Using Artificial Neural Networks to Predict Avatar Movement in Virtual Reality

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

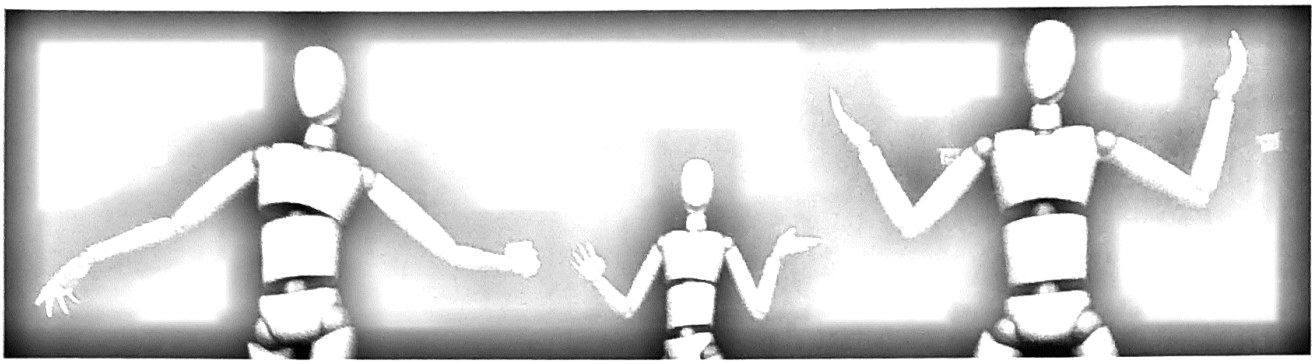

Figure 1: Motion captured avatars showing different postures.

CCS CONCEPTS

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

KEYWORDS

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

ACM Reference Format:
Jakob Fehle, David Halbhuber, and Jonathan Sasse. 2019. Using Artificial Neural Networks to Predict Avatar Movement in Virtual Reality . In *Proceedings of ACM Conference (Conference'19).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

State-of-the-art motion capturing systems like NaturalPoint's OptiTrack [7] can be utilized to transfer ~~an objects~~ movement almost perfectly to a computer generated space. This computer generated space can be used to integrate the objects movement into several different kinds of computer applications, such as games, movie recording software or business applications. In a human-centered approach the movement of a human gets transmitted to the application to allow the user to directly interact with the software through natural movement.

One step further in directly integrating natural human movement and behaviour into these kind of applications can be obtained by combining state-of-the-art motion capturing devices with head mounted displays (HMD) like HTC's Vive[3]. It allows the user to see the generated computer space, in the so called Virtual Reality (VR), directly without deflection. By combining a motion capturing system and a HMD the user is able to see its avatars movement in VR through its own eyes. Such systems have already been tested widely in the field of electronic learning. One example of this is the work of Chua et al. [2] in which the authors propose a method to learn Taijiquan, an ancient Chinese form of material arts, using a motion capture system in combination with a HMD. Although this type of system is promising 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 latency arises in the course of communication between devices. In concrete terms, this means that the user using 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 this movement to the HMD, which itself needs some time to display the received data correctly. The processing and

fascinating :)

new paragraph

Conference'19, May 2019, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

transmission time results in a perceptible latency and can ultimately lead to a reduced feeling of presence[8].

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]. **Figure 2** shows a generalized illustration of a motion capturing system in combination with a HMD. The red arrows depict the way the signal, in this cases movement of the user, is processed which ultimately leads to latency. The aim of this work

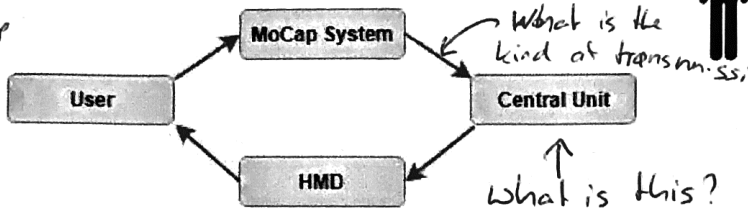

Figure 2: Illustrating the way a signal, caused by movement of the user, has to take in order to finally get displayed to the user trough the head mounted display. This transmission and processing time is perceivable by the user in delayed presentation of the own movement in VR.

is to solve the problem of latency through a movement prediction mechanism based on recurrent artificial neural networks (RNN). The presented system is capable of predicting user movement by calculating probabilities of movement based on a prior learned data set. RNNs, 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 [5] or the amount of expected rainfall in a certain region [6].

The remaining paper is structured as follows: The next chapter explains the developed system for determining the latency and predicting users movement using the motion capturing system and neural networks. Chapter 3 describes the designed study to evaluate the presented approach to increase presence.

2 SYSTEM

The entire system consists of two technical subsystems which on the one hand are to be understood as requirements for the movement prediction system to be developed and on the other hand represent the actual movement prediction system itself.

2.1 Latency Test Framework

To minimize the latency between the users actual movement and the shown view in VR using the developed movement prediction system, it is essential to determine the exact latency first. In order to accomplish an highly accurate measurement, independent of confounding variables, a Latency Test Framework (LTF), based on various hardware sensors coupled with a range of software interfaces was developed. The hardware part of the LTF consists of an Arduino microcontroller [1] which is linked to a shock sensor and a light sensitve photosensor. NaturelPoints OptiTrack motion capturing system is used to capture the motion of an user. The captured motions are sent to a computer running the 3D game engine Unity[12] coupled to HTCs HMD Vive. OptiTrack and HTCs Vive are used in the final movement prediction system as well. On the software side, multiple applications and scripts are used to react to the deflections of the hardware sensors. **Figure 3** depicts the workflow of the LTF. The figure shows how a user drops an object,

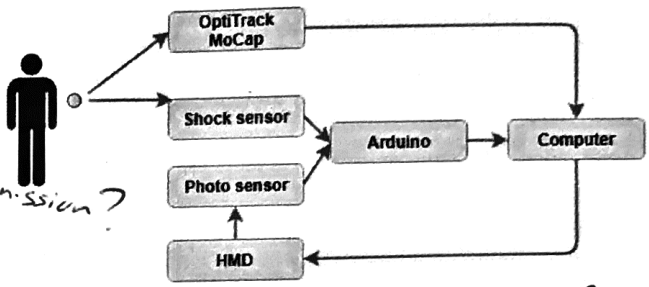

Figure 3: Latency Test Framework (LTF)

in the case of the figure a small blue ball. The object and its fall are tracked by the motion capturing system. When the object hits the ground the shock sensor is triggered by the resulting vibration. The Arduino connected to the shock sensor is informed about the event, which in turn passes it on to the central computer. A Python script running on the central computer is activated by the impact and records the exact time in microseconds. This marks the first time stamp in the latency test. Meanwhile, the OptiTrack system receives and processes the movement of the object and passes this information onto the the central computer. The computer, running the Unity 3D game engine, processes the movement data and renders the object to the HMD. The Unity application is configured to register the objects collision with the ground. Upon collision Unity starts a function to disable the HMD. The deactivation is registered by the photo sensor, since its able to detect change in light. As soon as the photo sensor is triggered a signal gets passed to the Arduino, which then passes this on to the central computer, which again sets a time stamp. By comparing the two gathered time stamps it is possible to accurately determine the time needed for the system to register an object and display it on the HMD.

This period of time represents the latency of the system excluding the time needed to communicate between Arduino and computer, which is already known.

2.2 Movement Prediction System

With the latency value at hand a second subsystem for predicting and rendering the actual body movements was built.

2.2.1 Motion data acquisition interface. For minimizing the effort of transferring data from the motion capturing system towards a comma-separated value file (CSV) a Python script was written. The process of exporting the captured data into CSV, importing it into Microsoft Excel and merge with previously acquired datasets was not satisfying in terms of time consumption. The purpose of the developed script is to collect and merge the streamed OptiTrack data and forward them towards the neural network.

2.2.2 Recurrent artificial neural network. For the actual predicting part of the system we used a neural network based on Google Brain

Teams Tensorflow [11]. Since the system predicts the movement of a user in VR, it must be able to process time-dependent data efficiently. RNNs are known to manage this type of data very well as they are able to process data bidirectionally [10].

2.2.3 Network stream interceptor. Another Python interface for intercepting the data exchange between the neural network and the Unity application was required. The resulting data from the network process are collected and converted into a format the Unity [12] application can handle.

2.2.4 Unity application. As last component of the movement prediction system a Unity application was required. This application receives a data stream from the neural network via the interceptor script. It then processes and renders the skeleton using the predicted marker position point values. As result the system user can see its own body movements inside the virtual reality without latency and problematic side effects.

3 STUDY DESIGN

In order to investigate the effect of latency reduction of a VR system on the presence and immersion of the user, we conducted a user study with 30 participants. Since no participant should be exposed to the test environment twice, the total number of participants was divided into two groups in order to investigate the feeling of presence before and after integrating the prediction mechanism of the neural network. Both groups carried out predefined tasks while wearing the HMD and a motion capturing suit coupled to the OptiTrack system. The experiences of the users were evaluated using the Igroup Presence Questionnaire (IPQ) [4], which comprises the three categories spatial presence, involvement and experienced realism.

REFERENCES

[1] Arduino. 2015. Arduino. https://www.arduino.cc/. Accessed on 2019-05-21.
[2] 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.
[3] HTC. 2015. VIVE Pro. https://www.vive.com/de/product/vive-pro/. Accessed on 2019-05-21.
[4] igroup. [n. d.]. igroup presence questionnaire. http://www.igroup.org/pq/ipq/index.php. Accessed on 2019-05-22.
[5] Xiaomo Jiang and Hojjat Adeli. 2005. Dynamic wavelet neural network model for traffic flow forecasting. *Journal of transportation engineering* 131, 10 (2005), 771–779.
[6] 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.
[7] Naturalpoint. 2015. Optitrack - Motion Capturing System. https://optitrack.com/. Accessed on 2019-05-21.
[8] 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).
[9] Maria V Sanchez-Vives and Mel Slater. 2005. From presence to consciousness through virtual reality. *Nature Reviews Neuroscience* 6, 4 (2005), 332.
[10] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.
[11] Google Brain Team. 2015. TensorFlow. https://www.tensorflow.org/. Accessed on 2019-05-21.
[12] Unity Technologies. 2005. Unity. https://unity.com/. Accessed on 2019-05-21.

why?

you can do within → probably people notice it more likely

↓ less variance

↓ less participants

↓ less time you need in putting people into smelling suits

Parameterize the Generation of Realistic Faces

Marlene Bauer
marlene.bauer@stud.uni-regensburg.de
University of Regensburg
Regensburg, Germany

Florian Habler
florian.habler@stud.uni-regensburg.de
University of Regensburg
Regensburg, Germany

Konstantin Seitz
konstantin.seitz@stud.uni-regensburg.de
University of Regensburg
Regensburg, Germany

ABSTRACT

Virtually created, realistic faces offer added value in many applications. Current solutions like the faceMaker produce images that do not yet look like realistic faces of real people, but offer the ability to fine tune facial features in avatars. However, neural networks such as Nvidia's Generative Adversarial Network (GAN) *StyleGAN* are able to create realistic-looking faces that can not be changed in detail. We want to find a solution to combine both worlds in order to create realistic faces from small-scale selectable parameters. To do this, we firstly will conduct a study in which attendees try to faithfully replicate faces of NVIDIA-generated images ~~in order~~ to find parameters that are suited for ~~frontal~~ face making. After that, we want to manipulate the necessary noise vector input for GANs in order to control the creation of realistic looking faces and finally integrate this feature into faceMaker.

ACM Reference Format:
Marlene Bauer, Florian Habler, and Konstantin Seitz. 2019. Parameterize the Generation of Realistic Faces. In *Regensburg '19: Praxisseminar*. ACM, New York, NY, USA, 2 pages.

Figure 1: Current interface of the faceMaker.

1 INTRODUCTION

An often important feature of online services is the creation of virtual avatars representing oneself or an online persona. This kind of individual customization is often seen in video games where users can create their own character but are often restricted by in-game user interfaces and the graphics of the used game engine. Therefore, users are not able to create realistic looking avatars even if they wanted to. The creation of a realistic looking avatar that is based on their preferred digital avatar would offer them a new dimension of individualization. Computational created faces not only add value to games, but offer a variety of applications, for example pre-visualizations prior to cosmetic surgery or identikit generation for police usage. GANs like Nvidia's [3] are able to create realistic looking faces based on trained models of human faces and can recreate poses, eyes, hair based on different input vectors. In this paper, we first want to update the already existing faceMaker tool from Schwind et al. [5] that will output a digital face along with noise vectors that will be used as inputs generate realistic looking faces leveraging an already trained GAN.

2 UPDATING FACEMAKER

To recreate faces we will have participants use the faceMaker tool from Schwind et al. [5]. We want to reduce the overall number of sliders and thus the parameters of the faceMaker to eliminate superfluous. The parameters "make-up: lip stick", "make-up: rouge", "throat size", "ear-size" and "eye-height" foe example are found to be used less for the task of creating super heroes or villains [4]. Figure 1 shows the current state of the faceMaker. We plan to conduct a preliminary study in which participants are supposed to replicate faces gathered from already existing computationally created faces. Figure 2 shows a realistic looking face generated by an open-sourced Nvidia GAN. We will invite ten people, each trying to reproduce a face generated by an already existing Generative Adversarial Network. This way we will have corresponding noise vectors to the reproduced face that we can later use to manipulate certain aspects of input vectors in order to create different facial features like skin tones. The slider positions and a subsequent survey are intended to determine which sliders in faceMaker are meaningful and which are rather less used and may not be used at all in order to create a frontal facing avatar. Based on those results, we will modify the user interface and modifiable parameters of faceMaker. In our follow up study, attendees use our redesigned faceMaker to create faces that, in the final step, will be further processed into realistic faces. Questionnaires are used to measure quantitative feedback like the user-friendliness, user experience and usability of the system, as well as the quality of the final result, task-success-rate and task-time. By observing and questioning users about their tasks we will also collect qualitative feedback that can be used to further improve faceMaker.

3 CREATION OF REALISTIC LOOKING FACES USING GANS

In our studies, we found two different approaches to generating realistic looking faces based on avatars. The first approach is to

Regensburg '19, 2019, Regensburg, BY

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

Parameterize the Generation of Realistic Faces

Anna-Maria Auer + Affiliation + E-Mail
Michael Hebeisen
Jonathan Seibold

Figure 1: NVIDIA Neural Network

ACM Reference Format:
Anna-Maria Auer, Michael Hebeisen, and Jonathan Seibold. 2019. Parameterize the Generation of Realistic Faces. In *FSM 18/19: Research Seminar Master Media Informatics, Summer Term 2019.* ACM, New York, NY, USA, 2 pages.

1 INTRODUCTION ~~AND MOTIVATION~~

Virtually designing human faces is part in a variety of areas. These artificial faces can be used for example as preview for plastic surgeries, as identikits or as part of avatars in a video game. However, people who work in these fields may not always be experts in designing 3D models or editing images, which is why there is a need for a easy and intuitive software to design faces. The application FaceMaker[1] achieves this intuitive process by using multiple sliders for different characteristics of the face. Users can manipulate these sliders and will immediately see corresponding changes on the face located in the center of the application. FaceMaker creates a 3D avatar, the result is therefore not photorealistic. In some cases, like for identikits, photorealistic images of a face would be preferred as output of such a system. We want to implement a system that accomplishes the generation of photorealistic faces using ~~NVIDEA~~ NVIDIA StyleGAN[2]. In a follow up study, we want to examine, if faces created with our application look realistic. On top of that, we are interested if all of the available sliders in FaceMaker are necessary for realistic face creation.

FSM 18/19, 2018, University of Regensburg, Germany
2019.

nice that you are interested. Say why it's important!

2 RELATED WORK

~~In order~~ To develop a neural network that is able to create realistic faces out of FaceMaker-avatars we need image to image translation. These has been done by several scholars with differing subjects before. The goal of such networks is to learn the mapping between an input and an output-image. For example, Zhu et al. [4] ~~used~~ who trains unpaired image-to-image translation to train a network that is able to transform horses into zebras and zebras into horses. The authors pursue this approach because of lacking paired image samples. The here presented project faces the same issues. Currently, there is no available database that pairs FaceMaker-faces with real faces. For training a neural network with paired image-to-image translation, about 1000 paired samples are needed. Others, like Wang et al. [3] follow similar approaches. As mentioned before, it is also possible to train a neural network with paired samples. For example, this is done by Zhu et al. [5] who train a network that is able to provide multiple possible output images based on one input image.

realistic

is it intuitive? it's easy and obvious

i think intuitive would be to adjust the face directly in the viewport

1000? (where do you get this number?)

3 SYSTEM SET-UP

The System's User Interface will be based on FaceMaker. We will keep the sliders used to manipulate the face. We will also keep a live preview of the face in the center of the application. We would like this preview to already show the photorealistic face, but if this is not possible, for example because composing the image needs to many resources, we would also be content with the usual 3D model of the face. In the latter case, we would add a button to finish the face design process and initiate the composition of the photorealistic image. Since we do not know which type of dataset is best suited for training our system, we developed two approaches. Our first approach is based on a set of unpaired images. Should this approach turn out to be unsuccessful, we will try a dataset with paired samples. Therefore, we would need a great amount of faces of NVIDEA StyleGAN and recreate them in FaceMaker.

4 STUDY DESIGN

Our aim is to develop a version of FaceMaker that provides the functionality to create realistic faces. Therefore, we pursue the following two research questions:

1. Is it possible to create realistic faces using FaceMaker with an additional neural network?
2. Which of the provided sliders are actually used during the generation of a face? Which sliders need to provide a higher or lower range of possible positions?

The study will be conducted by 30 participants who will be asked to create five faces with FaceMaker. The faces are based on real photographs which are given to the participants. Each participant will be given the same five photographs. This assures an adequate comparison between the generated faces. We assume a learning-effect when working with FaceMaker. Participants might get faster and better when repeating the task. Therefore each participant will receive the photographs in a different order. After generating the faces the participants will be asked to answer a questionnaire. Besides demographical data we collect participant's opinions towards the FaceMaker-UI in general and whether they think it is possible to create realistic looking faces. They also provide a self-assessment whether they were able to complete the tasks and if they noticed progress when working with FaceMaker several times. The questionnaire will also provide a section that asks about the sliders in specific. Our participants will provide their opinion towards which sliders are relevant for the tasks, which are confounding and which might be missing. We will also take field notes during the conduction of our experiment and will therefore be able to additionally provide a professional opinion on how well our participants were able to generate realistic faces. Such opinions can only be qualitative. However to provide an additional quantitative metric, we log the slider positions of each face that will be generated. This gives the following information about each slider:

1. Which position range is used in our dataset orf 30 x 5 generated images?
2. Which sliders are always left in the default position and could therefore be eliminated?

With these information FaceMaker-sliders could be updated.

REFERENCES

[1] V. Schwind K. Wolf, N. Henze. 2017. FaceMaker - A Procedural Face Generator to Foster Character Design Research. *Game Dynamics* (2017), 95–113.

[2] T. Karras S. Laine, T. Aila. 2018. A style-based generator architecture for generative adversarial networks. *CoRR* (2018).

[3] Chaoyue Wang, Chang Xu, Chaohui Wang, and Dacheng Tao. 2018. Perceptual adversarial networks for image-to-image transformation. *IEEE Transactions on Image Processing* 27, 8 (2018), 4066–4079.

[4] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.

[5] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. 2017. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*. 465–476.