Predicting Avatar Movement in Virtual Reality using Neural Networks

Jakob Fehle
Universität Regensburg
Regensburg, Germany
jakob.fehle@stud.uni-r.de

David Halbhuber
Universität Regensburg
Regensburg, Germany
david.halbhuber@stud.uni-r.de

Jonathan Sasse
Universität Regensburg
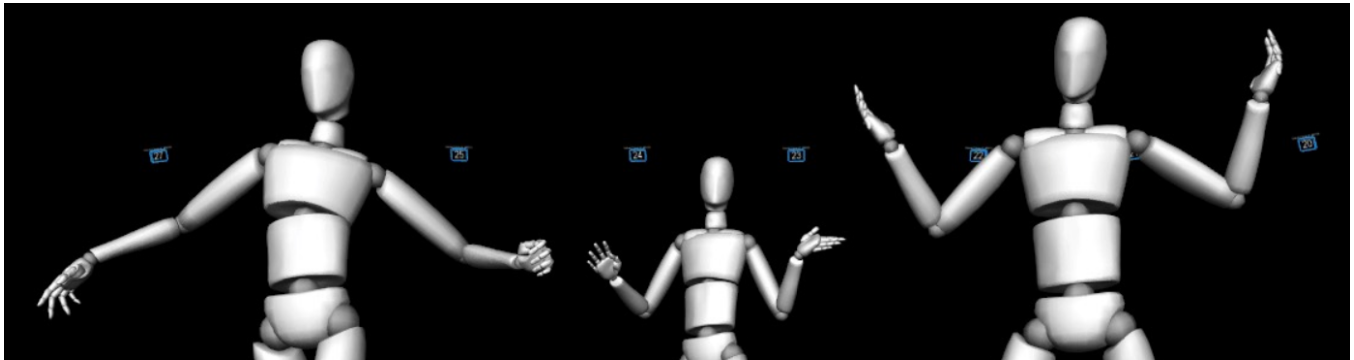Regensburg, Germany
jonathan.sasse@stud.uni-r.de

Figure 1: Motion captured avatars showing different postures.

ABSTRACT

In this work we present a system to predict human avatar movement in a virtual reality environment. This prediction is meant to compensate the system latency caused by the motion capturing system and the head mounted device used in the environment. Previous research has shown that latency immensely affects the immersion and presence feeling of users. Likewise, latency might also significantly hinder user performance in completing certain tasks. To minimize those negative effects, we firstly developed a system that is able to reliably determine the system latency of the used setup. Secondly, we present two different neural networks for predicting human body movement 50ms, respectively 104ms, in the future. Thirdly, we evaluated the proposed system with a user study. We showed that the used system neither improved nor worsened the presence or the feeling of immersion. Furthermore, no increase in performance was observed. We were not able to reveal a significant effect using either of the presented models. Additional research is needed to further investigate the link between latency in virtual reality systems and the experience a user undergoes.

CCS CONCEPTS

• **Human-centered computing** → **Interaction design**.

KEYWORDS

motion capturing, artificial neural networks, movement prediction, virtual reality

1 INTRODUCTION

State-of-the-art motion capturing systems can be utilized to transfer movement to a computer generated space. This space can be used to integrate movement in different kinds of computer applications, such as games, animation software or any kind of application that aims to mimic human movement as accurate as possible. In a human-centered approach the movement of a body gets transferred to the application, which allows the user to directly interact with the software through natural movement.

A further step towards the direct integration of natural human motion and behavior into the above applications can be achieved by combining state-of-the-art motion detection devices with head-mounted displays (HMD). These devices allow the user to see the generated computer space, the so called Virtual Reality (VR). By combining a motion capturing system and a HMD the user is able to see its avatars movement in VR through its own eyes. The user is able to see and move a virtual representation of its own body. Such systems have already been widely researched. This is shown exemplary in the work of Chua et al. [4] in which the authors propose and evaluate a method to learn Taijiquan, an ancient Chinese form of material arts, using a motion capture system in combination with a HMD. Although HMD's coupled with motion capturing software are a promising approach for a wide variety of applications, there are some disadvantages and system weaknesses.

Since many different technical components and systems are connected during operation, a certain technical latency arises in the course of communication between devices. This means that the user operating the system is not able to see its movements in real time. If, for example, the user lifts its arm, the system takes some time to register, process and transmit the movement to the HMD, which needs time to display the received data correctly. The processing and transmission time results in a perceivable latency and can ultimately lead to a reduced feeling of presence [23, 27].

The loss of presence can be detrimental to the experience the user encounters while using the system, as this determines whether the user feels integrated or out of place in the computer-generated world [9, 30]. Furthermore a high latency value is able to influence a users overall performance in completing and perceiving tasks as well [20]. Figure 2 shows a generalized illustration of a motion capturing system in combination with a HMD.

This work aims to solve the problems, associated with a high and perceivable system latency through a movement prediction mechanism based on artificial deep neural networks (DNN). The presented system is capable of predicting user movement by calculating probabilities of movement based on a prior learned data set. DNNs, and artificial neural networks in general, have already achieved remarkable results in predicting different system states, such as the anticipated traffic volume of a crossroad [14] or the amount of expected rainfall in a certain region [21]. Likewise, artificial neural networks are already being used to detect human movements, such as pedestrian movements in road traffic [19, 36, 42].

The main contribution of this paper is a deeper understanding on how movement prediction systems for VR environments can be developed and deployed. Firstly, we showed how the system latency of any motion capturing system coupled to a HMD can be determined reliably. Subsequently, we show how a generalized data

set describing natural human body movement can be obtained in a data collection study. We provide design insights on how the study should be designed, to generated data suitable for training a neural network. Additionally, we demonstrate that a simple dense neural network, with about 9 million trainable parameter, is sufficient for predict body movement in VR. We present two different neural networks for predicting human body movement 50ms, respectively 104ms, in the future. Lastly, we conducted a user study to evaluate the developed system. The study did not reveal any significant differences in users presence and performance. We discuss the outcomes and findings of the conducted study and conclude that further research is required.

2 RELATED WORK

This section provides a brief overview of existing research in the various adjacent research fields. First of all, we discuss existing possibilities for latency compensation as well as their applicability in our scenario. Subsequently, we present papers that deal with the prediction of human movements through neural networks. The following subsection deals with the measurement and interpretation of immersion in virtual reality environments. Additionally we dive into the field of predicting movement in a VR environment. Subsequently, related work will be presented that deals with the use of neural networks as a classification tool. This section concludes with a summary and on how the gained knowledge will find use in the presented paper.

Latency compensation

Wu and Ming demonstrated that latency greatly influences human performance in spatial tasks [46]. The authors developed a system which is able to track and predict the head movement of a human using a HMD. The presented system uses simple linear extrapolation, called *Kalman Filtering* [15], to assume and calculate future head positions. Kalman filtering is based on the assumption that the values to be predicted are describable by a mathematical model, such as a motion equation. A similar assumption is made when using a neural network to predict any value, with the exception that neural networks usually utilize a so called activation function. The used activation function transforms a former linear regression to a non linear problem. Using the Kalman filtering method the authors were able to evidently show that a human can improve task performance by up to 120 % when using latency compensation methods. Since neural networks are capable to coup with more complex problems, using them to predict the movement of the whole body and thus effectively reducing the system latency is a promising approach.

Movement prediction through neural networks

Human movement prediction is an active field of research in various areas. One extraordinary achievement in this research field lays in the work of Martinez, Black, and Romero [22]. In their paper, the authors show that traditional Recurrent Neural Networks are able to outperform the performance of complex state-of-the-art networks, like LSTM-3LR [7]. This superior performance in predicting human motion is achieved through meticulous hyper-parameter tuning, such as increasing the batch size instead of reducing the learning

rate and the use of a sequence-to-sequence approach similar to that used in common translation algorithms. The authors justify this approach with the difficulty to correctly implement noise vectors in a time-dependent data set of human movements. The sequence-to-sequence variant is implemented by two neural networks - an encoder network and a decoder network. The encoder network is trained to generate plausible noise vectors based on the available motion data. The decoder network is trained with the extended data set (motion data and noise vectors) for the actual motion prediction. Furthermore, the authors point out that the so-called zero velocity joints have received too little attention in the past. By using a baseline based on zero-velocity joints the authors were able to achieve an even better result in the short-term prediction of human movements. It is to be shown if RNNs in general and the above presented method of baseline building is applicable in this work.

Movement prediction in virtual reality

The prediction of movement in VR for various different reasons has already been widely researched. Since HMDs are worn on the head, most of research is focused on predicting head movements. One example is the work of Saad, Caudell and Wunsch in which they analyse the predictability of movement data in an VR environment [28]. They show that time series data gathered from human head movement is part deterministic and part chaotic. This is particularly interesting because if the data was just chaotic, it could not be predicted. However, since the authors showed that the data is also partly deterministic, it may be predicted to a certain degree. The proposed method of testing the predictability of data could be utilized in this work as well.

A different approach in predicting head movement is shown in the work of LaValle et al. The authors used two different methods to predict the future head position of a user wearing a HMD. The presented methods are not based on neural networks but on rather simple mathematical assumptions. In their work they show that by using predictive head tracking the latency can be reduced dramatically. They further argue that this reduction in latency can significantly enhance user experience. For the actual prediction they used different sensors built in the HMD. Based on the sensor data, primarily acceleration data, they built two different prediction models. The first model assumed the angular acceleration of the head is constant in a specific time frame. Based on this assumption the future head position could be easily calculated. The second model assumed the acceleration of the head not to be constant. To calculate a future head position in the second model the angular acceleration was reread from the sensor every time a calculation was needed. This caused additional processing time. Interestingly they found, that even with the simpler model, assuming a constant angular acceleration, the latency could be drastically reduced. A similar approach could be utilized in this work, by assuming the whole human body acceleration in a given time frame is constant. This would reduce problem complexity enormously.

Classification with neural networks

The use of neural networks to solve classification problems is widespread and now accepted best practise in many circumstances.

Neural networks have proven to be better or on par with traditional classification methods in different scenarios. For instance, Esteva et al. demonstrated how a network trained by them is able to diagnose skin cancer with the same accuracy as a professional dermatologist [6]. Spanhol et al. follow a similar approach in their work. In their paper, they show that initial experiments in classifying breast cancer using neuronal networks are already yielding better results than conventional methods of machine learning [34]. In the non-medical field, neural networks were also able to achieve remarkable successes in classification problems. In their work, Rahmouni et al. present a neural network that is able to distinguish real photographs from computer-generated images. The presented network surpasses other known methods and algorithms for image recognition [26].

Although neural networks have been shown to achieve exceptional classification results, they cannot be used for every type of problem. Usually, time-dependent problems, such as predicting a stock price or predicting a movement, are not considered classification problems. Traditionally, the type of problem would be solved with Recurrent Neural Networks (RNN). RNNs are able to include the time factor in the prediction and thus achieve better results than classification methods [16]. The problem presented in this paper can be interpreted as a time-dependent problem and also as a classification problem. It will be shown, which type of problem solution is more suitable in the case of this work.

Presence in virtual reality

The term presence refers to the feeling that the user is in the virtual world created by computer screens, in case of this work the world created and displayed on the HMD [30]. Presence in VR can be a decisive factor in various situations. In a study with almost 1000 participants, Tussyadiah et al. show that presence highly increases the general enjoyment of VR applications [43], furthermore the authors conclude, that a higher level of presence, over all positively betters VR experiences.

Also VR applications with much more specific use cases don't only benefit from a high presence feeling, but presuppose it compellingly. Consequently, Price and Anderson showed how the presence feeling affects the success of VR confrontation therapies [25]. Through the conducted study, they showed that there is a strong correlation between presence and therapy success. Also during the therapy a high presence feeling is decisive, because without the feeling to be actually in the VR, no fear triggering element could be released in many test persons. According to the authors, a confrontation therapy that cannot confront does not promise any real success.

Interestingly, related work also shows how a high levels of presence can help decouple the user from one's own body. Hoffmann et al. show how a sort of analgesia could be induced in test persons by using VR technology [18]. In their work, the authors showed how pain perception of volunteers can be significantly changed by the use of VR technology. The subjects were tested in three different settings whereat in each setting the subject was exposed to a low pain stimulus. Finally, the study concludes that a high presence in VR leads to reduced pain perception in the real world.

Summary

The presented work suggests a high connection between several phenomena, which are related to this work.Firstly, it is depicted that latency and performance correlates negatively. Secondly, it is shown that artificial neural networks are already capable of predicting human movement behaviour. Additionally, it shows that that presence tremendously influences the outcome and the execution of highly specialized VR applications. Lastly, it is shown how presence is able to influence the users perception bidirectionally. Furthermore, we showed that neural networks are best suited to solve certain classification problems. By combining the above findings, we aim to develop a system which not only increases the presence of VR users by depicting a more accurate presentation of themselves, but also increases the performance in completing certain tasks by compensating the occurring system latency.

3   LATENCY TEST FRAMEWORK

The latency reducing system presented in this paper consists of two technical subsystems. Our Latency Test Framework (LTF), the first technical subsystem, is needed to determine the actual latency of the used motion capturing system in combination with the utilized HMD. The movement prediction system, the second subsystem, is used to actually predict movement in the presented virtual environment. To minimize the latency between the users actual movement and the view shown in VR using the movement prediction system developed, it is essential to determine the exact latency first. In order to accomplish an highly accurate measurement, independent of confounding variables and users, our LTF, based on various hardware sensors coupled with a Python interface, was developed.

The hardware part of the LTF consists of an Arduino micro controller [1] which is linked to a shock sensor and a light sensitive photo sensor. NaturelPoints OptiTrack [24] motion capturing system is used to capture the motion of an object. The captured motions are sent to a computer running the 3D game engine Unity [41] which passes the motion on to the connected HTCs HMD Vive [12]. On the software side, multiple applications and scripts are used to react to the deflections of the hardware sensors.

Figure 2 depicts the workflow of the LTF. The process gets initiated by an user dropping an object, in the presented case a small blue ball. While the object and its fall are tracked by the motion capturing system, the vibration sensor ,attached to a panel, is triggered when hit. The Arduino connected to the vibration sensor is informed about the event and writes a timestamp in microseconds, which is then passed to a external computer. A Python script running on this computer receives and computes the timestamp of the vibration as starting value for the latency calculation.

In the meantime the OptiTrack system receives and processes the movement of the object and transfers the information onto the central computer. The movement data is being processed and rendered on the HMD using the Unity 3D game engine. Additionally the Unity application registers the objects collision with the wooden pane. Upon collision Unity starts a function to light up the rendered view which is dark by default.

The brightening is registered by the photo sensor detecting the change in light intensity. As soon as the photo sensor is triggered a
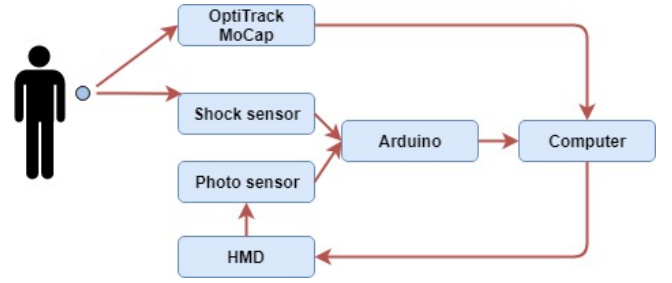

Figure 2: The workflow of the developed Latency Test Framework (LTF). The movement of an object gets tracked by the MoCap system. The object also triggers a shock sensor attached to an Arduino micro controller and a virtual trigger zone. Triggering this zone commands the computer to brighten the connected HMD. The change of brightness is received by the attached photo sensor. The overall latency is calculated by subtracting the time the photo sensor got triggered by the time the initial shock was registered.

signal gets passed to the Arduino, which then passes the second timestamp to the central computer.

By comparing the two timestamps it is possible to accurately calculate the duration needed for the system to register an object and display it on the HMD. Since the timestamps are recorded directly on the Arduino and only sent to the connected computer, there is no additional latency generated by the communication between computer and Arduino. Using the above described system we recorded 120 sequential measurements of the system latency. We determined a mean latency of 51ms with a standard derivation of 7 ms.

Our presented system is highly accurate, but there a some minor shortcomings in the used methodology. Firstly, it was not possible to reliably determine the response time (also known as activation time) of the used sensor modules manually. According to the sensors manufacturer, both have a response time of less than 2 micro seconds(Âţs). The second weak point represents the connection between the sensors and the Arduino micro controller. The time stamps required for the LTF are not set until the signal from the sensors reaches the Arduino. The time required for the transmission of the signal is not taken into account. Since the cables used are only comparatively short, loss-free transmission can be assumed [44]. The transmission speed in a lossless scenario is given by the general solution to Telegrapher's line transmission equations [29]:

$$u = \frac{1}{\sqrt{LC}} \tag{1}$$

The result, u, of the above formula is known as the propagation speed (often also phase speed) of electromagnetic waves. The value u depends on L, the inductance of the conductor, as well as on C, the capacitance of the conductor. Both values are dependent on the length of the conductor, as the conductors used in the LTF are extremely short, the term LC closes to near zero values. This results in the signal transmission from sensors to the Arduino close to the maximum transmission speed of electromagnetic waves, the speed of light. The time required for the transmission is so short, in fact

less than 1 microsecond, that it can be neglected when using the LTF. The last weak point of the system is the used Arduino itself. It cannot be reliably determined how long the Arduino needs to process the received data. In order to keep the processing time as short as possible, the code used had to be maximised in terms of efficiency. The theoretical minimum processing time of the Arduino is 4 microseconds,which is due to the resolution of the microsecond function (used to track latency and processing time) of the Arduino which only works with a resolution of 4 microseconds. This means that a processing time of less than 4 microseconds can only be represented by the Arduino as 4 microseconds.

In summary, the expected error due to all weak points amounts to a maximum of 7 microseconds. An error of this magnitude can be neglected in this paper without negative consequences.

4 DATA COLLECTION

A preliminary data acquisition study was conducted to collect the data necessary for training the presented artificial neural network.

Participants

For this purpose we recruited 20 volunteers (11 m, 9 f) with an mean age of 24,4 years (SD 4,0), which were recorded while participating. A few requirements and criteria for the participants were required in order to capture usable data. Subjects where the requirements were not met were excluded from the study and their data deleted, regardless of whether specifications failed prior or during the study.Participants should not bet compromised in their movement due to illness, physical disability or high age. Minors did not participate in the study because of the requirement for signing a consent form for using their data which would cause an administrative burden. Aside from these requirements, there were no prior experience needed. At the start of the data acquisition study participants had to sign a consent form to accept the collection of personal data as part of the used questionnaire and motion capturing. Afterwards they were introduced to the temporal sequence, their involvement and a basic guidelines on the study. Each task began and ended with the participant posing in a T-position.

Tasks and Conditions

For the study the participants had to execute 17 varying short tasks, like waving with switching hands or miming a sport they love. The tasks were designed to detail the entire range of motion of the human body, with as many variations as possible. In order to increase the individualization and the later generalization capability of the neuronal network, only the scope of some tasks was given by the experimenter. How the tasks were subsequently executed by the test person could be freely decided by the test person themselves. There were 4 different task sequences in a randomized order. Each individual was encouraged to execute each task for about 30 seconds while changing directions, speed and limbs, if achievable. For capturing participants were wearing the HMD and a motion capturing suit coupled to the OptiTrack system. To record the data, we use a self-developed client. The client catches the bit-stream of the Optitrack system, unpacks the packets and saves their contents in a CSV file. Each packet contains one frame of the 240Hz stream

of the motion capturing system. A frame in turn describes the corresponding skeleton created using OptiTrack. The skeleton created by OptiTrack, which consists of 51 different bones. For each bone, OptiTrack provides information on the current position and rotation. For the correct representation of the skeleton in third party applications, such as Unity, only the position of the hip as well as the rotational movement of all bones is necessary. The remaining position information was therefore not recorded by our client. In sum, the float values to be recorded for each frame amounted to a total of 207 (X-position hip, Y-position hip and Z-position hip, as well as X-rotation, Y-rotation, Z-rotation and W-rotation for each bone). To minimize the presented problem complexity, we decided to only predict 20 base bones including the hip bone, excluding finger bones completely. This results in an actual sum of 87 float values for each frame.

Using the method described above, we have recorded approximately 4.2 million individual OptiTrack frames. Thus the collected data for training the neural network summarizes to a size of about 20 GB. Figure 3 shows a fragment of the recorded data.

Frame	Timestamp	TaskId	ProbandId	PS_SS_2019_Hip_POSITION_X
398780	1563538222.2692175	01	01	-0.09854848682880402
398781	1563538222.2736356	01	01	-0.09846188127994537
398782	1563538222.277632	01	01	-0.09838151931762695
398783	1563538222.281597	01	01	-0.09830460697412491

Figure 3: Extract from the collected data. Shows frame, timestamp, task ID, respondent ID, and X coordinates of hip position.

Data preprocessing

Before the data set could be used for training the neural network, the data had to be preprocessed. The first step was to ensure that data, which was lost during data recording, was reconstructed, as the OptiTrack stream is only available through UDP and hence does not implement any kind of flow control. Linear interpolation made it possible to reproduce a completely missing frame and thus solve the above problem, even if this was only necessary once in the entire data set.

Secondly, we noticed that in some cases no values for certain position or rotation data were transmitted by OptiTrack. The problem of missing values was solved by linear interpolation as well. Interpolation between the corresponding values of the predecessor and successor frames was performed to estimate the missing value.

Finally, linear interpolation was reused to chain the data subsets of the different subjects, resulting in a large data set containing all frames of all subjects. Additionally we removed all data rows not needed by the neural net. The resulting data set was slightly slimmer and contained only rotation and position information of the bones. The assignment of the raw data to the corresponding bones or joints is done exclusively in our *Intercepter* Client, which is presented in section 6. Figure 4 shows a fragment of the cleansed, finalized data.

PS_SS_2019_Hip_POSITION_X	PS_SS_2019_Hip_POSITION_Y	PS_SS_2019_Hip_POSITION_Z
-0.09854848682880402	0.8931387066841125	-0.021672192960977554
-0.09846188127994537	0.8931452035903931	-0.02176666259765625
-0.098338151931762695	0.8931528925895691	-0.0218804944306612
-0.09830460697412491	0.8931491374969482	-0.02198389172554016

Figure 4: Cleansed extract from the preprocessed data. Shows X,Y,Z hip position of one participant. Data not needed for training the neural network is removed.

5 APPARATUS

With the system latency value at hand, the second technical subsystem for predicting and rendering the actual body movements was built. Utilizing the presented data set in the previous section a neuronal network was designed and developed. For the development we used Google Brain's Tensorflow [39], which allows not only the simple and fast creation of neural networks but also the use of the high-level application programming interface (API) Keras [2] to create highly complex networks [33].

Neural Network

The developed network may be classified as a classical Fully Connected Neural Net (FQNN), also known as Deep Neural Net (DNN) [38]. DNNs are usually used for classification problems [10, 37]. Although the problem presented in this paper cannot be considered as a classification problem, we decided to use a customized DNN. A crucial factor in choosing the network architecture was the time required for interference. Since the presented problem is exceedingly time-critical, it was necessary to minimize the interference time - this matter will be further addressed in the next subsection. To achieve a real time computation, we designed the network to be as simple as possible. In contrast to the classical and conventional implementation of DNNs, the network in this paper does not use the SoftMax function, since a categorical representation of the probability distribution of the output yields no benefit. The network consists of 87 input neurons, which pass data to the first of a total of two hidden layers. The first hidden layer contains 4096 units and is fully connected to the second hidden layer of 8096 units. A dropout function has been added to the second hidden layer [11, 35] to battle *overfitting*. The dropout rate was deliberately chosen to be relatively low, at 20 percent, to avoid slipping in the under-learning range. Subsequently follows the output layer with built-in ReLu activation function [8]. Figure 5 depicts the essential structure of the developed neural network.

The aim of the network is to use the available 87 data points from the OptiTrack stream to calculate 87 data points that are in the future displaced by a certain time. These 87 data points correspond to the position data of the hip and the rotation data of the other 21 joints in the frame. This procedure can be illustrated by using figure 4. The first line of the figure reflects the input (X) into the neural network. The last line shows the expected output (Y), in case the network should predict 4 frames into the future. Since the OptiTrack stream is available in 240Hz and the presented data was also recorded at this frequency, a prediction of 4 frames would correspond to a prediction of 12 milliseconds. For stochastic

optimization we used a variant of the ADAM [17] Optimizer, which is included by Tensorflow. The training process was initiated with a learning rate of 0.001 and a batch size of 512 samples. By using Keras callback functions [3], the learning rate could be dynamically adjusted during the training process. Depending on the validation accuracy, the learning rate was either increased or decreased up to a fixed threshold value. This process is comparable to classical learning rate decay in stochastic gradient descent method implemented in Tensorflow [40], for our problem our implementation proved to be much more effective in dealing with optimization plateaus. The Mean Squared Error (MSE), as shown in formula 2, has successfully proven itself for the loss functions in our simulation environment.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2. \qquad (2)$$

First experiments demonstrated that the developed network was already able to solve the given problem extraordinarily well. However, the accuracy did not diverge clearly against a value, which was caused by the implemented dropout function in the last hidden layer. To fix this problem, we implemented another Keras callback function which allowed the model to cancel the training process if performance deteriorated. In such case the previous version of the model was loaded and saved. This procedure is called *Early Stopping* (ES). Figure 6 shows the validation accuracy over 50 epochs without the implementation of the ES function. Using neural network presented above, we built various models that differ in prediction value. The first model was built to predict 12 Frames into the future, which roughly equals a time value of 50ms (49.992ms). This also equals the mean of the measured latency in section 3. Utilizing this model we are able to create a zero latency OptiTrack motion capturing environment. Besides the 12 frame model, we trained and optimized a 25 frame model, which equals to a time value of roughly 104ms (104.15ms). With this model, we are not only able to compensate for system latency, but also predict actual future movements of human avatars. Both models are extremely well suited to their task, with the 12 frame model achieving a prediction accuracy of 94.2 percent on an unknown validation data set. The 25 frame model, still achieved a very good accuracy of 91.4 percent. To be able to test and evaluate the developed models it was necessary to interrupt the OptiTrack stream, intercept the streamed frames, feed the model with input parameters and return the prediction to the stream. To achieve this, we have developed our own *Intercepter* client, which is presented in the following subsection.

Intercepter Client

The developed client has several time-critical tasks:

(1) Unpacking the bit-stream provided by OptiTrack
(2) Feeding the raw data into the loaded Tensorflow model
(3) Waiting for model interference and accepting the prediction
(4) Repacking the prediction of the model into the bit-stream

The intercepter client had to do this for each individual packet received, because one packet contained exactly one frame. This creates the time criticality already mentioned above. OptiTrack streams with a frequency of 240Hz, which means that every 4.16ms a new packet arrives at the *Intercepter* Client. The client has a maximum of about 4.16ms seconds before a new packet has to
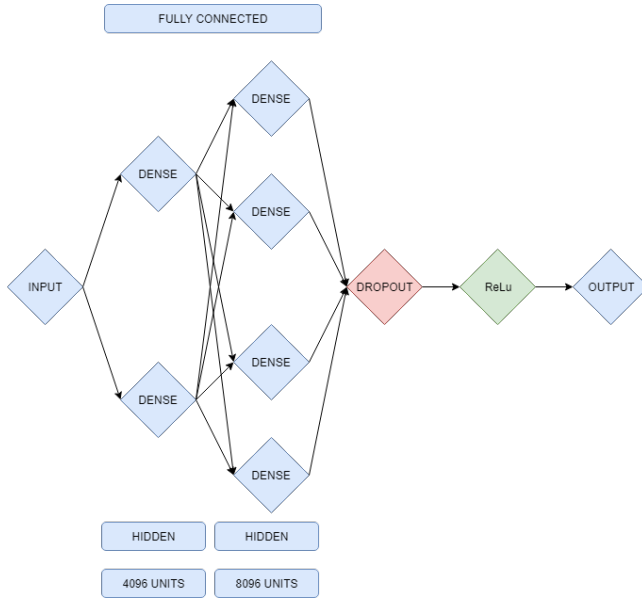
Figure 5: Shows a schematic representation of the developed network. The two hidden layers are fully connected and are composed of 4096 and 8096 units respectively. In combination with input and output layers, this amounts to a cumulative number of about 9 million trainable parameters.
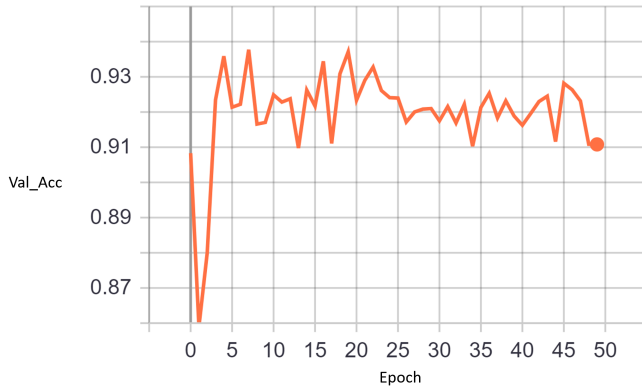

Figure 6: Showing validation accuracy of the trained network before implementing ES-function. The figure shows that a fixed number of epochs does not result in the optimized outcome.

be processed. During this time the client must have processed the packet completely and sent it onwards, otherwise the socket is blocked and the incoming packet can only be discarded. The discarded packages would cause the target application to tremble in the avatar representation. Due to the missing information, a continuous representation of the avatar would not be possible. Unpacking (1) and repacking (4) the data to a bit-stream takes the final client no significant amount of time, and thus can be neglected. Consequently the client has less then 4.16ms time to

get the prediction from the model. To be able to to achieve this, the client pre-loads all used models (12 frame model and 25 frame model), thus having them ready at run-time. Additionally the client sets the loaded model as the default Tensorflow graph, meaning the graph does not have to be rebuilt every time a prediction is needed. In addition to these measures, we evaluated the prediction time of various network architectures in advance. We evaluated the classical DNN structure used here, a classical RNN, an RNN based on CudnnLSTM units and an RNN based on CudnnGRU units. Figure 7 shows the evaluation of the measurements. It is shown that only a classical DNN structure can be considered for the presented problem, since all other architectures do not reliably remain below the technical 4.16ms limit.
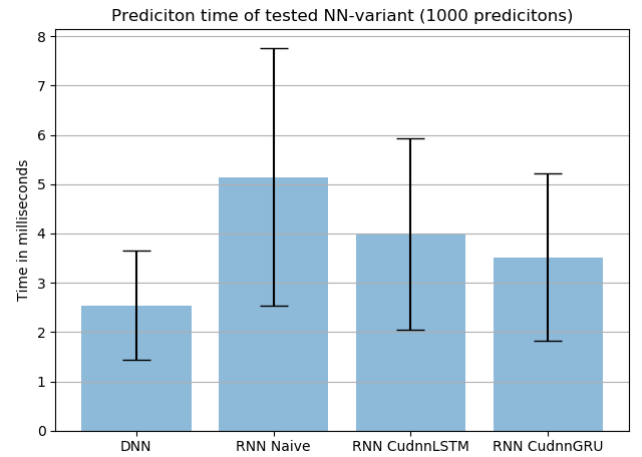

Figure 7: Shows the average time needed for different models to predict one single frame. Each model needed to predict 1000 frames (n=1000) with 87 float values as input (X) and 87 float values as output (Y). Not considered in this graph is the time needed to train the corresponding model or prediction quality.

6 VALIDATION OF THE MODEL

To evaluate the presented system we conducted an additional study.

Participants

We invited 12 test persons (6f, 6m) with an average age of 26,5 (SD 3,4) to participate in the study. Requirements for the participants were identical to those presented in the data acquisition study. At the start of the evaluation study participants had to sign a consent form again to accept the collection of personal data as part of the used questionnaire and motion capturing process.

Tasks and Conditions

The study consisted of 2 tasks with 3 conditions each. The three different tested conditions were:

(1) VR MoCap with 0ms prediction, real system latency
(2) VR MoCap with 50ms prediction, zero system latency
(3) VR MoCap with 104ms prediction, 54ms future prediction

Both tasks were performed by the participants in VR. For the study, a replica of the real room in which the subjects were at the time of the study was used. Figure 8 shows the real world replica in which the study was conducted.

Figure 8: Showing the virtual reality environment in which the evaluation study was conducted.

The first task was based on the task from the data collection study. In doing so, the test persons again had to simply perform movements from everyday life in VR, such as doing rotating their arms. The second task consisted of a performance task in which the test persons were to complete a given task as quickly as possible. The given task consisted of the respondent touching alternately placed objects only using the same hand in VR. These objects were also present in the real world, this way the test person could experience real haptic feedback when touching the objects in VR. The count of contacts was recorded by the experimenter. The combination of conditions and tasks resulted in 6 distinct test scenarios for each respondent:

(1) T0: Interaction Task with 0 ms prediction
(2) T12:Interaction Task with 50 ms prediction
(3) T25:Interaction Task with 104 ms prediction
(4) P0: Performance Task with 0 ms prediction
(5) P12:Performance Task with 50 ms prediction
(6) P25:Performance Task with 104 ms prediction

After each scenario, the subjects had to complete a questionnaire and answer three short qualitative questions. The questionnaires were displayed directly in VR and the participants were able to complete them without having to remove the HMD. This procedure saves time and the respondent is not as easily influenced or distracted by the real world [31]. The questionnaire itself contained 3 questions of the igroup presence questionnaire (IPQ) [13] which have shown to have the highest reliability compared to similar standardized presence questionnaires [31]. Additionally we used 7 questions to evaluate the virtual limb ownership of the test subject. The used questions were originally designed to evaluate limb ownership of the hand in VR [32]. We adapted the questions to evaluated the limb ownership of the whole body. For each statement participants chose a rating on a 5-point Likert scale ranging from 1 for *I strongly disagree* to 5 for *I strongly agree*. Table **??** shows all questions asked via questionnaire, as well as the means of the answers. The qualitative questions after each scenario aimed to reveal model preferences and or remarks to the overall experiences. The following questions were verbally asked and recorded:

(1) QL1: Did you have the feeling that the virtual environment was in tune with reality?
(2) QL2: Did you have the feeling that your real movements matched your virtual movements?
(3) QL3: Did you feel nauseous during the last part of the study?

Upon finishing a task (T or P) we asked the participants which model they liked best to conclude the task. Once all the tasks had been completed, we asked them which model they liked the most overall. The participants did not know which model they had tested, but named a unique identifier that could be assigned to the model by the experimenter. Findings of the conducted study will be pointed out in the next section.

Data Analysis

First, the data collected were examined for their significance. The IPQ1, IPQ6 and IPQ10 questions of the IPQ questionnaire that we used can be considered and evaluated as one unit. The questions of the performance (P) task and the interaction task (T) were considered separately. Neither the distribution of the answers in the performance task nor the distribution of the answers in the interaction task could be proven to be significant with the help of the Friedman test (p-value T-Tasks: > 0.9, p-value P-Tasks: >0.9). Figure 8 9 shows the calculated means as well as the corresponding standard derivation. Secondly, we examined the collected data in terms of limb-ownership (VL1 to VL7, shown in **??**). The collected data regarding the limb-ownership questionnaire needs to be evaluated item by item. Before being able to analyze the data, the data set containing the VL-questions needed to be transformed to a arbitrary distribution. We used the ART method [45] available in R through the CRAN content delivery network [5]. After transformation, the individual measurements for each questions were entered into a repeated measure analysis of variance (ANOVA). Again, the questions data was separated by the two performed tasks interaction (T) and performance (P). We found that VL4 in the performance task *"It seemed to me that my body was placed in the virtual world"* significantly changed in dependencies of the used model,*p = 0.029, F-value = 4.187*. Furthermore, question VL3 *"My movements in virtual reality were performed by me"* asked in the interaction task has also revealed significant differences, *p = 0.03, F-value = 7.917*. All other questions did not reveal any significant differences. The above findings will be discussed in the section *Discussion and Conclusion*.

The performance task was a repeated measure, it has to be expected that a certain learning effect can be observed in the test persons during the measurement. To account for this effect, the sequence was considered an influencing factor in the data analysis. **??** shows the mean counts for each model, as well as the confidence interval for each bar. Although a certain trend could be recognized, ANOVA showed that there is no significant effect, *p = 0.53*.

Qualitative Analysis

All participants stated that they felt physically and mentally well after the study. The qualitative data indicate that none of the subjects felt nauseous or dizzy during or after the study. No *motion sickness* was induced by the use of either of our models (QL3). Furthermore the qualitative feedback showed that the test persons had the feeling that the real world is depicted in the VR (QL2).
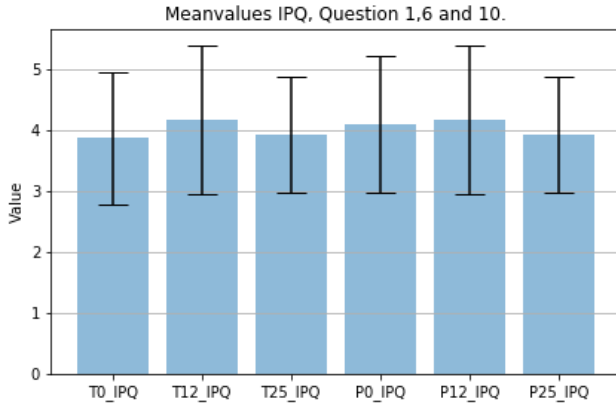
Figure 9: Shows IPQ means for each condition as well as SD as error bar. No significant difference between the variances could be proven.

Only one respondent answered the question in the negatively and went on to explain that especially the strange representation of the fingers had a strong negative influence on the immersion feeling. The problem described by the test person can probably be traced back to the very large tracking range of the OptiTrack system. At greater distances, the accuracy of the representation decreases due to transmission and differentiation problems. The motion capturing system is no longer able to clearly identify the individual fingers. All participants affirmed the question *Did you have the feeling that the virtual environment was in tune with reality?*(QL1). Particularly acclaimed was the realistic representation of the real world in VR. Surprisingly, when asked about the model that was most liked, half of the subjects preferred the 12F model for the interaction task (T) and thus at least qualitatively confirmed the assumption that zero latency can increase the overall feeling of VR. Only three subjects preferred the 0F model for the interaction task, another three subjects liked the 25F model the most. There were no differences in the performance task(P), all three models were named by four participants each. Answering the concluding question as to which model the participants had liked the most on the whole, six participants favored the 12F model again. Two participants chose the 25F model as the best model and another three participants chose the 0F model. One participant could not decide on a best model.

7 GENERAL DISCUSSION

Based on the evaluation of the IPQ it can be determined that the use of the presented models does not improve the immersion feeling. At the same time, however, it also shows that the above-mentioned immersion feeling is not impaired by the use of the models. There can be several possible explanations for this situation. One reason could be that ten out of twelve participants had no or very little previous experience in and with VR. It is possible that the participants were so overwhelmed by the motion capturing system that they were not able to perceive a difference in the presented condition. To solve this problem, a stronger contrast between the conditions should have been chosen during the design of the conducted study. The

differences between a 0 frame model and a 100 frame model, which corresponds to a prediction of 416ms (416.66ms), might have been noticed by a subject with very little previous experience. Another reason for the lack of significance may be the small sample size (n=12). By increasing the sample size, effects could be shown which have not been significantly proven in the presented study.

The evaluation of the questions about limb ownership showed significant differences in answers for questions VL4 (P) and VL3 (T). Even though, a significant result was obtained by appropriate test procedures, there was no actual detected effect. This result is probably attributable to a type 1 error. Type 1 errors occur when randomly increased measurements lead to a rejection of the null hypothesis, although it should have been accepted. The assumption that there is a type 1 error is further reinforced by the relatively weak significance at p-values around 0.03. A reduction of the accepted significance level, for example to 0.01, would reduce type 1 error probability, but at the same time increase the error probability for a type 2 error.

The evaluation of the performance task showed no significant results. Although a general trend can be seen in the data and in figure ??, no universal effect may be established at this point. Three points could play a crucial role here: Firstly, the sample size is insufficient, and the task could benefit enormously from an increase. Secondly, the performance task was not ideally planned and could have been optimized. Instead of considering learning effects in the evaluation, a learning phase should have been implemented in the study. In this learning phase, the subjects could have familiarized themselves with the task and the environment. This phase would not be considered in subsequent evaluations. In this way, the actual performance of the test persons could be examined strongly decoupled from learning effects. Furthermore, test persons should be selected based on VR experience to prevent participants feeling intimidated or too impressed by the system. The last point influencing the outcome of this task is the chosen measurement. We only measure one variable - the number of times the test persons touches the objects. In order to be more reliable we could have established a second or third variable. Additional performance variables could have consolidated the result and made it more credible.

8 CONCLUSION

To conclude this paper, we successfully developed and implemented a reliable framework to test the latency of any motion capturing system coupled to a Unity application and a HMD. We successfully developed a neural network which is able to dependable predict human avatar movement in VR up to a value of 100ms. We successfully intercepted the available OptiTrack stream and feed the stream our own predicted values. Unfortunately, we were not yet able to evidently prove that a zero latency system is potentially able to increase the feeling of immersion or user performance, yet. In future work we will keep on working on the presented system. Firstly, we need to develop a more suitable task scenario for the performance task measuring more than only one variable. Secondly, we want to improve the neural network and implement additional prediction values. Thirdly, a proper study needs to be designed because our chosen condition probably were too similar to be noticed by the participants. In a future design we will implement a

condition which clearly distinguish itself from the 0 Frame baseline model. Furthermore, a future study should focus to a greater extent on the actual avatar representation and self-awareness of the subjects in VR. Which could, for instance, be achieved by placing mirrors in the virtual environment.
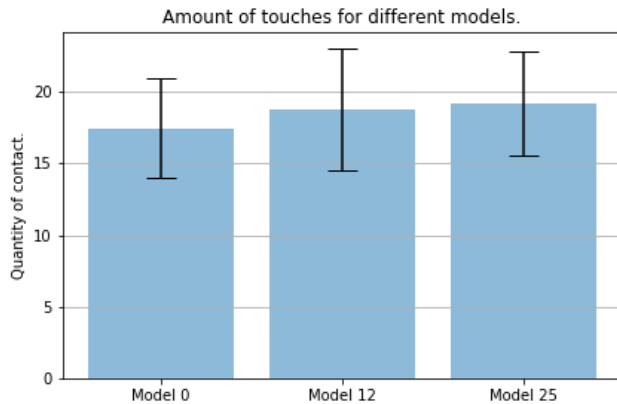

Figure 10: Shows the average hit count for the performance Task (P), as well as SD as error. Although a certain trend is presumably, there is no actual significant effect provable.

REFERENCES

[1] Arduino. 2015. Arduino. https://www.arduino.cc/. Accessed on 2019-05-21.
[2] Francois Chollet. 2019. Keras. https://keras.io. Accessed on 2019-08-12.
[3] Francois Chollet. 2019. Keras. https://keras.io/callbacks/. Accessed on 2019-08-12.
[4] Philo Tan Chua, Rebecca Crivella, Bo Daly, Ning Hu, Russ Schaaf, David Ventura, Todd Camill, Jessica Hodgins, and Randy Pausch. 2003. Training for physical tasks in virtual environments: Tai Chi. In *IEEE Virtual Reality, 2003. Proceedings.* IEEE, 87–94.
[5] CRAN. 2015. The CRAN Projekt. https://cran.r-project.org/web/packages/ART/index.html. Accessed on 2019-08-30.
[6] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. 2017. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 542, 7639 (2017), 115.
[7] Partha Ghosh, Jie Song, Emre Aksan, and Otmar Hilliges. 2017. Learning human motion models for long-term predictions. In *2017 International Conference on 3D Vision (3DV).* IEEE, 458–466.
[8] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics.* 315–323.
[9] Mar Gonzalez-Franco and Jaron Lanier. 2017. Model of illusions and virtual reality. *Frontiers in psychology* 8 (2017), 1125.
[10] Kun Han, Dong Yu, and Ivan Tashev. 2014. Speech emotion recognition using deep neural network and extreme learning machine. In *Fifteenth annual conference of the international speech communication association.*
[11] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).
[12] HTC. 2015. VIVE Pro. https://www.vive.com/de/product/vive-pro/. Accessed on 2019-05-21.
[13] igroup. [n. d.]. igroup presence questionnaire. http://www.igroup.org/pq/ipq/index.php. Accessed on 2019-05-22.
[14] Xiaomo Jiang and Hojjat Adeli. 2005. Dynamic wavelet neural network model for traffic flow forecasting. *Journal of transportation engineering* 131, 10 (2005), 771–779.
[15] Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. *Journal of basic Engineering* 82, 1 (1960), 35–45.
[16] Nguyen Lu Dang Khoa, Kazutoshi Sakakibara, and Ikuko Nishikawa. 2006. Stock price forecasting using back propagation neural networks with time and profit based adjusted weight factors. In *2006 SICE-ICASE International Joint Conference.* IEEE, 5484–5488.
[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[18] Merel Krijn, Paul MG Emmelkamp, Roeline Biemond, Claudius de Wilde de Ligny, Martijn J Schuemie, and Charles APG van der Mast. 2004. Treatment of acrophobia in virtual reality: The role of immersion and presence. *Behaviour research and therapy* 42, 2 (2004), 229–239.
[19] Jingjing Liu, Shaoting Zhang, Shu Wang, and Dimitris N Metaxas. 2016. Multispectral deep neural networks for pedestrian detection. *arXiv preprint arXiv:1611.02644* (2016).
[20] Zhicheng Liu and Jeffrey Heer. 2014. The effects of interactive latency on exploratory visual analysis. *IEEE transactions on visualization and computer graphics* 20, 12 (2014), 2122–2131.
[21] Kin C Luk, James E Ball, and Ashish Sharma. 2001. An application of artificial neural networks for rainfall forecasting. *Mathematical and Computer modelling* 33, 6-7 (2001), 683–693.
[22] Julieta Martinez, Michael J Black, and Javier Romero. 2017. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2891–2900.
[23] Michael Meehan, Sharif Razzaque, Mary C Whitton, and Frederick P Brooks. 2003. Effect of latency on presence in stressful virtual environments. In *IEEE Virtual Reality, 2003. Proceedings.* IEEE, 141–148.
[24] Naturalpoint. 2015. Optitrack - Motion Capturing System. https://optitrack.com/. Accessed on 2019-05-21.
[25] Matthew Price and Page Anderson. 2007. The role of presence in virtual reality exposure therapy. *Journal of anxiety disorders* 21, 5 (2007), 742–751.
[26] Nicolas Rahmouni, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. 2017. Distinguishing computer graphics from natural images using convolution neural networks. In *2017 IEEE Workshop on Information Forensics and Security (WIFS).* IEEE, 1–6.
[27] Simon Riches, Soha Elghany, Philippa Garety, Mar Rus-Calafell, and Lucia Valmaggia. 2019. Factors Affecting Sense of Presence in a Virtual Reality Social Environment: A Qualitative Study. *Cyberpsychology, Behavior, and Social Networking* (2019).
[28] E.W. Saad, Thomas Caudell, and Donald Wunsch. 1999. Predictive head tracking for virtual reality, Vol. 6. 3933 – 3936 vol.6. https://doi.org/10.1109/IJCNN.1999.830785
[29] Matthew NO Sadiku. 2014. *Elements of electromagnetics.* Oxford university press.
[30] Maria V Sanchez-Vives and Mel Slater. 2005. From presence to consciousness through virtual reality. *Nature Reviews Neuroscience* 6, 4 (2005), 332.
[31] Valentin Schwind, Pascal Knierim, Nico Haas, and Niels Henze. 2019. Using Presence Questionnaires in Virtual Reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems.* ACM, 360.
[32] Valentin Schwind, Lorraine Lin, Massimiliano Di Luca, Sophie Jörg, and James Hillis. 2018. Touch with foreign hands: the effect of virtual hand appearance on visual-haptic integration. In *Proceedings of the 15th ACM Symposium on Applied Perception.* ACM, 9.
[33] Rajalingappaa Shanmugamani. 2018. *Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras.* Packt Publishing Ltd.
[34] Fabio Alexandre Spanhol, Luiz S Oliveira, Caroline Petitjean, and Laurent Heutte. 2016. Breast cancer histopathological image classification using convolutional neural networks. In *2016 international joint conference on neural networks (IJCNN).* IEEE, 2560–2567.
[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.
[36] Mate Szarvas, Akira Yoshizawa, Munetaka Yamamoto, and Jun Ogata. 2005. Pedestrian detection with convolutional neural networks. In *Intelligent vehicles symposium.* 224–229.
[37] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. 2013. Deep neural networks for object detection. In *Advances in neural information processing systems.* 2553–2561.
[38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).
[39] Google Brain Team. 2015. TensorFlow. https://www.tensorflow.org/. Accessed on 2019-05-21.
[40] Google Brain Team. 2015. TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/SGD. Accessed on 2019-05-21.
[41] Unity Technologies. 2005. Unity. https://unity.com/. Accessed on 2019-05-21.
[42] Denis Tomè, Federico Monti, Luca Baroffio, Luca Bondi, Marco Tagliasacchi, and Stefano Tubaro. 2016. Deep convolutional neural networks for pedestrian detection. *Signal processing: image communication* 47 (2016), 482–489.
[43] Iis P Tussyadiah, Dan Wang, Timothy H Jung, and M Claudia tom Dieck. 2018. Virtual reality, presence, and attitude change: Empirical evidence from tourism. *Tourism Management* 66 (2018), 140–154.
[44] Hans Georg Unger and Johann Heyen Hinken. 1991. *Elektromagnetische Wellen auf Leitungen.* Hüthig.

[45] Jacob O Wobbrock, Leah Findlater, Darren Gergle, and James J Higgins. 2011. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In *Proceedings of the SIGCHI conference on human factors in computing systems.* ACM, 143–146.

[46] Jiann-Rong Wu and Ming Ouhyoung. 2000. On latency compensation and its effects on head-motion trajectories in virtual environments. *The visual computer* 16, 2 (2000), 79–90.

APPENDIX

Question	Questionnaire	T0	T12	T25	P0	P12	P25
In the computer generated world I had a sense of being there.	IPQ1	3.8±0.7	4.1±1.2	3.9±1.0	4.1±1.1	4.0±1.1	4.25±0.8
I feel present in the virtual space.	IPQ6	3.6±1.6	4.0±1.6	3.8±0.9	3.8±0.9	3.8±1.0	3.76±1.3
I was completely captivated by the virtual world.	IPQ10	4.1±1.1	3.8±1.5	4.0±0.9	3.6±1.8	4.0±1.5	3.9±1.0
I had the feeling that my virtual body was in the same position as in reality.	VL1	3.1±1.3	3.6±1.2	4.1±1.6	3.8±1.3	4.0±1.4	3.6±1.3
I had the feeling that the touch in reality was created by the touch in the virtual world.	VL2	4.6±0.9	4.5±0.7	2.6±1.6	4.3±0.9	3.8±1.5	4.3±0.9
My movements in virtual reality were performed by me.	VL3	3.9±0.8	4.3±0.7	4.0±0.9	3.9±0.5	5.1±0.9	4.3±0.8
It seemed to me that my body was placed in the virtual world.	VL4	3.2±1.2	2.7±1.4	2.8±1.0	3.1±1.3	3.1±0.9	3.4±1.3
The virtual body resembled my real body in form and appearance.	VL5	3.2±1.3	3.3±1.3	3.4±1.0	3.3±1.1	3.4±1.0	3.1±1.5
My feelings were influenced by what I saw in virtual reality.	VL6	3.3±1.7	3.9±0.9	3.2±1.2	3.8±1.2	3.8±1.2	3.8±1.1
My virtual body felt like my real body.	VL7	4.1±0.8	3.8±1.5	3.8±0.9	4.1±1.1	3.9±1.0	4.3±0.6

Table 1: Questions asked after each test case. Also showing the means of the answers, as well as SD. Test persons answered these question directly in VR.