dataset was obtained from HMD in laboratory

space. Thus the unique name of csv-files on HMD system was guaranteed for the day of experiment. In this thesis the names will be used to identify each of all three datasets.
All traces were recorded over 10 minutes long on average 12 minutes. All traces were then shortened to a precise length of 10 minutes to ensure equal data length for the purpose of visualization and analysis. The observations based on the sample traces can be made similar as it done by *Gül et al., 2020* in their work [11]. The user rarely moves along the y-axis. The y-axis shows the vertical movement that the users could make if they sit down or stand up. Based on the data obtained, users walked around a volumetric object in virtual reality and did not make particularly noticeable and prolonged attempts to examine the object at the lower point of the projection on a laboratory's floor since vertical movement requires more effort to crouch down and stand up. The laboratory space where the dataset was obtained was not cluttered with furniture thus users could walk around the volumetric object projected into their HMD. The figure 2 shows an enlarged y-axis in the range from 400ms to 500ms and thus proves there is no significant change in the vertical position of the user.

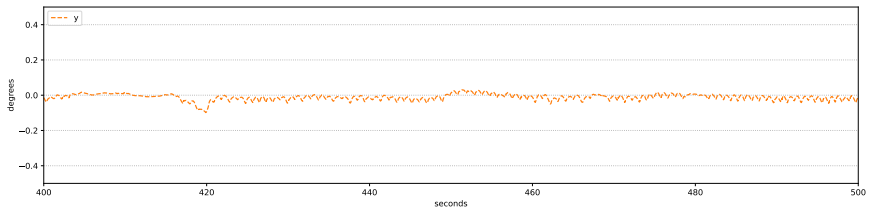


Figure 2: Changes in the user's position along the Y axis in the range from 400ms to 500ms

3.1.3 Data preprocessing

As was mentioned in section **??**, the raw sensor data obtained from the HoloLens was unevenly sampled at 60 Hz and had different temporal distances between consecutive samples. *Gül et al., 2020* obtained the similar raw dataset from same HMD and interpolated it to obtain temporally equidistant samples. Same as it was done in work [11], the position and velocity data were upsampled using linear interpolation and for quaternions SLEP was used. During preprocessing step Euler angles (yaw, pitch, roll) were calculated from quaternions and these parameters are used for visualization purposes. Although the interpolated csv-file contains additional Euler angles columns, only described in section **??** parameters were used for training and prediction. The table 3.2 lists first 20 rows for columns $timestamp$ and position (x, y, z) from interpolated dataset. This interpolated dataset is created with linear interpolation and SLEP mathematical methods from raw dataset shown in table 3.1.

timestamp	x	y	z
0.0	0.004954389	0.003402365	0.01010712
5000000.0	0.004833102666666667	0.0033084996666666667	0.010506206666666667
10000000.0	0.004711816333333333	0.0032146343333333332	0.010905293333333333
15000000.0	0.00459053	0.003120769	0.01130438
20000000.0	0.004485576166666666	0.003099052833333333	0.011548609999999999
25000000.0	0.004380622333333333	0.0030773366666666667	0.011792839999999999
30000000.0	0.0042756685	0.0030556205	0.01203707
35000000.0	0.004170714666666667	0.003033904333333333	0.0122813
40000000.0	0.004065760833333334	0.003012188166666667	0.01252553
45000000.0	0.003960807	0.002990472	0.01276976
50000000.0	0.00390328375	0.00300229975	0.0128401975
55000000.0	0.0038457605	0.0030141275	0.012910635
60000000.0	0.00378823725	0.00302595525	0.0129810725
65000000.0	0.003730714	0.003037783	0.01305151
70000000.0	0.0036510438333333334	0.0031129863333333335	0.01315696
75000000.0	0.003571373666666667	0.003188189666666667	0.01326241
80000000.0	0.0034917035000000003	0.003263393	0.01336786
85000000.0	0.0034120333333333332	0.0033385963333333333	0.01347331
90000000.0	0.0033323631666666667	0.0034137996666666667	0.01357876
95000000.0	0.003252693	0.003489003	0.01368421

Table 3.2: Interpolated data from HoloLens 2

After the interpolated dataset was plotted as figure 3, the important observations based on the sample trace could be done. While the user position data plots look appropriate for machine learning algorithms, the two graphs with orientation show data that is not the perfect case for usage with machine learning technologies and could decrease the prediction rate. The real part qw and the component qy of quaternion and yaw of Euler angles have obviously discontinuous (sharp change of sign) making it hard for a predictor to learn. A orientation on quaternions is used in training, thus this data requires a few additionally preprocessing steps. Usually, when doing calculation with quaternions, quaternions must be normalized to a unit length in order to represent valid rotations [**principles_robot_motion_book**]. The normalized quaternion can be calculated using formula:

$$U_g = \frac{q}{||q||} = \frac{w}{||q||} + i \cdot \frac{x}{||q||} + j \cdot \frac{y}{||q||} + k \cdot \frac{z}{||q||} \tag{3.1}$$

where $||q||$ is a magnitude and can be found with formula:

$$||q|| = \sqrt{w^2 + x^2 + y^2 + z^2} \tag{3.2}$$

During experiments with quaternions in dataset obtained from HoloLens was found that quaternion magnitudes $||q||$ in HoloLens dataset are equal to 1. Thus data came from HMD already normalized, so that a quaternion in dataset kept the same orientation as it was during user's movement but its magnitude is 1.0.

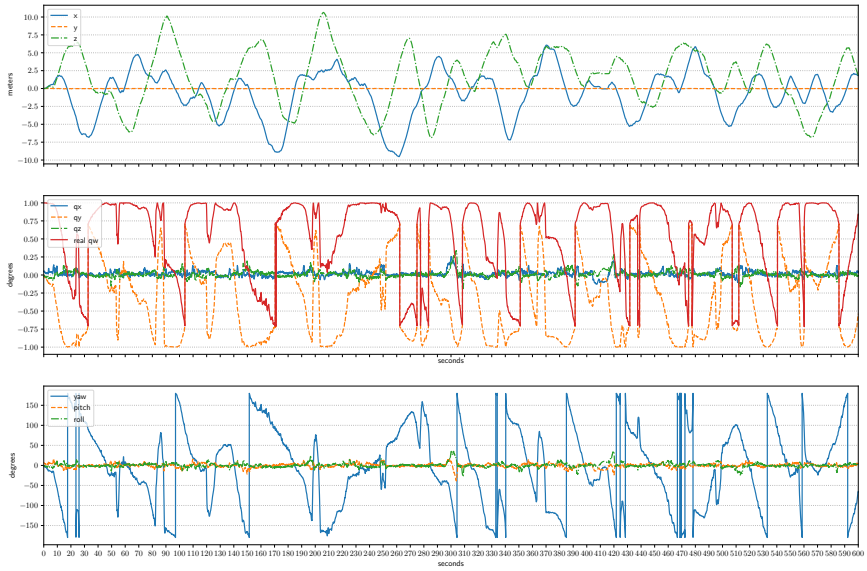


Figure 3: Interpolated 6-DoF dataset's user position and orientation in quaternions and Euler angles.

Next, quaternions between neighboring points in obtained dataset represent the very similar orientation made by user wearing HMD step by step. The middle plot on figure 4 has discontinuities that can be seen on qw line. As a consequence of the discontinuity (sharp change of line from negative to positive area with the same amplitude) the two neighboring quaternions with similar rotation have significant 4D vector space between them. It makes prediction worse what can be proved by RMSE and MSE rotation metrics. Flipping the sign will not affect the rotation, but it will ensure that there are no large jumps in 4D vector space when the rotation difference in rotation space (SO(3)) is small. If negative component of quaternions will be flipped into positive then the dataset representing same rotation without creating an artificial discontinuity in the space will be available for model training.

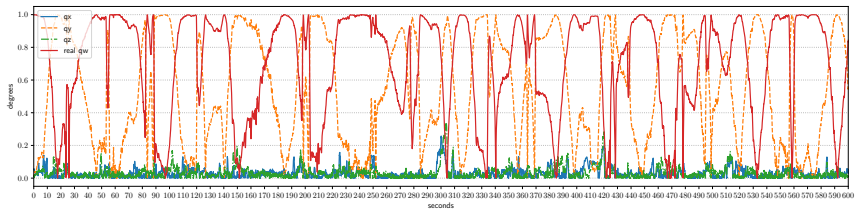


Figure 4: Quaternions from 6-DoF dataset's flipped if their real part is negative

Thus after normalization step, the two representations of quaternions are blended into one data set, omitting to discontinuities in the time series as can be seen

presented on Figure 4. Indeed, the RMSE and MSE rotation metrics were improved when model was trained with dataset with quaternions without sharp sign changes. More information can be found in sections **??** and **??**. The figure 5 represents quaternions of the original interpolated dataset on the upper part of the plot and the normalized flipped quaternions on the lower part of the plot. The quaternion's components were flipped only if the if their real part became negative. Different to figure 4 the limit of y-axis is set to [-1, 1] on figure 5 so that the result of inverting of quaternion is easy co compare to original data. Figure 4 shows plotted data with length of 20 seconds in range 162 - 182 s from both datasets.

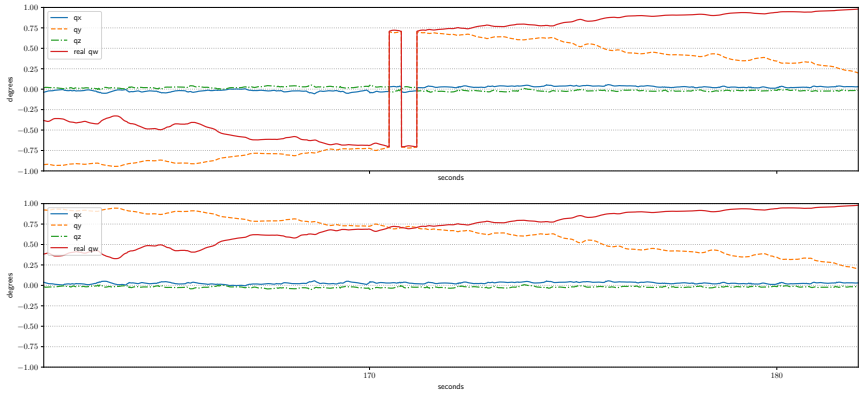


Figure 5: Quaternions from 6-DoF dataset's flipped if their real part is negative

3.2 Model

3.2.1 Model inputs

3.2.2 Architecture

LSTM Model

GRU Model

Bidirectional GRU Model

3.2.3 Development

This section presents the developments of the Unity application for obtaining the dataset and development of LSTM and GRU models with Python and PyTorch.

Unity application

An application was developed in Unity with the Mixed Reality Toolkit and deployed on HoloLens 2. The goal of the application is to obtain the user position and orientation during the time a user wears a HMD. As this research aims to find an approach to reduce the M2P latency during rendering and delivering the volumetric content to end-user device, the volumetric animated object was placed three meters ahead of the user in Unity application. Users wearing HMD thus were asked to look on animated volumetric object and to move freely inside the laboratory space.
In Unity, the Main Camera is always the primary stereo rendering component attached to HMD and it is rendering everything the user sees [2]. The starting position of the user is set to $(0, 0, 0)$ during the application launch and the Main Camera tracks movement of the user's head. Although HoloLens allows to build a world-scale application, the room-scale experience was selected for spatial coordinate system. This lets users to walk around within the 10-meter boundary what is quite enough for user's movements inside the laboratory space and simultaneously watching the volumetric video object.

User position and rotation data were logged in csv-file. This raw data has been converted into datasets on the preprocessing step and thus original interpolated dataset, the transformed with flipped negative quaternions and several normalised datasets were used in experiments during model development and hyperparameters search.

Training and evaluation

The LSTM and GRU models development and implementation are done using Python and PyTorch.

[2] https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/camera-in-unity

Hyperparameter search

The hyperparameters search is done using VCA GPU cluster which is installed with the SLURM resource manager/scheduler and Singularity container is used to containerize the application.

4 Evaluation

4.1 Baseline model

4.2 Goal of evaluation

4.3 Evaluation metrics

4.4 Experiments

4.4.1 First experiments

Datasets

As already stated in section **??**

Batch size

A high impact on the performance e.g. the prediction accuracy has a batch size used in LSTM or GRU Model. The batch-size helps to learn the common patterns as important features by providing a fixed number of samples at one time. So that the model thus can distinguish the common features by looking at all the introduced samples of the batch. In most cases, an optimal batch size is set to 64. When this batch size was initially used with LSTM model, it gave significant high MSE, RMSE, train and validation errors. Based on the performance observation during experiments with LSTM parameters, batch size fine-tuning was done. The experiments done by *Aykut et al* in their works [2] and [3] proved that appropriate batch size can be found in range 2^9 - 2^{11} (512 - 2048). Notice that a power of 2 is used as a batch size. The overall idea is to fit a batch of samples entirely in the the CPU/GPU. Since, all the CPU/GPU comes with a storage capacity in power of two,

it is advised to keep a batch size a power of two. Using a number different from a power of 2 could lead to poor performance.

Learning rate

4.4.2 Prediction with LSTM

4.4.3 Prediction with GRU

4.4.4 Prediction with Bidirectional GRU

5 Conclusion

The Python application $UserPrediction6DOF$ is a result of this work and can be used for future preprocessing of the new obtained datasets, training routine and prediction of user position and rotation in 6-DoF virtual environment.

5.1 Analysis

5.2 Limitations

5.3 Suggestions for future work

Bibliography

[1] A. Deniz Aladagli, Erhan Ekmekcioglu, Dmitri Jarnikov, and Ahmet Kondoz. *Predicting head trajectories in 360° virtual reality videos*. https://ieeexplore.ieee.org/document/8251913. date access on 29.03.22. 2017. DOI: 10.1109/IC3D.2017.8251913.

[2] Tamay Aykut, Christoph Burgmair, Mojtaba Leox Karimi, and Eckehard Steinbach. *Delay Compensation for a Telepresence System With 3D 360 Degree Vision Based on Deep Head Motion Prediction and Dynamic FoV Adaptation*. https://arxiv.org/abs/2007.14084. date access on 23.02.22. 2018. DOI: 10.1109/WACV.2018.00222.

[3] Tamay Aykut, Eckehard Steinbach, and Jingyi Xu. *Realtime 3D 360-Degree Telepresence With Deep-Learning-Based Head-Motion Prediction*. https://www.researchgate.net/publication/330861228. date access on 25.03.22. 2019. DOI: 110.1109/JETCAS.2019.2897220.

[4] Y. Bengio, P. Simard, and P. Frasconi. *Learning long-term dependencies with gradient descent is difficult*. http://www.cs.unc.edu/techreports/93-010/93-010.pdf. date access on 31.03.22. 1994. DOI: 10.1109/72.279181.

[5] Devesh K Bhatnagar. *Position trackers for Head Mounted Display systems: A survey*. http://www.cs.unc.edu/techreports/93-010/93-010.pdf. date access on 31.03.22. 1993.

[6] Yun-Kai Chang, Mai-Keh Chen, Yun-Lun Li, Hao-Ting Li, and Chen-Kuo Chiang. *6DoF Tracking in Virtual Reality by Deep RNN Model*. https://ieeexplore.ieee.org/document/9394069. date access on 02.04.22. 2020. DOI: 10.1109/IS3C50286.2020.00057.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. https://arxiv.org/abs/1412.3555. date access on 30.03.22. 2014. DOI: 10.48550/arXiv.1412.3555.

[8] Xavier Corbillon, Gwendal Simon, Alisa Devlic, and Jacob Chakareski. *Viewport-adaptive navigable 360-degree video delivery*. https://ieeexplore.ieee.org/document/7996611. date access on 05.04.22. 2017. DOI: 10.1109/ICC.2017.7996611.

[9] Alessandro Crivellari and Euro Beinat. *LSTM-Based Deep Learning Model for Predicting Individual Mobility Traces of Short-Term Foreign Tourists*. https://www.researchgate.net/publication/338377314. date access on 08.04.22. 2020. DOI: 10.3390/su12010349.

[10] Fanyi Duanmu, Eymen Kurdoğlu, S. Hosseini, Yong Liu, and Yao Wang. *Prioritized Buffer Control in Two-tier 360 Video Streaming*. https://www.researchgate.net/publication/319048432. date access on 05.04.22. Aug. 2017. DOI: 10.1145/3097895.3097898.

[11] Serhan Guel, Sebastian Bosse, Dimitri Podborski, Thomas Schierl, and Cornelius Hellge. *Kalman Filter-based Head Motion Prediction for Cloud-based Mixed Reality*. https://arxiv.org/abs/2007.14084. date access on 19.02.22. 2020. DOI: 0.1145/3394171.3413699.

[12] Serhan Gül, Dimitri Podborski, Thomas Buchholz, Thomas Schierl, and Cornelius Hellge. *Low-latency Cloud-based Volumetric Video Streaming Using Head Motion Prediction*. https://arxiv.org/abs/2001.06466. date access on 19.02.22. 2020. DOI: 0.1145/3394171.3413699.

[13] Sepp Hochreiter and Jürgen Schmidhuber. *Long Short-term Memory*. https://www.researchgate.net/publication/13853244. date access on 31.03.22. Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[14] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. *LSTM Fully Convolutional Networks for Time Series Classification*. https://arxiv.org/abs/1709.05206. date access on 14.04.22. 2017. DOI: 10.48550/arXiv.1709.05206.

[15] Hao-Ting Li, Yung-Pin Liu, Yun-Kai Chang, and Chen-Kuo Chiang. *Action recognition and tracking via deep representation extraction and motion bases learning*. https://www.researchgate.net/publication/358012181. date access on 01.04.22. 2022. DOI: 10.1007/s11042-021-11888-8.

[16] Anh Nguyen, Zhisheng Yan, and Klara Nahrstedt. *Your Attention is Unique: Detecting 360-Degree Video Saliency in Head-Mounted Display for Head Movement Prediction*. https://www.researchgate.net/publication/328370817. date access on 15.03.22. 2018. DOI: 10.1145/3240508.3240669.

[17] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. *Optimizing 360 video delivery over cellular networks*. https://dl.acm.org/doi/10.1145/2980055.2980056. date access on 12.03.22. 2016.

[18] Silvia Rossi, Irene Viola, Laura Toni, and Pablo Cesar. *A New Challenge: Behavioural Analysis Of 6-DOF User When Consuming Immersive Media*. https://ieeexplore.ieee.org/document/9506525. date access on 03.04.22. 2021. DOI: 10.1109/ICIP42928.2021.9506525.

[19] Silvia Rossi, Irene Viola, Laura Toni, and Pablo Cesar. *From 3-DoF to 6-DoF: New Metrics to Analyse Users Behaviour in Immersive Applications*. https://www.researchgate.net/publication/357172010. 1-7. 2021. date access on 13.04.22. 2021.

[20] Afshin Taghavi, Anahita Mahzari, Joseph Beshay, and Ravi Prakash. *Adaptive 360-Degree Video Streaming using Scalable Video Coding*. https://www.researchgate.net/publication/320542716. date access on 05.04.22. Oct. 2017. DOI: 10.1145/3123266.3123414.

[21] Zhiguang Wang, Weizhong Yan, and Tim Oates. *Time Series Classification from Scratch with DeepNeural Networks: A Strong Baseline*. https://www.researchgate.net/publication/318332658. date access on 25.03.22. 2017. DOI: 10.1109/IJCNN.2017.7966039.

[22] Lan Xie, Zhimin Xu, Yixuan Ban, Xinggong Zhang, and Zongming Guo. *360ProbDASH: Improving QoE of 360 Video Streaming Using Tile-based HTTP Adaptive Streaming*. https://www.researchgate.net/publication/320542716. date access on 05.04.22. Oct. 2017. DOI: 10.1145/3123266.3123291.

[23] Dingqi Yang, Benjamin Fankhauser, Paolo Rosso, and Philippe Cudre-Mauroux. *Location Prediction over Sparse User Mobility Traces Using RNNs: Flashback in Hidden States*. https://www.researchgate.net/publication/338377314. date access on 07.04.22. 2020. DOI: 10.24963/ijcai.2020/302.

[24] Emin Zerman, Radhika Kulkarni, and Aljosa Smolic. *User Behaviour Analysis of Volumetric Video in Augmented Reality*. https://ieeexplore.ieee.org/document/9465456. date access on 13.04.22. 2021. DOI: 10.1109/QoMEX51781.2021.9465456.