universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Using GANs and Facial Landmark detection for Virtual Reality Conferencing

verfasst von | submitted by

Bernhard Kokesch BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2024

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 935
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Medieninformatik
Betreut von | Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Helmut Hlavacs

"You might not think that programmers are artists, but programming is an extremely creative profession. It's logic-based creativity."

– John Romero

Acknowledgements

First of all, I would like to thank my wife **Janine Kokesch**, who has accompanied and motivated me through all these years of writing and has not let me give up. She has my deepest gratitude for providing me with continuous encouragement and love throughout writing the thesis. I could not have done it without you. Thank you.

I would also like to thank my thesis advisor Professor **Helmut Hlavacs**, who always helped me when I was stuck and gave me new ideas and input as well as new viewing angles to solve technical problems.

I would also like to thank my **parents** and **grandparents**, who never stopped supporting and motivating me and my **friends** for testing.

I also want to thank my cats, **Perry** and **Kalani**, who played a big part in helping me calm down and clear my head.

Bernhard Kokesch

Abstract

Using GANs and Facial Landmark detection for Virtual Reality Conferencing

Because of the corona virus crisis, the need for possibilities to talk and interact with each other not in place is getting more important than ever. Virtual reality can fill this gap and make it possible to hold conferences, meetings and even family gatherings virtually without losing the possibility to interact with each other. Also with the latest milestones of virtual reality headsets and computer graphics a certain level of a good feeling inside the virtual world can be achieved and let it feel a little bit like the real world.

The goal of this work is to find a different approach to the "traditional" solutions like 3D heads and avatars. The idea is to leverage and use facial landmark detection and methods to bring motion to simple images and package it into a VR conferencing application. For this idea vid2vid was used in combination with GANs and motion transfer to "move" a single image. The resulting prototype has been evaluated in terms of the feeling of presence experienced inside the applications virtual environment.

For the evaluation 10 people tested the application and took a survey, based on the igroup presence questionnaire scale to measure the spatial presence, involvement and the sense of realness inside the virtual conference room afterwards.

The results of the conducted experiments indicate a positive outcome of the prototype and a positive feeling of presence and involvement which also indicates that the prototype is a promising foundation for the future.

Zusammenfassung

Aufgrund der COVID-19 Pandemie ist es wichtiger denn je, dass es Möglichkeiten gibt, miteinander zu sprechen und zu interagieren, ohne am selben Ort sein zu müssen. Die virtuelle Realität kann diese Lücke schließen und es ermöglichen, Konferenzen, Besprechungen und sogar Familientreffen virtuell abzuhalten, ohne die Möglichkeit zu verlieren, miteinander zu interagieren. Mit den neuesten Entwicklungen bei Virtual-Reality-Headsets und Computergrafiken kann auch ein gewisses Maß an Wohlbefinden und Realismus in der virtuellen Welt erreicht werden, so dass sie sich ähnlich der realen Welt anfühlt.

Das Ziel dieser Arbeit ist es, einen anderen Ansatz als die „traditionellen“ Lösungen wie 3D-Köpfe und Avatare zu finden. Die Idee ist, Methoden zur Erkennung von Gesichtsmerkmalen zu nutzen, um Bilder, ähnlich wie ein Video, in echtzeit zu bewegen und diese in eine Anwendung für eine virtuelle Konferenz zu integrieren. Für diesen Ansatz wurde vid2vid in Kombination mit GANs und Motion Transfer verwendet, um ein einzelnes Bild zu „bewegen“. Der daraus resultierende Prototyp wurde im Hinblick auf das feeling of presence in der virtuellen Umgebung der Anwendung evaluiert. Für die Evaluierung testeten 10 Personen die Anwendung und nahmen anschließend an einer Umfrage teil, die auf der Skala des igroup presence questionnaire basiert, um die Spatial Presence, das Involvement und the sense of realness innerhalb des virtuellen Raums zu messen.

Die Ergebnisse der durchgeführten Experimente deuten auf ein positives und zufriedenstellendes Ergebnis des Prototyps und ein positives Gefühl in Hinblick auf Presence und Involvement hin. So kann am Ende gesagt werden, dass der Prototyp eine vielversprechende Grundlage für die Zukunft darstellt.

Contents

List of Tables

List of Figures

Listings

1. Introduction

1.1. Motivation

Because of the COVID-19 pandemic, the necessity for remote work emerged, fundamentally altering how people interacted with each other in the virtual space in their professional environments. As offices transitioned working from home, the need for effective and realistic communication tools became very important. This shift has paralleled a significant rise in interest and development within the field of virtual reality (VR) and related technologies. The concept of the "Metaverse", is now also gaining traction as tech giants invest heavily in creating immersive virtual worlds. The release of advanced VR devices, such as the Vision Pro, alongside other advanced achievements and inventions, has placed VR into the media spotlight like never before.

Despite these advancements, achieving a high level of realism in virtual interactions remains a significant challenge. Traditional video conferencing solutions, while functional, often fall short in replicating the nuances of face-to-face meetings. The lack of physical presence and the limitations of 2D video feeds can hinder the effectiveness of virtual collaboration, especially in scenarios that require close interaction, such as meetings, conferences, and casual colleague interactions.

Several different methods have been developed to solve these problems. Traditional methods involve using webcam feeds projected onto virtual planes, while more advanced techniques utilize 3D faces or fully rendered 3D avatars. Each of these methods comes with its own advantages and disadvantages. Webcam feeds are straightforward but lack depth and realism. 3D avatars offer a more immersive experience but can be resource-intensive and challenging to animate convincingly.

In response to these challenges, this thesis explores an approach that leverages Generative Adversarial Networks (GANs) [10] and Motion Transfer, like the First Order Motion Model from Siarohin et al. [27]. The focus is on using the First Order Motion Model to generate and animate faces from a single image, creating a more realistic and dynamic representation of individuals in virtual spaces. By packaging this technology into a functional application, the goal is to enhance the realism of virtual interactions with an alternative approach, in the best case providing a more authentic and engaging experience for users. This exploration tries to contribute to the broader field of VR by offering a new solution that could be integrated into

various virtual communication platforms in the future.

The idea of this thesis is to create and test an additional approach to virtual reality conferencing with training a GAN and generate a face for the user. In this scenario the user only needs one source image, for example a portrait of himself, which will then be moved with the help from motion transfer. The resulting images should then be presented in a realistic looking virtual conference room to the other user. The first draft of the idea can be seen in Figure 1.1.

The thesis aims to bring an alternative approach to 3D models, avatars and other traditional approaches. It is intended to serve as a foundation for future applications. It is also intended to test the limits of what is possible with "normal" consumer hardware without having large data centers or multiple graphics cards available in the background.

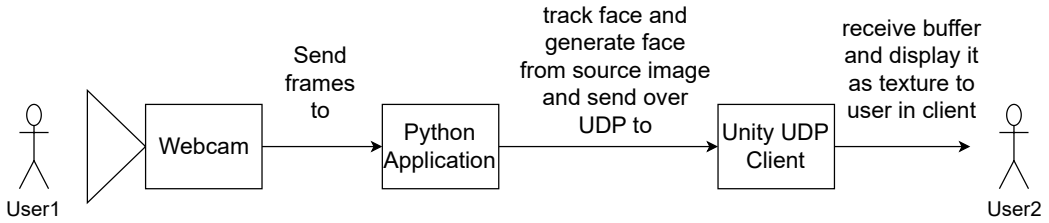


Figure 1.1.: First draft of the system

1.2. Problem Statement and Contribution

The landscape of virtual reality (VR) chat rooms, conferencing systems, and virtual worlds—often referred to as "metaverses"—is vast and varied, yet each existing solution comes with its own set of limitations, downsides, strengths, and weaknesses. A common issue among these platforms is their reliance on 3D avatars that bear little or no resemblance to the real individuals they represent. While some efforts, such as those referenced in [17], strive to replicate the real person using 3D face and head scans, these solutions often fall short of achieving true realism and immersion.

The primary objective of this thesis, along with the development of a virtual reality conferencing application prototype, was to explore and implement state-of-the-art methods, libraries, and frameworks that are good in generating and animating facial images. The goal was to lay a foundation VR avatars created of images by creating a system that not only generates lifelike facial images from photographs but also animates these images in a way that reflects real-time

movements, thereby enhancing the realism and personal connection in virtual meetings.

A secondary objective of the virtual reality conferencing application prototype was to assess how the use of generated and animated avatars impacts the sense of immersion and presence within a virtual reality environment. This focus on presence and immersion is critical, as these elements are key to creating a compelling and effective VR experience. The project faced several significant challenges, including:

- **Identifying methods for generating facial images**: The project required the selection of advanced techniques capable of producing high-quality, realistic images of faces that could be animated and integrated into a VR environment.
- **Developing methods to animate these generated faces**: A crucial aspect of the project was linking the movement of the generated faces to real-time webcam feeds, allowing the avatars to mirror the users' expressions and movements accurately.
- **Designing a realistic 3D environment**: Creating an immersive virtual space that enhances the overall experience was another key challenge, requiring careful attention to detail and design.
- **Enabling network transmission of generated images**: The system needed to efficiently send generated images over a network, maintaining high performance and minimizing latency.
- **Achieving a target of 90 frames per second (fps)**: To ensure a smooth and immersive experience, the application aimed to render at 90 fps, which is critical for maintaining user comfort and preventing motion sickness in VR.

The main contributions of this thesis are:

- **Development of a functional virtual reality conferencing application prototype**: This prototype enables two users to meet and interact within a virtual reality environment, using avatars generated from real images that reflect their facial movements in real-time.
- **Evaluation of presence within the application**: The prototype's effectiveness was assessed through tests involving 10 participants. These users interacted with another individual who used the generated images as their avatar, providing insights into the sense of presence and immersion created by the application.

An complete overview of the implemented features of the VR conferencing application prototype followed with a more detailed description about the mechanics can be found in Section 3.

1.3. Synopsis

This thesis is structured in such a way that it begins with an introductory section that outlines the objectives and context of the work, followed by a comprehensive review of existing solutions in the field. Section 2 Related Work offers a detailed exploration of the state-of-the-art technologies and methodologies relevant to virtual reality applications, face tracking, face generation, and VR interaction. It includes a literature review that examines previous and current research and achievements in these areas.

In Section 3 System Overview the technologies, software, hardware, and algorithms used to achieve the project's goals are presented. This section provides a broad overview of the final application's features, offering insight into the tools and methods employed.

It also shows in more detail the implementation and the the technical details of the project, describing the entire process from facial image generation to the real-time rendering of these images within the Unity engine. It outlines the structure and sequence of operations within the application, providing a step-by-step guide to the development process.

Section 4: Evaluation evaluates the effectiveness of the application using a questionnaire based on the Igroup Presence Questionnaire [42]. This section presents the evaluation methodology, analyzes the questionnaire responses, and discusses the results in detail.

Section 5: Conclusion is the final section and reflects on the outcomes of the project, discussing the results from both the questionnaire and the overall application. It also highlights the challenges encountered during development and suggests potential approaches for future research and improvements.

2. Related Work

There are several publications and papers which are dedicated to virtual reality conferencing with interesting ideas and good looking results. Since the new technologies open up new possibilities for the field, there are several papers like [37], [34], [29], [20] or [24] which dedicate to the improvement of and new possibilities for the experience and a more realistic result.

In the scope of this paper the surveyed papers and articles are dedicated to the general field of Virutal Reality and especially GANs from the fields of video synthesis, 2D-based talking-head synthesis and head pose estimation.

2.1. Analysis

For now let just focus on the overall field of VR Conferencing. There are several papers which go back to the 90s. A more actual work on the field is "Mixing real and virtual conferencing: Lessons learned" from Surendernath et al. [9].

It describes a conference which linked several remote sites via a virtual environment so that the virtual audience could follow the presentations and interact with real presenters. For that purpose they used VirtualOffice, a system designed to enable workplace collaboration. Two presenters were at a physical location. The audience was at different remote locations. The audience sees the whole presentation in a virtual set up room where the VirtualOffice windows were projected to see a live video feed from the physical conference and the presentation files [9].

The conclusion of the paper was that it needed more social and technical management as well as maybe artificial solutions [9]. So even low technical improvements would improve the quality of the virtual reality experience. Also interesting was the reveal that the audio quality was much more important than the quality and fidelity of the virtual reality environment and avatars.

Another interesting approach can be found in Gunkel et al. [24]. This paper also wants to create a photo realistic experience instead of artificial avatars. They use a RGB+depth camera to capture a color depth image of the user. Then depth-based background removal is applied to the image. The resulting image is injected into the web client to transmit it over WebRTC. With this method they achieve a real time 360 degree photo stream of the user as well as of the environment, a picture of the result can be seen in Figure 2.2. It is also a very interesting approach as it

Figure 2.1.: Figure from "Mixing real and virtual conferencing:lessons learned" showing the conference from the audience perspective [9]

achieves in a very easy and fast way a relatively good looking result. But with this approach the system lacks in interaction between the users. Also the face of the other users is not completely visible, only the lower parts can be seen. Because of this, other approaches must be investigated, like face tracking with face estimation and the generating and imitating of the users face.

Another interesting approach for reconstructing the face is Trevithick et al. [37]. The authors present a one-shot method to infer and render a photorealistic 3D representation from a single unposed image (e.g., face portrait) in real-time [37].

The used method directly predicts a canonical triplane representation for volume rendering conditioned on an unposed RGB image. With this single RGB input, their image encoder directly predicts a canonical triplane representation of a neural radiance field for 3D-aware novel view synthesis via volume rendering [37]. To not go further into detail you can see in Figures 2.3 and 2.4 the overview of the used encoder architecture and the training pipeline. The results are very good and can

Figure 2.2.: Figure from "360-Degree Photo-realistic VR Conferencing" showing a conference with 3 users [24]

be seen in Figure 2.5. The right picture is the resulting image. It is a very good method to represent a human face in a 3D space and should be more investigated to leverage it's use in Engines for Virtual Reality conferencing and also be tested how it works in conjunction with a HMD.

A lot has happened in the field of GANs since Goodfellow et al. [10] and there have been many positive and promising developments. For example in the field of video synthesis Aberman et al. [14] present a new video-based performance cloning technique [14]. By training a deep generative network on a reference video that captures the appearance and movements of a specific actor, the method can create new videos where the actor replicates different performances. The training data and the driving performances consist solely of regular video segments, without the need for motion capture or depth data [14].

Chan et al. [22] also present a method in the field of video synthesis, a simple method for "do as I do" motion transfer [22]. Starting with a source video of a person dancing, the method transfers that dance performance to a new target after just a few minutes of the target performing basic movements. This approach treats this task as video-to-video translation, using pose as an intermediate representation. To achieve the motion transfer, Chan et al. extract poses from the source and then apply a learned pose-to-appearance mapping to generate the target's appear-

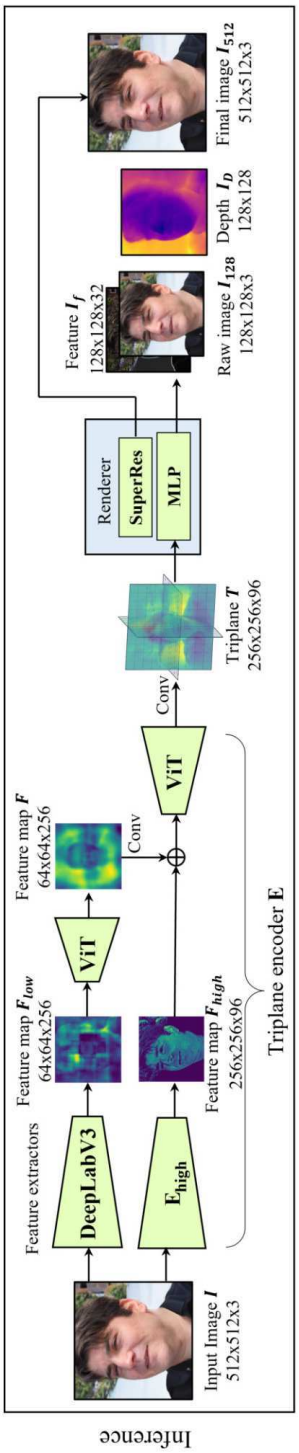


Figure 2.3.: Overview of the encoder architecture and training pipeline of Trevithick et al. [37]

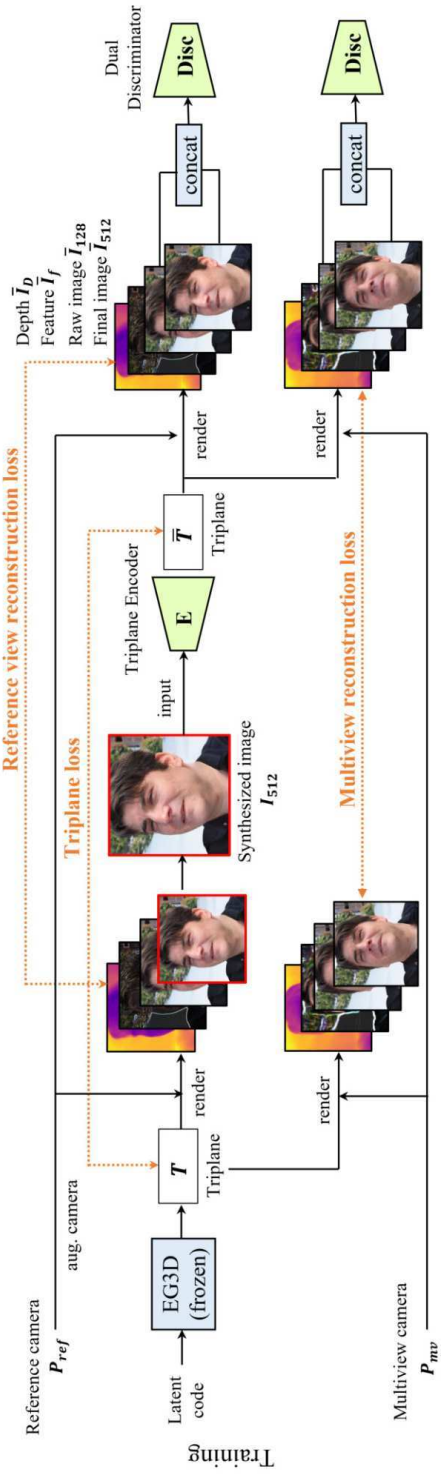


Figure 2.4.: Overview of the encoder architecture and training pipeline of Trevithick et al. [37]

Figure 2.5.: Figure from "Real-Time Radiance Fields for Single-Image Portrait View Synthesis" showing an input image (left) and a pixel-aligned rendering and geometry from a novel view [37]

ance. To ensure temporally coherent video, the method predicts two consecutive frames at a time, and introduces a separate process for synthesizing realistic facial expressions [22]. It create very good results which can be seen in Figure 2.6.

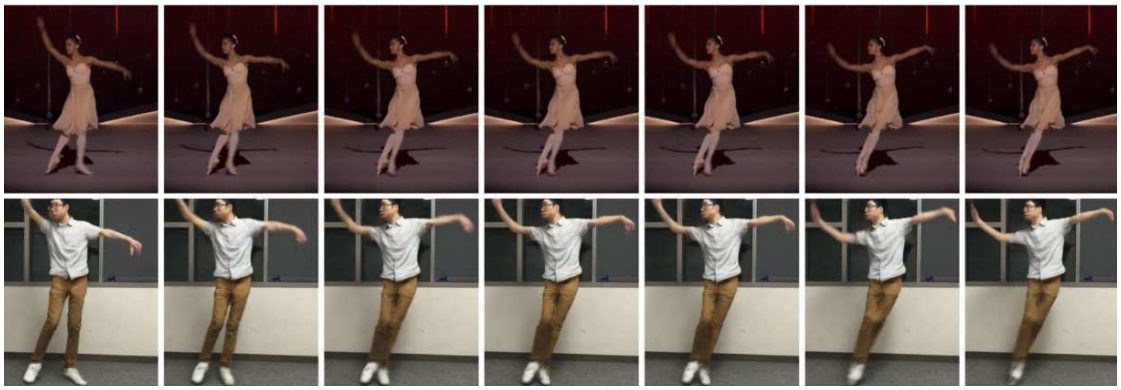


Figure 2.6.: The results of the "Do as I Do" motion transfer from Chan et al. [22]

Kim et al. [16] focuses on the head and face. Kim et al. introduce a method for achieving photo-realistic re-animation of portrait videos using just an input video. At the heart of their approach is a generative neural network. The network processes synthetic renderings of a parametric face model to predict photo-realistic video frames for a specified target actor. Realism in this rendering-to-video conversion is accomplished through careful adversarial training, allowing their method to produce altered target videos that replicate the actions of the synthetic input. To

enable re-animation from a source video to a target video, Kim et al. first generate a synthetic target video using the reconstructed head animation parameters from the source, and then input it into the trained network [16].

In a similar category as [14] and [22] is the work from Liu et al. [25]. Liu et al. introduce a method for creating video-realistic animations of real humans under user control [25]. Unlike traditional human character rendering, their approach does not require a photo-realistic 3D model of the person. Instead, they use a video sequence along with a moderately detailed, controllable 3D template model of the person. Their method involves training a neural network to convert basic synthetic images of a human character into realistic visuals. To train the network, the method first tracks the 3D motion of the person in the video using the template model, then generate a synthetically rendered version of the video. These synthetic images are used to train a conditional generative adversarial network that transforms them into realistic depictions of the person [25].

Zhou et al. [31] present computational methods for transferring body movements from one person to another with any videos collected [31]. For this idea they train a personalized model on a video downloaded from the Internet. This model is able to generate videos of this target person driven by the motions of other people [31]. The model is built on 2 generative networks, a human synthesis net which generates photo-realistic imagery of the target person in a novel pose and a fusion net which combines the generated foreground with the scene, adding shadows or reflections as needed to enhance realism [31].

Siarohin et al. [26] presents a deep learning framework for image animation. Given an input image featuring a target object and a driving video sequence showing a moving object, their framework generates a video where the target object is animated to follow the motion of the driving sequence. This is accomplished through a deep architecture that separates appearance from motion information [26].

Siarohin et al. [27] also proposes a framework which animates a specific object without using any annotation or prior information about the said object. After training on a set of videos depicting objects of the same category, for example faces or human bodies, the method can be applied to any object of this class. To achieve this, Siarohin et al. decouples the appearance and motion information. For complex motions, a representation consisting of a set of learned keypoints along with their local affine transformations is used. A generator network models occlusions arising during target motions and combines the appearance extracted from the source image and the motion derived from the driving video [27]. A result using the vox celeb dataset [13] can be seen in Figure 2.7.

Averbuch-Elor et al. [11] present a method which automatically animates a still portrait, making it possible for the subject in the photo to come to life and express various emotions [11]. Their method is using a driving video (of a different subject)

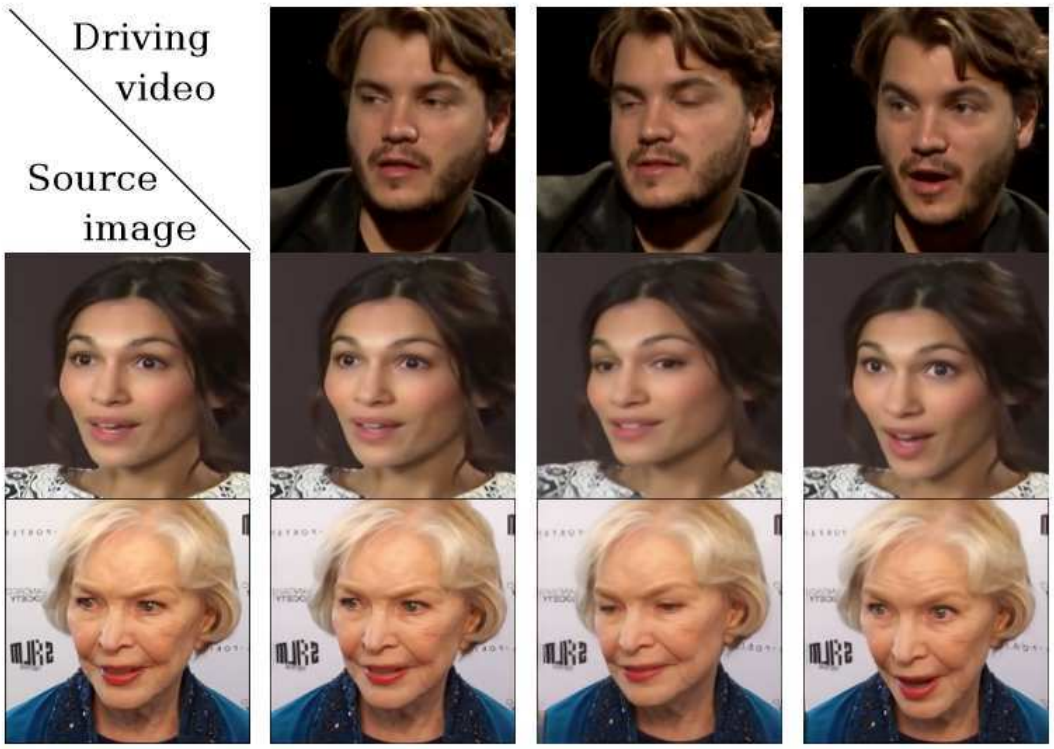


Figure 2.7.: Animation produced with Siarohin et al. method on vox celeb dataset [27]

.

and develop means to transfer the expressiveness of the subject in the driving video to the target portrait. To animate the target image, they use 2D warps that imitate the facial transformations in the driving video. They additionally add fine-scale dynamic details which are. They also try to imitate parts of the target face, such as the inner mouth [11]. It is an very interesting approach and similar to the idea of this work. You can see their impressive results in Figure 2.8.

Ruiz et al. [19] present a method to determine the head pose by training a multi-loss convolutional neural network on 300W-LP, a large synthetically expanded dataset, to predict intrinsic Euler angles (yaw, pitch and roll) directly from image intensities through joint binned pose classification and regression [19]. Estimating the head pose is a very important task for creating and generating fake images for virtual reality conferencing and a very crucial problem. Ruiz et al. argues that the traditional computation of head poses by estimating some keypoints from the target face and solving the 2D to 3D correspondence problem with a mean human

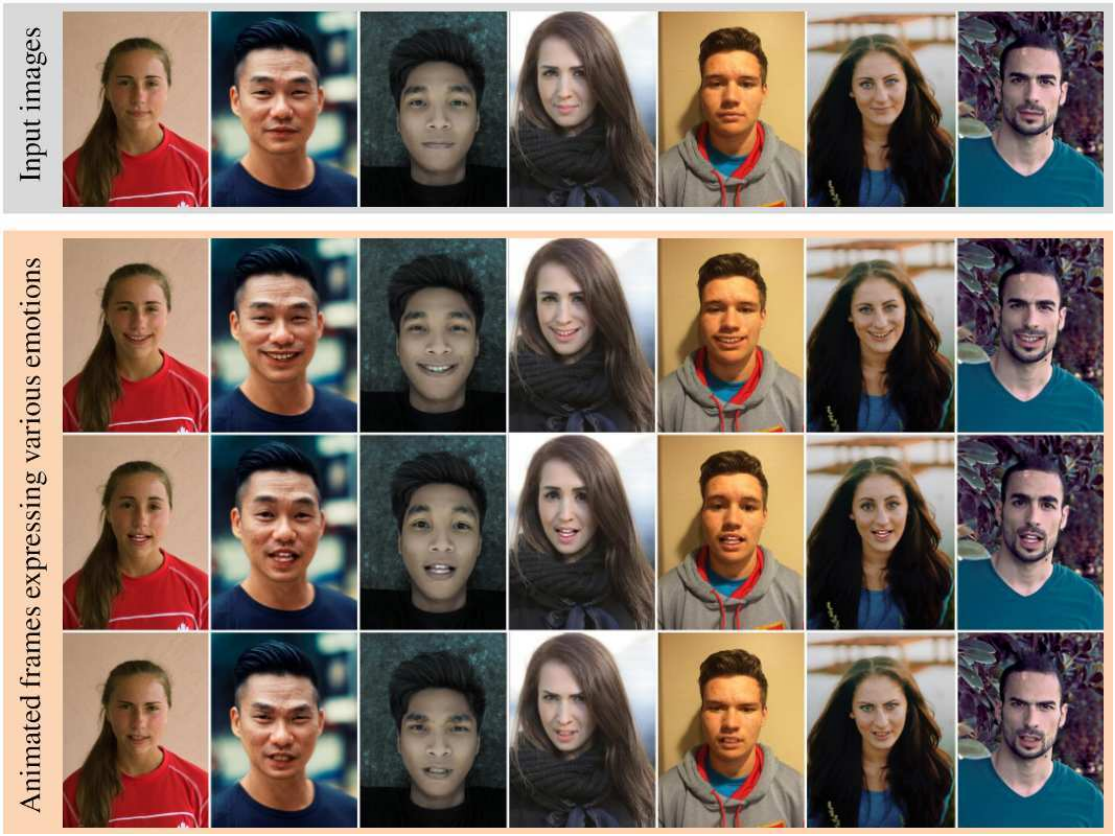


Figure 2.8.: Photo-realistic videos with emotions of Averburch-Elor et al. [11]

head mode, is very fragile [19]. Hence they introduced their method, which is also used for this paper. In Figure 2.9 you can see their network as well as some results in Figure 2.10.

Burkov et al. [32] introduce a neural head reenactment system that uses a latent pose representation to drive the prediction of the RGB image and the corresponding foreground segmentation. The latent pose representation is learned within the reenactment system, relying only on image reconstruction losses. Despite its straightforward approach, with a large and diverse training dataset, this method effectively separates pose from identity. The system can then mimic the expressions of the driving person and even perform reenactment across different individuals [32].

Similar to the previous stated works, Geng et al. [15] present a novel method for real-time portrait animation using just a single photo. Their approach requires only one portrait image and a set of facial landmarks obtained from a driving source, such as a photo or video sequence, to create an animated image with detailed facial expressions. The key innovation in their method is a warp-guided generative model that seamlessly integrates fine facial details, like creases and wrinkles, onto a pre-warped image to produce a high-fidelity facial expression. Their technique separates the nonlinear geometric transformations of facial expressions using lightweight 2D warps, while the synthesis of appearance details is handled by conditional generative neural networks. The also show as a result that this factorization not only improves the network's ability to capture the complex non-linearity of facial expressions but also aids in the design of the network architecture [15].

Most of the previous cited works mostly use single face image as a source, and generate and apply facial animations by merging other person's facial features. Gu et al. [23] point out that certain facial regions, such as eyes or teeth, which may be obscured in the source image, cannot be synthesized accurately or consistently [23]. To address this, they introduce a landmark-driven two-stream network designed to generate more faithful talking facial animations by utilizing multiple source images rather than just one. Their approach includes a network composed of a learning and fetching stream. The fetching sub-network attentively warps and merges facial regions from five source images, each with distinct landmarks, while the learning pipeline renders facial features from the training face space to fill in any gaps. This method ensures that more facial details are preserved and accurately transferred [23]. You can see their network in Figure 2.11.

In the same category but with a slightly different approach, Chung et al. [12] propose a new method with audio. The method accepts two inputs, which are still images of the target face and an audio speech segment, and produces a video of the target face synchronized with the audio [12]. This approach operates in real time and can be applied to faces and audio not encountered during training. To accomplish this, the method utilizes an encoder-decoder Convolutional Neural

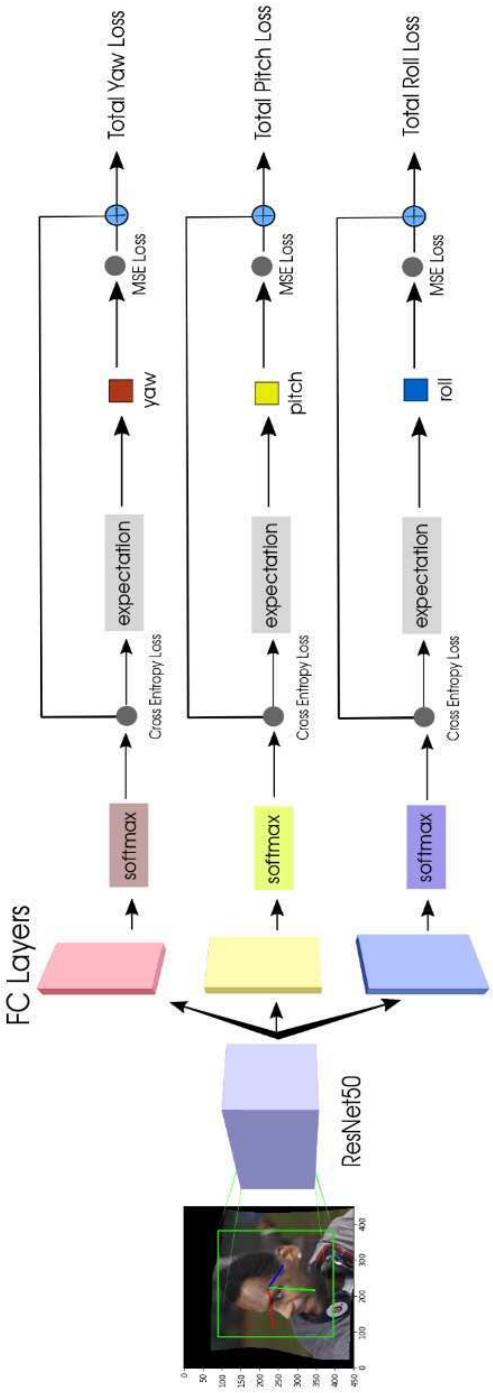


Figure 2.9.: Network architecture of Ruiz et al. with combined Mean Squared Error and Cross Entropy Losses [19].

Figure 2.10.: Example poses detection's from Ruiz et al. [19]

Network model that generates synthesized talking face video frames by leveraging a joint embedding of the face and audio [12].

Song et al. [28] also presents a new method for creating realistic talking face videos using Conditional Recurrent Adversarial Networks in combination with audio. This approach focuses on generating synchronized and natural facial movements based on input audio, with potential applications in video conferencing, virtual avatars, and entertainment [28]. They propose a new video generation network that uses audio input to guide the recurrent adversarial network, ensuring smooth lip and facial movements over time. They also use a multi-task adversarial training approach to enhance the realism and accuracy of lip synchronization. Additionally, they introduce a sample selection method based on phoneme distribution from the audio, which reduces the training dataset size without compromising video quality [28].

Pumarola et al. [18] tries a slightly different approach. They present a new GAN conditioning method that uses Action Units annotations to represent facial movements that define human expressions. This approach allows precise control over the intensity of each facial movement and the combination of multiple actions. They also introduce a fully unsupervised training strategy that only requires images labeled with their active Action Units and uses attention mechanisms to make the

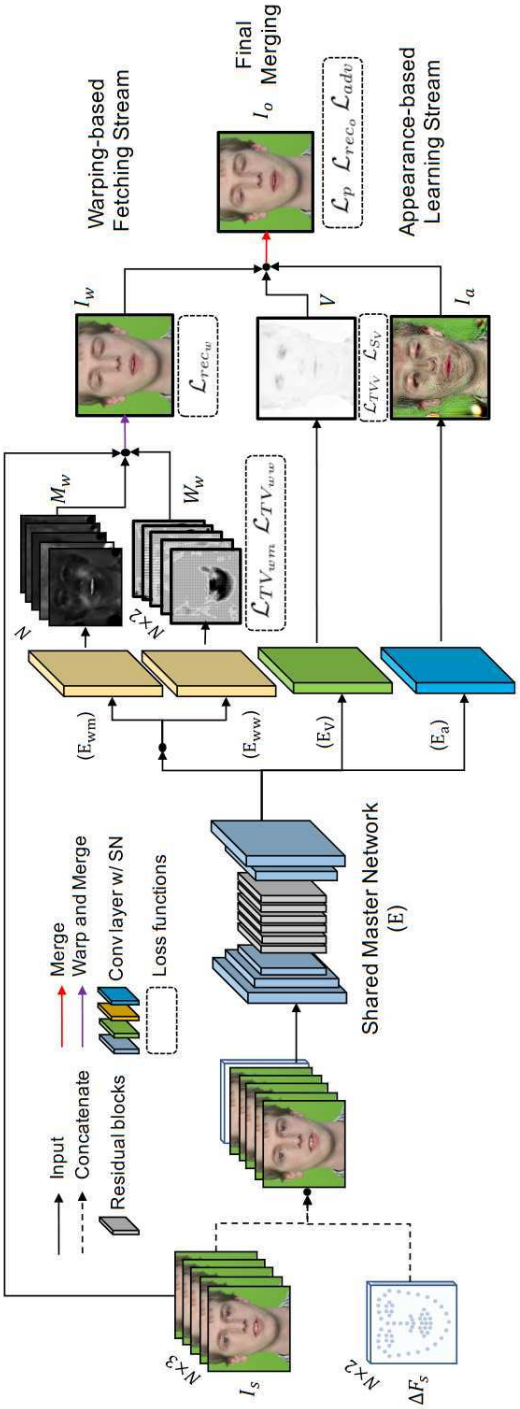


Figure 2.11.: Network structure of Gu et al. [23]

network resilient to changes in backgrounds and lighting [18].

Wiles et al. [21] present a network, X2Face, that can manipulate a source face—taken from one or more frames, using a driving frame to create a new frame that retains the identity of the source but adopts the pose and expression of the face in the driving frame [21]. They also introduce a fully self-supervised method for training the network using a large video dataset. Additionally, they demonstrate that the generation process can be controlled by other inputs, such as audio or pose codes, without requiring additional network training [21].

Zakharov et al., 2019 [30] introduce a system that can create personalized talking head models from just a single image using few-shot learning [30]. The system first undergoes extensive meta-learning on a large video dataset. After this, it can quickly train talking head models for new, unseen people by treating it as an adversarial training problem with powerful generators and discriminators. Importantly, the system customizes the initial settings for both the generator and discriminator based on the specific person, allowing it to train rapidly with only a few images, even though it involves tuning millions of parameters [30].

Zakharov et al., 2020 [33] also propose another neural rendering-based system that creates head avatars from a single photograph [33]. Their method models a person's appearance by breaking it down into two layers: the first is a pose-dependent coarse image generated by a small neural network, and the second is a pose-independent texture image that captures fine details. This texture image is created offline, then warped and combined with the coarse image to achieve high-resolution synthesized head views [33]. You can see a comparison from Zakharov et al., 2020 results to [29], [27] and [30] in Figure 2.12.

Wang et al., 2018 [20] propose a video-to-video synthesis approach under the generative adversarial learning framework [20]. They created generators and discriminators which are coupled with a spatio-temporal adversarial objective. With them they get very good temporally coherent video results on a diverse set of input formats including segmentation masks, sketches, and poses [20]. The next work from Wang et al., 2019 [29] introduces an approach to transfer the facial expression of an actor onto an image of a specific person, called Few-shot Video-to-Video Synthesis (few-shot vid2vid). The few-shot vid2vid framework learns to synthesize videos of previously unseen subjects or scenes by leveraging few example images of the target at test time [29].

The primary goal of video-to-video synthesis (vid2vid) is to transform an input semantic video into a photo-realistic output. However, many existing methods have significant drawbacks, such as requiring large amounts of data from a target subject or scene and having limited generalization capability [29].

The Few-shot Video-to-Video Synthesis framework from Wang et al., 2019 [29] addresses these issues using Generative Adversarial Networks (GANs) with a

Figure 2.12.: Zakharov et al., 2020 [33] results compared to [29], [27] and [30] [33]

network weight generation module that employs an attention mechanism. This allows the model to take a second input of one or more example images from the target domain, in addition to the input semantic video [29].

These images dynamically configure the video synthesis process via the network weight generation mechanism. The module is trained to generate network weights based on the example images. Figure 2.13 shows example outputs based on different input combinations.

Again Wang et al., 2021 [34] proposes a neural talking-head video synthesis model and demonstrate its application to video conferencing. The model learns to synthesize a talking-head video using a source image containing the target person's appearance and a driving video that dictates the motion in the output [34]. The method uses feature extraction on the source image and the driving video. The video synthesis can be seen in Figure 2.14. The keypoints of the source image and the driving video are used to estimate the flows. The results of the warps of the source feature are combined and fed to the motion field estimation network M to

Figure 2.13.: Visualization of face video synthesis results based on different inputs [29]

produce a flow composition mask m. A linear combination of m and wk's then produces the composited flow field w, which is used to warp the 3D source feature. Finally, the generator G converts the warped feature to the output image [34].

Wang et al., 2021 [34] compares the results to other state-of-the-art methods, the results can be seen in Figure 2.15. We will go more into detail of this approach later in Chapter 3.2 since this paper's method is based on this method.

2.2. Summary and Result

All surveyed papers have promising and sometimes very good and realistic results and many different approaches to same or similar problems.

Surendernath et al. [9] explored early attempts at integrating real and virtual environments for conferencing. Their work emphasized the importance of technical and social factors in improving user experience, noting that audio quality often outweighed the need for high-fidelity visual environments.

Gunkel et al. [24] introduced a method for achieving photorealistic VR experiences using RGB and depth cameras. While this approach provided high-quality visual results, it lacked interaction capabilities, limiting its effectiveness in collaborative environments.

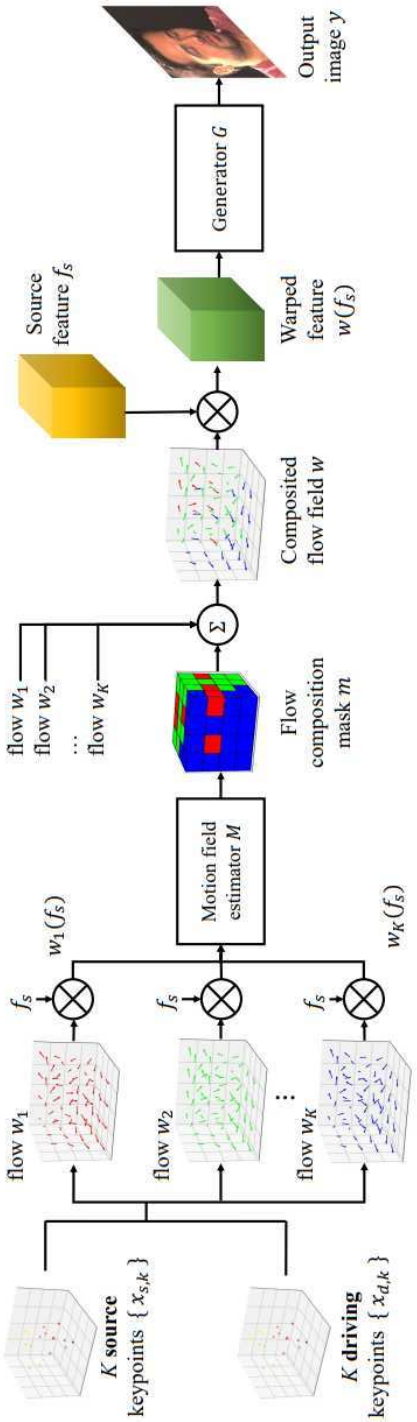


Figure 2.14.: Network structure of Wang et al., 2021 [34]

Figure 2.15.: Comparison of results from Wang et al., 2021 [34]

In the field of Face Generation and Animation Techniques Aberman et al. [14] and Chan et al. [22] developed video synthesis methods that transfer motion from one video to another. These techniques enable the creation of new videos where an actor replicates different performances without the need for motion capture or depth data.

Kim et al. [16] and Liu et al. [25] focused on facial reanimation, generating realistic facial expressions from input videos. Their methods leverage neural networks to convert simple 3D models or synthetic renderings into photorealistic video frames.

Zhou et al. proposed a method to transfer body movements between individuals in videos, while Siarohin et al. [27] developed a framework for animating objects in images based on driving videos, achieving high-quality results in facial animation. Averbuch-Elor et al. [11] presented a method for animating still portraits by transferring expressions from a driving video to the portrait, producing lifelike facial animations. Ruiz et al. [19] introduced a method for head pose estimation, crucial for creating realistic and dynamic facial animations in VR environments.

For Advanced Talking Head Models and Audio-Driven Animation Zakharov et al., 2020 [33] presented a system capable of creating personalized talking head models from a single image using few-shot learning. This approach enables rapid training of neural talking head models for new individuals, with applications in video conferencing and avatar creation. Chung et al. [12] and Song et al. [28] focused on audio-driven facial animation, where facial movements are synchronized with audio input. Their methods ensure realistic lip-syncing and smooth transitions in generated videos. Pumarola et al. [18] introduced a GAN-based method that uses Action Units to control facial expressions, allowing for precise manipulation of

facial movements.

Wang et al., 2021 [34] already tests the whole work in a video conferencing setup and also has promising and good results. That is the reason why it was chosen for the starting point and foundation of this work and to leverage this approach to VR conferencing.

Another decisive factor in the decision was that the results, shown in Figure 2.15, were better compared to [33], [29] and [27]. A combination of both Siarohin et al. [27] and Wang et al., 2021 [34] is used for this work as well as the Fine-Grained Head Pose Estimation without Keypoints from Ruiz et al. [19] which is also adopted from Wang et al., 2021 [34].

The goal of this work is also to create a right away usable VR conferencing prototype with ordinary consumer hardware and inexpensive but powerful enough equipment. AS already stated Wang et al., 2021 [34] builds the starting point for this paper.

3. System Overview

One of the most challenging and time-consuming aspects of this project was the process of identifying and selecting the appropriate software and libraries. The goal was to leverage the latest frameworks and technologies without having to build everything from scratch, which required careful consideration and extensive research. The selection process involved evaluating various options to ensure that the chosen tools would be both compatible with the project's needs and capable of delivering the desired performance. Table 3.1 provides a comprehensive list of the hardware, frameworks, and libraries that were selected for the development of the application. The used software will be discussed in more detail in Section 3.2.

Title	Author	Type	Info
HTC Vive	HTC Corporation	Hardware	Homepage [1]
Unity LTS 2022.3.21f1	Unity Technologies	Engine	Homepage[2]
Python 3.11.5	Python Software Foundation	Software	Homepage[3]
Anaconda 23.9.0	Anaconda Inc.	Software	Homepage[4]
Logitech C920s HD PRO Webcam	Logitech	Hardware	Homepage[5]
Steam VR library for unity	Valve Corporation	Software	Homepage[6]
Playstation VR Headset	Sony	Hardware	Homepage[7]

Table 3.1.: Used Hardware and Software for application

[1]https://www.vive.com
[2]https://unity.com/
[3]https://www.python.org/
[4]https://www.anaconda.com/
[5]https://www.logitech.com/de-at/products/webcams/c920s-pro-hd-webcam.html
[6]https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647
[7]https://www.playstation.com/de-at/ps-vr/

3.1. Hardware

The application was developed with the specific constraints and capabilities of virtual reality (VR) head-mounted displays (HMDs) in mind, particularly focusing on the HTC Vive (https://www.vive.com). However, to ensure broader accessibility, the application was also designed with a fallback option that allows it to be used without an HMD. This flexibility was important, as it enabled the application to reach a wider audience, including those who may not have access to VR hardware.

In terms of performance, achieving a smooth and responsive experience within the VR environment was also very important. A target of 90 frames per second (fps) was set as an optimal benchmark, as this frame rate is essential for maintaining immersion and reducing motion sickness for VR users. To reach this target, it was determined that a mid to high-tier gaming PC or laptop running Windows 10 or 11 would be necessary. Additionally, to handle the computational demands of image generation and manipulation, a Nvidia graphics card with CUDA support was required. To facilitate face tracking, an additional webcam was also required.

For the development and testing phases of the project, a standard/mid gaming PC by today's standards was used. This system, as detailed in Table 3.2, was equipped with Windows 11, an AMD Ryzen 5 5600X 6-Core Processor, a Nvidia RTX 3060Ti graphics card, and 16GB of RAM. This setup met and exceeded the system requirements for the HTC Vive, ensuring that the application could run smoothly and efficiently during testing. The system requirements for the HTC Vive are outlined in Table 3.3.

In addition to the primary development PC, a secondary system was used to test the application under different conditions and to be able to send and receive images as a second recipient. This secondary system was a MacBook Pro M1 2020, which, due to the absence of a Nvidia graphics card, was limited to sending a webcam feed rather than performing the image generation. Despite this limitation, the MacBook Pro played a valuable role in testing the application's fallback capabilities, ensuring that it could function effectively even on less powerful hardware and without a Nvidia graphics card.

Operating System	Windows 11
Graphics card	Asus Nvidia RTX 3060Ti 8GB
Processor	AMD Ryzen 5 5600X 6-Cores
RAM	16GB

Table 3.2.: System for testing and generating faces

Component	Recommended system requirements
Processor	Intel Core i5-4590/AMD FX 8350 equivalent or better
GPU	NVIDIA GeForce GTX 1060, AMD Radeon RX 480 equivalent or better
Memory	4 GB RAM or more
Video output	HDMI 1.4, DisplayPort 1.2 or newer
USB port	1x USB 2.0 or newer
Operating system	Windows 7 SP1, Windows 8.1 or later, Windows 10

Table 3.3.: HTC Vive recommended system requirements [41]

3.2. Feature Overview

The next section provides an overview of the application's features, with a more detailed description available in Section 3.2. One of the most important aspects of the application is a realistic looking environment, specifically designed as an office room for two interacting attendees. The user is placed in an immersive office setting where they are seated in front of a desk. On the opposite side of the desk, the other attendee is represented on a receiving cube.

A very important feature of every conference system is its ability to connect with different users. The application allows users to connect using an IP address and offers the option to either send the webcam feed or the generated images. This functionality is facilitated through an input mask where the user can enter the IP address and select a button to start either the image generation or the webcam feed. The input mask can be seen in Figure 3.1.

Within this virtual office, a cube is positioned to act as the receiver. The receiving cube, located opposite the user behind the desk, serves as the display for incoming images. If no images are being received, the cube remains blank or black. When images are transmitted to the cube over UDP, the application automatically extracts the background from generated images and replaces it with a green screen. This green background is then removed using a chroma key shader, ensuring that the images are seamlessly integrated into the virtual environment.

Upon starting the application, the user is greeted by an input mask that offers two options: the HMD button or the Webcam button. If the user selects the HMD button, a Python script is triggered in the background. This script uses OpenCV

to capture the webcam feed, track the user's face, and generate images. The script also extracts the background of these images, replacing it with a green screen before sending the frames over UDP to the specified IP address.

If the user opts to use the webcam directly, another Python script is executed in the background. This script uses OpenCV to capture the webcam feed and then transmits the video stream over UDP to the specified IP address.

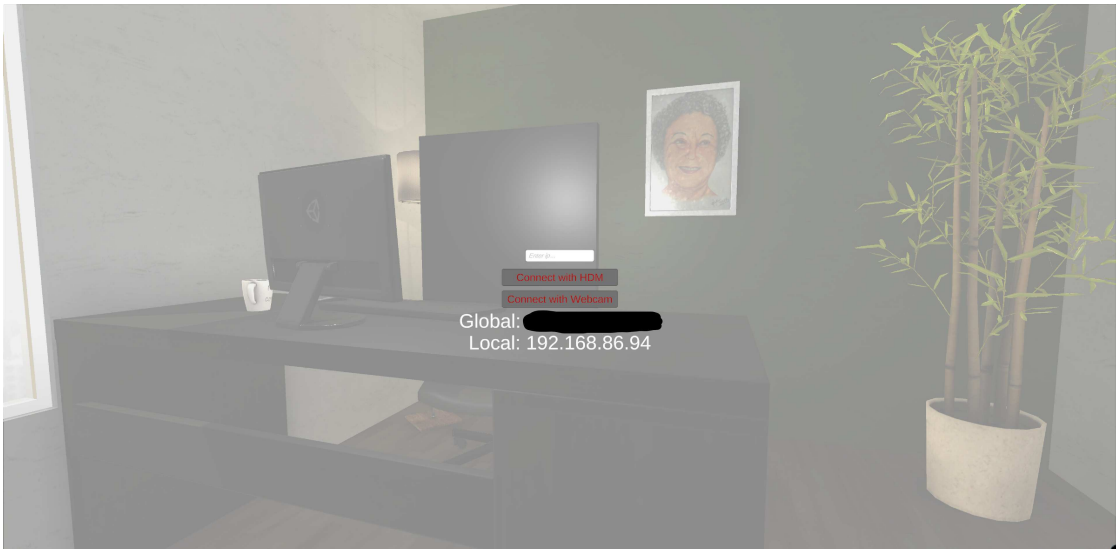


Figure 3.1.: Welcome screen of the Unity application

After extensive research and evaluation of various software and libraries, as discussed in Section 2, it was determined that the combination of Python, Anaconda, and several specialized tools would be the most effective approach for the project. Specifically, the work from Wang et al. [34], the Pytorch implementation known as "FaceAnimation" [50], and the GitHub repository "Face Animation in Real Time" [36], in conjunction with the Unity engine, were selected as the foundation for the development of the application.

Initially, there was consideration given to generating personalized facial animations using a Generative Adversarial Network (GAN). However, this idea was ultimately set aside due to the significant hardware limitations that would have made such an approach impractical. Instead, the vox celeb dataset [13] was chosen as a more feasible alternative, offering a robust source of data for the facial animation process.

For the implementation of the project, Python 3.11 was utilized in combination with Anaconda 23.9. These tools provided a flexible and powerful environment for managing the various dependencies and libraries required for the project. To tailor

the existing resources to the specific needs of the thesis, several modifications were made to the original "Face Animation in Real Time" implementation [36]. These adjustments were necessary for enabling effective face tracking and facilitating the transmission of data via UDP to the Unity application.

The core of the implementation revolves around a Python script designed to capture individual frames from a webcam feed. These frames are then processed through multiple machine learning models responsible for detecting, estimating, and extracting human faces. Following this, a generator is employed to create synthetic imagery based on the target input image provided by the user. Before the final image is sent to Unity, the background is replaced with a plain green color, a process that simplifies subsequent chroma keying in Unity. The image is then transmitted to Unity using Python sockets over UDP.

On the Unity side, a C# script is employed to handle the incoming data. This script receives the buffered image, stores it, and converts it in real-time into a texture. This texture is then applied to a 3D Cube object within the Unity environment. The Cube utilizes a Chroma Key shader, which effectively removes the green background, allowing the generated face to be seamlessly integrated into the virtual scene.

Figure 3.2 illustrates this workflow, offering a high-level overview of the face generation process. The Figure illustrates the interaction between the various components, from initial image capture through to the final images which will be send to the Unity application. Each of these components has been described in greater detail in Section 3.4.

In summary, the combination of Python/Anaconda, machine learning models and Unity, along with the integration and combination of these tools through scripting and data handling processes and image processing has resulted in a good functioning system for real-time facial animation on images. This system effectively leverages existing technologies while addressing the practical constraints of hardware and performance, providing a solid foundation for further development and refinement.

3.3. Third Party Software

To achieve the goals described in Section 1.2, various Third Party tools as well as the work from [34] in combination with [36] have been utilized and modified. The reason behind those choices was to reduce the workload and complexity of the given task by using libraries and implementations that have been proven to be useful and efficient at solving some of the problems which occurred during development. The following third party software was used for this thesis and will be briefly described in the upcoming subsections:

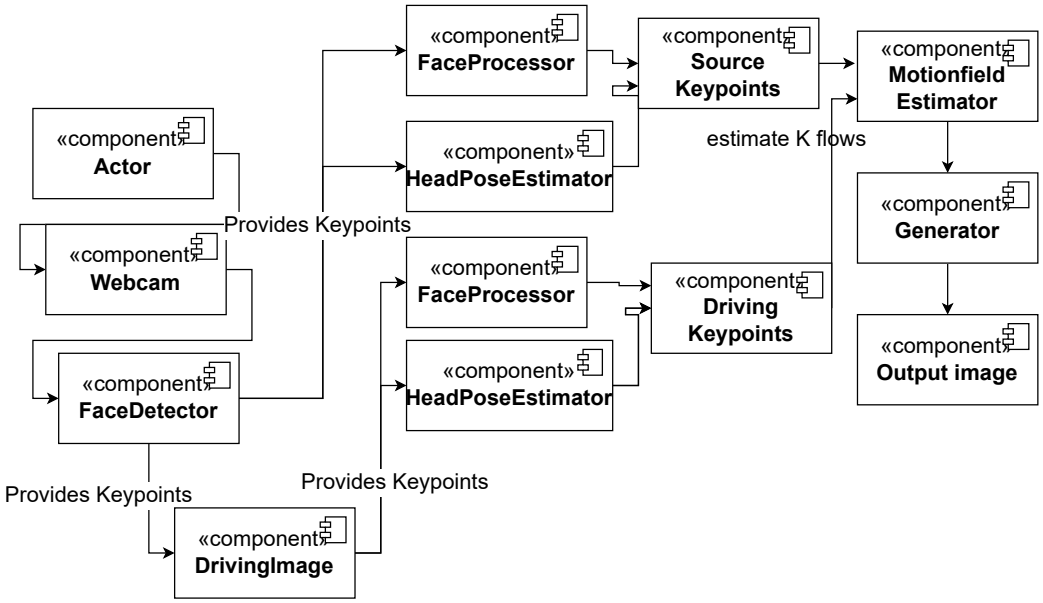


Figure 3.2.: Overview of the Face Generation

- OpenCV 4.9.0 [8]
- PyTorch 2.2.0 [9]
- NVIDIA CUDA Toolkit 11.8 [10]
- NVIDIA cuDNN 8.7.0 [11]
- batch-face 1.4.0 [12]
- Unity 2023LTS [13]
- Chromakey shader uChromaKey [14]

[8] https://opencv.org/
[9] https://pytorch.org/
[10] https://developer.nvidia.com/cuda-toolkit
[11] https://developer.nvidia.com/cudnn
[12] https://github.com/elliottzheng/batch-face/tree/master
[13] https://unity.com/products/unity-engine
[14] https://github.com/hecomi/uChromaKey

3.3.1. OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. It comes with Python, c++, Java and MATLAB interfaces and supports all major operating systems: Windows, Linux, Android and Mac OS. It s written natively in C++ and has a templated interface that works seamlessly with STL containers [45].

PyTorch

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs [46]. It is a fully featured framework for building deep learning models which are used in applications like image recognition and language processing [47]. It is written in Python.

NVIDIA CUDA Toolkit

The CUDA Toolkit provides a development environment for creating high-performance, GPU-accelerated applications. The toolkit features GPU-accelerated libraries, debugging and optimization tools, a C/C++ compiler, and a runtime library [43]. One of the main features is its API which allows software to use GPUs manufactured by NVIDIA for general purpose processing, also known as general-purpose computing on GPUs (GPGPU), by giving access to the GPU's virtual instruction set and parallel computational elements [8].

NVIDIA cuDNN

The NVIDIA CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks [44]. It's key features are accelerated learning, Expressive Op Graph API and fusion support. Deep learning neural networks are used in computer vision, conversational AI and other systems. NVIDIA's GPU-accelerated deep learning frameworks speed up the time needed for training for these systems and technologies, reducing several-day sessions to just a few hours [44].

batch-face

batch-face provides out-of-box face detection and face alignment with batch input support and enables real-time application on CPU. It features Batch input support

for faster data processing, a smart API, ultrafast with inference runtime acceleration, automatic download of pre-trained weights and minimal dependencies [38].

Unity2023LTS

Unity's industry-leading engine provides tools to create and operate amazing games and other real-time interactive experiences and publish them to a wide range of devices [48]. For VR experiences it offers the XR Plug-in Management system to install and enable XR provider plug-ins for the devices you want to support [49].

Chroma key shader

The uChromaKey shader for the Unity Engine is a simple shader asset to use a chroma key effect in Unity [40].

3.4. Implementation

The core component of the application lies in the generation of images and the application of movement to the source image. This process is pivotal to the overall functionality and user experience of the application, as it directly influences the realism and fluidity of the visual output. Achieving this required a series of complex steps, which were implemented using Python, leveraging the libraries previously mentioned, and building upon the methodologies outlined in [34].

To provide a clearer understanding of this intricate process, the following sections will delve into the most crucial steps involved. Each step will be described in detail, outlining the sequence of operations that transform the source image into the final, dynamically animated image that is displayed within the application.

The process begins with the capturing of the webcam feed. The frames are first processed to ensure they are in a suitable format for the application of movement and other transformations. Following this, the images are passed through several machine learning models, each designed to handle different aspects of the animation process—such as facial detection, feature estimation, and motion mapping. These models work together to finally apply the animation on the static source image and generate a corresponding animated version that accurately reflects the intended movements and expressions.

Once the preliminary processing and animation steps are completed, the generated image undergoes further refinement. This includes adjustments to the image's background and overall composition, ensuring that it seamlessly integrates into the virtual environment when displayed. The final stage involves sending the processed image to the Unity application, where it is rendered in real-time. The image is displayed within the application's virtual space, where it is further enhanced by

the application of chroma keying and other visual effects to create a compelling and immersive user experience.

The detailed breakdown of these steps, from the initial source image to the final display within the application, should help to better understand the technical complexities that make this application both functional and effective. The following sections will provide a step-by-step explanation of each component.

3.4.1. Vid2vid

In order to better understand the method used, the concept of vid2vid should first be briefly explained. Vid2Vid [20] is a deep learning model for video-to-video translation developed by researchers at NVIDIA (Wang et al. [20]). The model is designed to generate high-quality videos from input videos with different visual styles or properties. It uses CGAN framework to learn the mapping between two different visual domains, such as from day to night or from a sketch to a realistic image [35].

The Vid2Vid model has two main components: an encoder-decoder and discriminator networks. The encoder-decoder network generates the output video, while the discriminator network distinguishes between real and fake videos [35]. The architecture for low res videos can be seen in Figure 3.3.The encoder-decoder network is trained on the input video and a target video, and it generates the output video frame by frame. The discriminator network delivers a response to the generator network on how realistic the generated videos are [35].

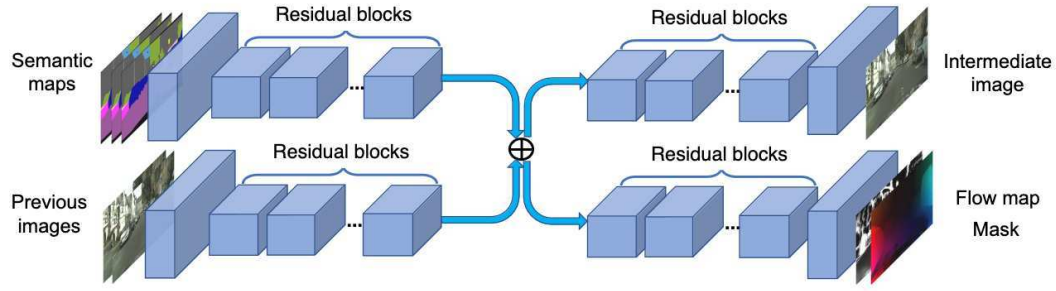


Figure 3.3.: Architecture of vid2vid for low res videos [20]

3.4.2. Python application: Generating images

The Python application starts with the generation of fake images by using advanced techniques from recent research. Specifically, the motion transfer model described

in the paper [27], the One-Shot Free-View Neural Talking-Head Synthesis method from the paper [34], and the VoxCeleb dataset v1 [13] were all used in the image generation process.

Initially, there was a consideration to train a Generative Adversarial Network (GAN) with a set of personal facial images and utilize the resulting trained weights. However, this approach was abandoned due to several significant challenges. One major issue was that more than 10 or 20 images were necessary otherwise the results were insufficient for achieving high-quality results. Furthermore, the training process was very time-consuming, with each session potentially taking up to 12 hours or more depending on the selected parameters. These factors made the GAN-based approach impractical for the scope of this project.

Another challenge encountered during the development was related to face detection and facial landmark recognition. Most existing frameworks and libraries for these tasks require clear visibility of the eyes to accurately identify an "object" as a face. This requirement lead to a new problem: When users were wearing a head-mounted display (HMD), which obscured a significant portion of the face, including the eyes. To address this issue, a creative solution was implemented: a drawing of half a face was attached to the HMD. This drawing provided the necessary visual cues for the libraries to detect facial landmarks more reliably, thereby facilitating the accurate generation and animation of the user's face. Figure 3.4 illustrates the drawing of the half-face that was used in this process.

This workaround proved effective in overcoming the limitations of the face detection algorithms, ensuring that the application could function smoothly even when users were fully immersed in the virtual environment with an HMD. This combination of advanced techniques and innovative problem-solving was crucial in achieving the project's goals, allowing for the creation of a realistic and responsive virtual conferencing experience.

3.4.3. Process first frame

The application starts by capturing the webcam feed, which serves as the input for further processing. The first step is to determine whether the current frame being processed is the very first frame captured or if it is one of the subsequent frames. This distinction is important because the application handles the first frame differently from the others.

If the application identifies that the current frame is the initial one, it triggers the function called *Process first frame*.

Within the *Process first frame* function, the application works to detect the presence of a face in the frame. Once a face is detected, the function moves on to identifying keypoints on the face. These keypoints are specific landmarks, like the corners of the eyes, the tip of the nose, and other significant facial features. The

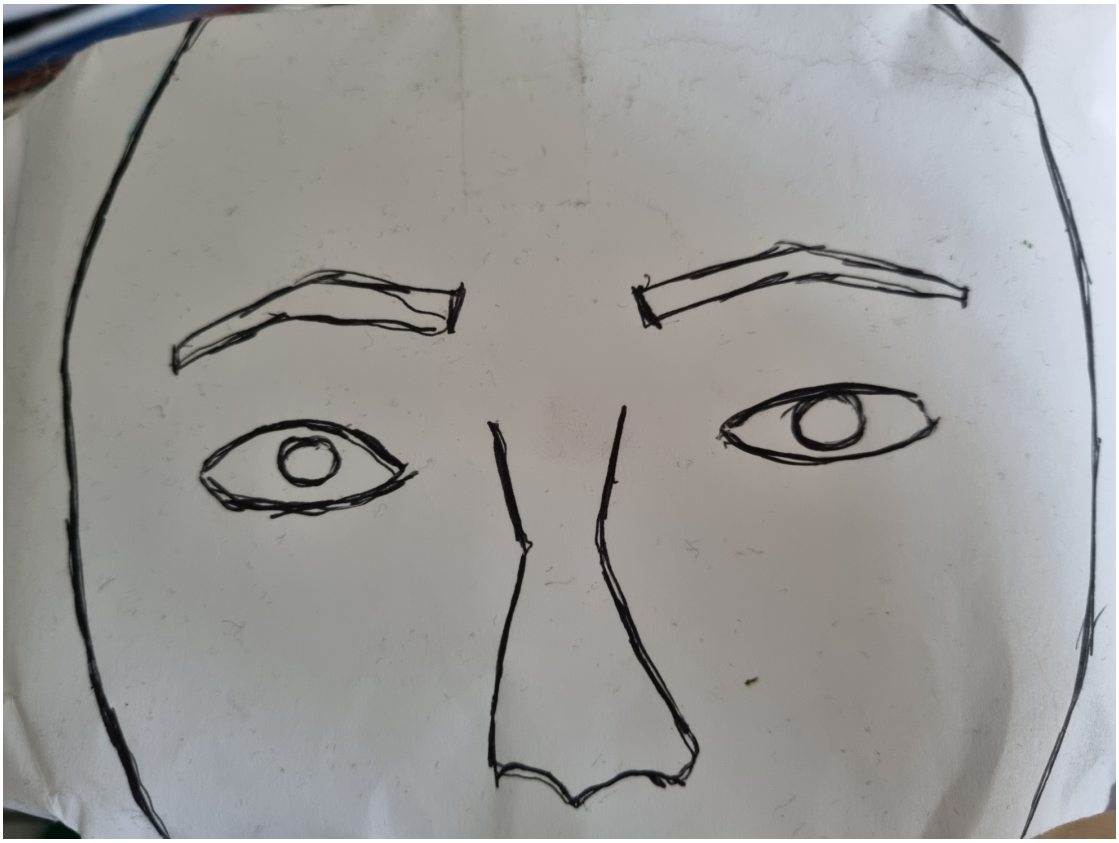


Figure 3.4.: Drawing of half a face used on the VR Headset

application uses a keypoint detector, which is based on a method from reference [34], to accurately locate these keypoints.

In addition to detecting the keypoints, the function also calculates the head pose. This involves determining the orientation of the head, such as whether it is tilted to the side, looking up or down, or facing straight ahead. The keypoint detector not only identifies the positions of these keypoints but also provides additional information, known as the Jacobian, near each keypoint. This data is essential for understanding how the face moves and changes expression.

Finally, the function estimates the overall facial expression by analyzing the detected keypoints and the head pose. This resulting data serves as the foundation for processing the subsequent frames.

You can see the implementation of the processing of the first frame and the described flow in 3.1. After the first frame is processed it goes over to the general frame processing, which is described in the next section.

```
def _process_first_frame(self, frame):
        print("Processing first frame")
        # function to process the first frame
        faces = self.face_detector(frame, cv=True)
        if len(faces) == 0:
            raise ValueError("Face is not detected")
        else:
            self.coords = faces[0][0]
        face = get_square_face(self.coords, frame)
        self.last_coords = self.coords

        # get the keypoint and headpose from the source image
        with torch.no_grad():
            self.kp_canonical = self.kp_detector(self.source)
            self.he_source = self.he_estimator(self.source)

            face_input = self._conver_input_frame(face)
            he_driving_initial = self.he_estimator(face_input)
            self.kp_driving_initial, coordinates =
   keypoint_transformation(self.kp_canonical, he_driving_initial,
   output_coord=True)
            self.kp_source = keypoint_transformation(
                self.kp_canonical, self.he_source, free_view=True,
    yaw=coordinates["yaw"], pitch=coordinates["pitch"], roll=
   coordinates["roll"]
```

Listing 3.1: Processing of First frame [36]

3.4.4. Processing Frames

In Listing 3.2, the process begins with checking whether the current frame is the first one being processed. This check is performed in lines 2 to 9. If the current frame is not the first one, the application then proceeds to the next step, which involves detecting a face within the frame, as indicated in lines 10 to 16.

When a face is detected, the application uses the previously calculated coordinates and keypoints. These keypoints are essential for accurately tracking the face's position and movement. The code converts these keypoints into tensors, which are a form of structured data that is efficient for processing in machine learning models.

Following this, the application recalculates the keypoints and head pose for the current frame, as seen in lines 21 and 22. This step ensures that the application accurately captures any changes in the face's orientation or expression. After recalculating, the results are normalized in line 23, which adjusts the values to a common scale, making them easier to work with in the next steps.

These processed and normalized values are then passed to the generator. The

generator is a key component of the system, responsible for creating the final image. The details of how the generator operates are further explained in [34]. In brief, the generator takes the warped features, projects them into 2D space, and multiplies them with an occlusion mask from the motion field estimator.

The generator then applies a series of 2D residual blocks and upsampling layers to refine the image further. Residual blocks help retain important features while minimizing information loss, and upsampling layers increase the resolution, making the image sharper and more detailed.

Finally, the generator produces the final image, which is then used as part of the application's output. Figure 2.14 illustrates this entire process.

```
def _inference(self, frame):
        # function to process the rest frames
        with torch.no_grad():
            self.n_frame += 1
            if self.first_frame:
                self._process_first_frame(frame)
                self.first_frame = False
            else:
                pass
            if self.n_frame % self.detect_interval == 0:
                faces = self.face_detector(frame, cv=True)
                if len(faces) == 0:
                    raise ValueError("Face is not detected")
                else:
                    self.coords = faces[0][0]
                    self.coords = smooth_coord(self.last_coords,
   self.coords, self.smooth_factor)
            face = get_square_face(self.coords, frame)
            self.last_coords = self.coords
            face_input = self._conver_input_frame(face)

            he_driving = self.he_estimator(face_input)
            kp_driving = keypoint_transformation(self.kp_canonical
   , he_driving)
            kp_norm = normalize_kp(
                kp_source=self.kp_source,
                kp_driving=kp_driving,
                kp_driving_initial=self.kp_driving_initial,
                use_relative_movement=True,
                adapt_movement_scale=True,
            )

            out = self.generator(self.source, kp_source=self.
   kp_source, kp_driving=kp_norm, fp16=True)
            image = np.transpose(out["prediction"].data.cpu().
   numpy(), [0, 2, 3, 1])[0]
            image = (np.array(image).astype(np.float32) * 255).
```

```
    astype(np.uint8)
34              result = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
35
36          return face, result
```

Listing 3.2: Processing of all frames [36]

To remove the background and isolate the face, the application uses the Python library CVZone [39]. Specifically, it employs the SelfiSegmentation module from this library. This segmentor is designed to effectively separate the face from the background, offering two modes for operation: mode 0 for normal use and mode 1 for landscape orientation, which operates more quickly.

Once the background is removed, the next step is to send the processed frames over the network. For this purpose, the application uses Python's standard network library and its socket functions. Given the need for rapid transmission of data, the User Datagram Protocol (UDP) was chosen as the communication protocol. UDP is well-suited for this task because it allows packets to be sent quickly without the overhead of ensuring delivery, which is ideal for real-time applications like this one.

3.4.5. Unity Application

The Unity application is the heart of the conferencing system. It is responsible for receiving the generated frames and displaying them in a realistic virtual environment. This application not only projects the frames for the user but also offers the option to generate frames or simply send the webcam feed directly to the receiver.

When you start the Unity application for the first time, you have the option to choose whether you want to generate frames using a CUDA-capable graphics card or just send the webcam feed. You will need to enter the IP address of the receiver, and the Python application will then attempt to connect to this IP. Meanwhile, the Client in the Unity application waits for this connection to be established.

Using the capabilities of the Unity Engine, a virtual reality meeting room was created. This meeting room provides a consistent experience for both users. The receiver is always positioned in front of a desk, looking at the sender, who is seated behind a desk with a computer. The layout of the office room and the receiver's point of view are illustrated in Figures 3.5 and 3.6.

3.4.6. Receiving Buffer

To receive the generated and processed frames from the Python application, a UDP Client needs to be set up within the Unity Client. This is done by creating a new thread specifically for handling this task, as shown in line 22 of the code in Listing 3.3. The application utilizes the UDP Client class from the System.Net.Sockets library,

Figure 3.5.: Photorealistic meeting room in the Unity Application

which continuously listens for incoming data while the application is running. When the data is received by the ReceiveData function, detailed in line 43, the application captures and stores the incoming data in a byte array (byte[]), ensuring it can be processed and used within the Unity environment.

```
1  public class ReceiveImages : MonoBehaviour
2  {
3  Thread thread;
4      public int connectionPort = 25002;
5      private UdpClient udpclient;
6      bool running;
7      private Texture2D mainTexture;
8      private byte[] recvBuffer;
9      private int bytesReceived;
10     private int maxMessageSize;
11     private byte[] lengthBuffer;
12     public GameObject renderImage;
13     private IPEndPoint anyIP;
14
15
16     void Start()
17     {
18         Application.targetFrameRate = 40;
19         // Receive on a separate thread so Unity doesn't freeze
       waiting for data
20         ThreadStart ts = new ThreadStart(GetData);
21         this.lengthBuffer = new byte[sizeof(int)];
```

```
22          this.maxMessageSize = 0;
23          thread = new Thread(ts);
24          thread.Start();
25          mainTexture = new Texture2D(2, 2);
26      }
27
28      void GetData()
29      {
30
31          // Create a client to get the data stream
32          udpclient = new UdpClient(connectionPort);
33          // Start listening
34          running = true;
35          while (running)
36          {
37              ReceiveData();
38          }
39      }
40
41      private  void ReceiveData()
42      {
43          Debug.Log("Receiving UDP...");
44          while (true)
45          {
46
47              try
48              {
49                  // Bytes empfangen.
50                  anyIP = new IPEndPoint(IPAddress.Any, 0);
51                  byte[] data = udpclient.Receive(ref anyIP);
52                  if (data != null)
53                  {
54                      Debug.Log("Received data");
55                  }
56                  recvBuffer = data;
57
58              }
59              catch (Exception err)
60              {
61                  print(err.ToString());
62              }
63          }
64      }
65
66
67      // Use-case specific function, need to re-write this to
    interpret whatever data is being sent
68      public static void ParseData(byte[] textureData, Texture2D
    destination)
```

```
69      {
70          destination.LoadImage(textureData);
71      }
72
73      void Update()
74      {
75          ParseData(recvBuffer, mainTexture);
76          renderImage.GetComponent<Renderer>().material.mainTexture
    = mainTexture;
77
78      }
79
80
81      void OnApplicationQuit()
82      {
83          thread.Abort();
84      }
85  }
```

Listing 3.3: ReceiveImages class

Figure 3.6.: Point of view of the receiver in the Client

3.4.7. Displaying Buffer as Texture

In every frame, the Update function is called, and within it, the ParseData function is executed (as shown in line 75 of Listing 3.3). This ParseData function takes the

data from the buffer and the texture of the game object, updating the texture by converting the buffer into a Texture2D each time the function is invoked. Although this method is resource-intensive, it is necessary to ensure that the texture is updated accurately and promptly.

As the final step in this process, the Unity plugin uChromaKey is used to remove the green background from the texture. This plugin is a simple shader asset that applies a chroma key effect in Unity, effectively cutting out the green screen and leaving only the desired content visible [40]. The outcome of this process, along with a view of the image generator in Python, is illustrated in Figure 3.7. Figure 3.8 illustrates more results with different avatars.

Figure 3.7.: Receiving the images in the Unity Application

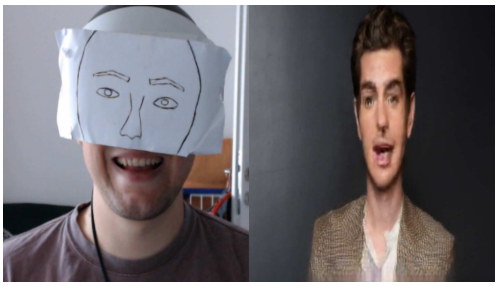
3.5. Summary

In summary, there are two applications running concurrently to facilitate the system's functionality. The first application is the Unity application, which plays a central role by launching the Python application in the background.

The Python application has two primary tasks: it can either directly send the raw webcam feed or generate images based on the feed. Whichever task it performs, the resulting data—whether it be the webcam feed or the generated images—is transmitted over UDP to another instance of the Unity application at the specified IP address.

(a) Anne Hathaway

(b) Anne Hathaway



(c) Andrew Garfield

(d) Mona Lisa

(e) Mona Lisa

Figure 3.8.: More results with different Avatars

The Unity application serves a dual purpose, acting as both a transmitter and a receiver. As the transmitter, it sends out the data processed by the Python application. When functioning as the receiver, the Unity application captures the incoming data—whether it's the generated images or the live webcam feed—and stores it in a buffer. The application then converts this buffered data into a texture, which is subsequently rendered and displayed within the virtual environment. This process ensures that the virtual space is populated with the appropriate visual content, creating a seamless and interactive user experience.

The application performs good, maintaining a good frame rate and demonstrating stability throughout its operation. Figure 3.9 again shows the frames per second of the running application and more real-time data on the graphics card. In the following section, the application will undergo an evaluation across various categories related to presence within the virtual space. This evaluation will involve testing by 10 participants to assess the application's effectiveness and user experience in different aspects of virtual presence in virtual reality. You can watch a demonstration video through the Link in Appendix Figure A.2.

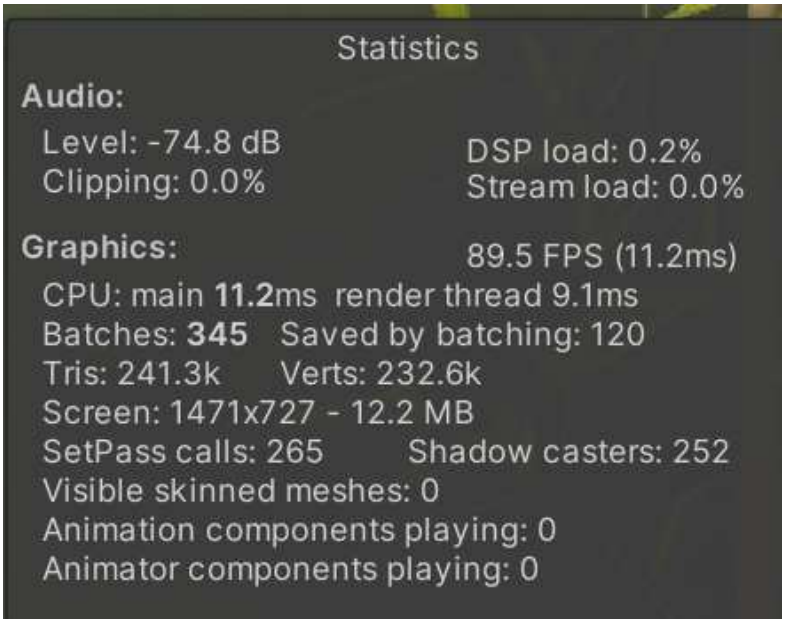


Figure 3.9.: Frames per Second of the running Application

4. Evaluation and Discussion

To evaluate the application in terms of the users' sense of presence within the virtual environment (VE), a questionnaire was utilized with questions from the Igroup Presence Questionnaire (IPQ) [42]. The IPQ is a well-established tool designed to measure the sense of presence that individuals experience while interacting with a virtual environment. This scale was developed through rigorous research, involving a large pool of items and conducted over two survey waves to ensure its validity and reliability.

The current version of the IPQ is organized into three distinct subscales, each of which emerged from principal component analyses. Additionally, there is one general item that does not belong to any specific subscale [42]. The three subscales are:

- Spatial Presence (SP) - This subscale measures the extent to which users feel physically present within the virtual environment, as though they are actually part of the virtual world rather than merely observing it.

- Involvement (INV) - This subscale gauges the level of attention users devote to the virtual environment and the degree of emotional and cognitive involvement they experience while interacting with it.

- Experienced Realism (REAL) - This subscale assesses how realistic the virtual environment feels to users, focusing on their subjective experience of the environment's authenticity.

The additional general item included in the questionnaire is designed to assess the overarching "sense of being there" within the virtual environment. This item is particularly important as it has high correlations with all three subscales.

For the purposes of this study, the questionnaire consisted of 10 carefully selected questions. Participants were asked to respond to each question on a scale from 1 to 5, with 1 indicating "totally disagree" and 5 indicating "totally agree". This format allowed for a nuanced understanding of how users perceived their sense of presence, involvement, and realism within the virtual environment.

Further details on the development and theoretical foundation of the Igroup Presence Questionnaire can be found in the following references: [7], [2], [4], [6] and

[5]. These sources provide additional context and validation for the use of the IPQ in virtual reality research, underscoring its relevance and applicability to our study.

The igroup presence questionnaire was originally only tested in its German version and the igroup.org – project consortium warns of the possibility that some terms in English might be lost in translation. Because the testers have different backgrounds, the questionnaire was conducted in English. Table 4.1 shows the questions from the questionnaire as well as their sources.

4.1. Test Setup

The test users were instructed to sit comfortably in an armchair and to equip the head-mounted display (HMD) on their heads. Before the test began, each user was provided with a brief explanation outlining what they should expect during the test, how to handle any potential issues such as motion sickness, and the importance of approaching the test with a calm and impartial mindset. This introduction was designed to ensure that the users felt comfortable and prepared, allowing them to fully engage with the virtual reality experience without undue concern.

The first task assigned to the users was to simply look around, explore their virtual surroundings, and acclimate themselves to wearing the HMD. This initial period was crucial for helping the users become familiar with the immersive environment and the sensation of being within a virtual space. During this time, the application was already actively receiving generated images from another user who was also participating in the test by wearing an HMD. However, this second user was only receiving the webcam feed and did not interact with the test environment in the same way as the primary users.

After the initial exploration phase, the test users were instructed to focus on a specific task: looking at the face of a virtual representation of Leonardo DiCaprio. They were then asked to briefly interact with this avatar in a short conversation. This interaction was designed to test the application's ability to facilitate natural, immersive communication between users within the virtual environment.

Upon the completion of this interaction, the test was concluded. The users who had received the generated images were then asked to complete the questionnaire, providing feedback on their experience.

A total of 10 individuals participated in this test, and their responses were carefully analyzed to assess the effectiveness of the application. The results of this analysis, along with an evaluation of the key findings, are presented in the following section. This analysis provides valuable insights into the strengths and areas for improvement within the application, informing future development and refinement efforts.

Number	Question	Category	Anchors	Copyright (item source)
1	In the computer generated world I had a sense of "being there".	PRES	Not at all - very much	Slater & Usoh (1994) [1]
2	How responsive was the environment to actions that you initiated (or performed)?	INV	Very bad - Very good	Witmer & Singer (1994) [3]
3	Somehow I felt that the virtual world surrounded me?	SP	Fully Disagree - Fully Agree	IPQ
4	I felt like I was just more than perceiving pictures?	SP	Fully Disagree - Fully Agree	IPQ
5	I felt present in the virtual space.	SP	Fully Disagree - Fully Agree	IPQ
6	I had a sense of acting in the virtual space, rather than operating something from outside.	SP	Fully Disagree - Fully Agree	IPQ
7	I was completely captivated by the virtual world.	INV	Fully Disagree - Fully Agree	IPQ
8	How well could you read the emotions of the other users avatar?	REAL	Very bad - Very good	Self
9	How much did your experience in the virtual environment seem consistent with your real-world experience?	REAL	Not consistent - Very consistent	Witmer & Singer (1994) [3]
10	The virtual world seemed more realistic than the real world	REAL	Fully Disagree - Fully Agree	IPQ

Table 4.1.: Used questions in the questionnaire

4.2. Evaluation

The overall feedback received was notably positive, with respondents generally expressing a sense of being impressed by the technology, especially after receiving brief explanations about how it operates. The results of the surveys were encouraging, with the majority of responses falling above the midpoint of the rating scale. Specifically, more than 64 % of all responses were rated above 3 out of 5, indicating a favorable reception. A total of 100 responses were recorded across all questions, with 64 of those responses scoring higher than 3. This translates to a 64 % proportion of positive feedback, as reflected in Table 4.2, where two out of the three categories scored above 3, reinforcing the overall positive sentiment. However, this also underscores the area of improvement, particularly in enhancing experienced realism.

IPQ item	mean
Spatial presence	4.14
Experienced realism	2.83
Involvement	3.75

Table 4.2.: Mean results for each category

Delving deeper into the individual survey questions reveals additional insights. The general sense of presence within the application was rated quite positively, with all questions related to body presence achieving scores above 3. This suggests that the application successfully created a strong sense of immersion, allowing users to feel as though they were physically present within the virtual environment.

As illustrated in Table 4.3, the overall score across all questions remained above 3 except for Question 10, further confirming that the application was well-received by users. Notably, Question 3, which asked, "Somehow I felt that the virtual world surrounded me," received the highest average score of 4.4, indicating a strong sense of spatial presence. On the other hand, Question 10, "The virtual world seemed more realistic than the real world," had the lowest average score of 1.8, highlighting a key area where realism could be significantly improved.

The relatively lower scores in certain areas, such as Question 2, which pertains to interaction with the environment, and Question 8, which addresses the recognition of the avatar's emotions, suggest that these aspects of the application require further refinement. While Question 2 scored just above 3, indicating a satisfactory level of interactivity, there is clearly room for enhancement to create a more engaging and realistic user experience. Similarly, the ability to generate and accurately convey emotions through the avatar, as reflected in Question 8's slightly positive score, presents another opportunity for improvement. Now we should also look at the

individual answers to the questions in detail.

Question	Average Score
Question 1	4.0
Question 2	3.3
Question 3	4.4
Question 4	4.1
Question 5	4.2
Question 6	3.9
Question 7	4.2
Question 8	3.3
Question 9	3.4
Question 10	1.8

Table 4.3.: Average Score of each question

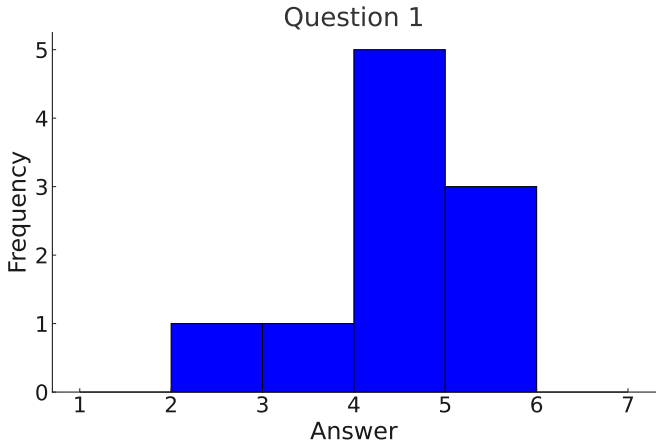


Figure 4.1.: Distribution of the Question 1 “In the computer generated world I had a sense of “being there””

Question 1, “In the computer-generated world, I had a sense of “being there””, as shown in Figure 4.1, reveals that the overall feeling of body presence among testers was very strong. The responses indicate that participants felt as though they were genuinely immersed in an actual environment, suggesting that the application successfully conveyed a realistic sense of being present within the virtual world.

The results for Question 2, “How responsive was the environment to actions that you initiated (or performed)?”, as depicted in Figure 4.2, are somewhat less

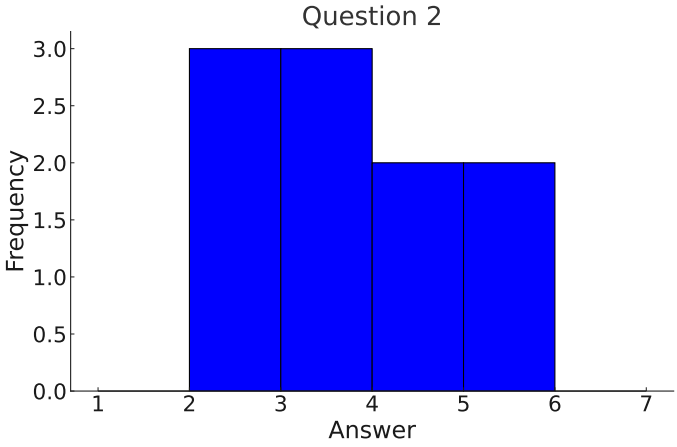


Figure 4.2.: Distribution of the Question 2 "How responsive was the environment to actions that you initiated (or performed)?"

impressive. With an average score of 3.3, the data suggests that while the environment's responsiveness was adequate, there is room for improvement. The results are not entirely negative, but they do indicate that enhancing the interactivity and responsiveness of the environment could significantly improve the user experience.

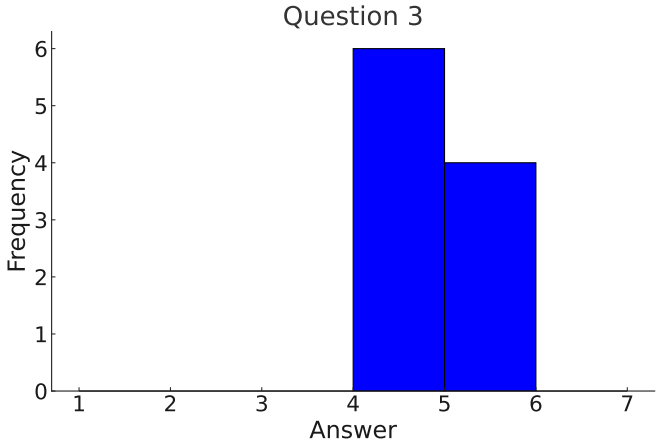


Figure 4.3.: Distribution of Question 3 "Somehow I felt that the virtual world surrounded me?"

For Question 3, "Somehow I felt that the virtual world surrounded me?", the results were very positive, as demonstrated in Figure 4.3. This indicates that the application effectively created a strong sense of spatial presence, further corroborated by the responses to Question 5, "I felt present in the virtual space," shown in

Figure 4.5. Together, these results suggest that users felt deeply immersed within the virtual environment.

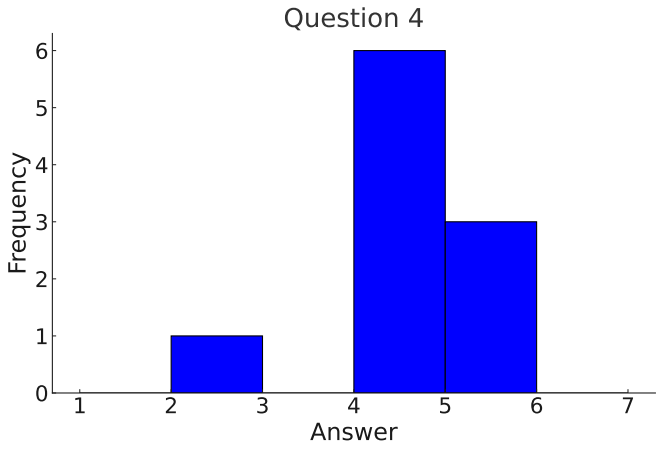


Figure 4.4.: Distribution of Question 4 "I felt like I was just more than perceiving pictures?"

Question 4, "I felt like I was more than just perceiving pictures?", also received positive feedback, with the exception of one response. The overall responses indicate that the transmission of generated images within the virtual space is functioning well. Users reported that the images conveyed a good sense of realism and that their movement within the virtual environment was smooth and fluid, rather than appearing as a disjointed slideshow. The details of these responses can be seen in Figure 4.4.

As previously mentioned, Question 5 received strong results, similar to Question 6, "I had a sense of acting in the virtual space, rather than operating something from outside." This question, which also falls under the category of Spatial Presence, highlights that testers felt they were genuinely within the environment, engaging directly with it rather than interacting from a detached perspective, such as sitting in front of a monitor.

Question 7, "I was completely captivated by the virtual world," shown in Figure 4.7, is categorized under involvement and suggests that the level of user engagement within the virtual world was very high. The participants were deeply absorbed in the virtual experience, indicating that the application was successful in capturing their full attention.

Questions 8 to 10 fall under the category of Experienced Realism, which, overall, is the weakest area of the survey. Question 8, "How well could you read the emotions of the other user's avatar?", received mixed responses with an average score of 3.3. This suggests that while testers could generally interpret the emotions of

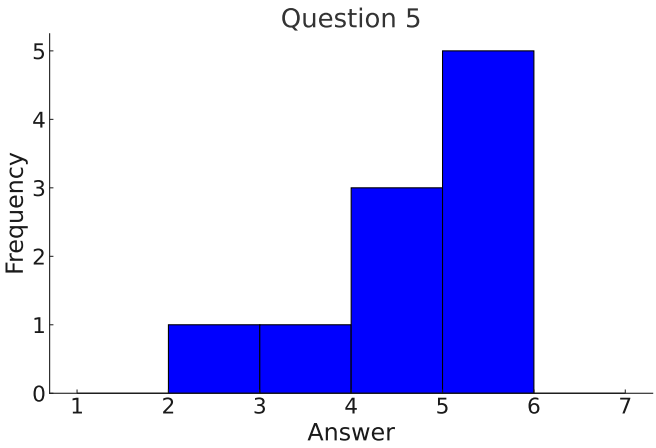


Figure 4.5.: Distribution of Question 5 "I felt present in the virtual space"

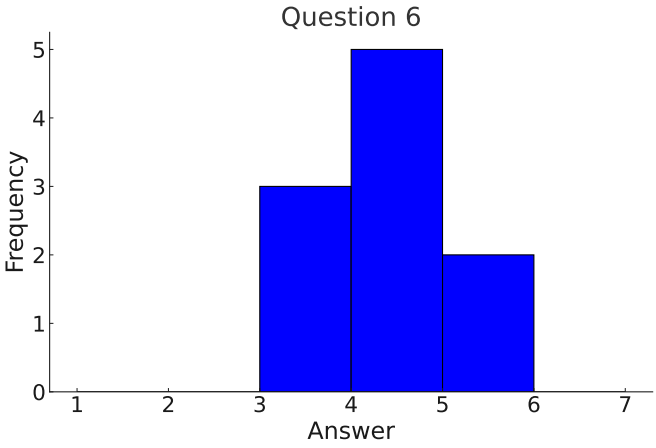


Figure 4.6.: Distribution of Question 6 "I had a sense of acting in the virtual space, rather than operating something from outside."

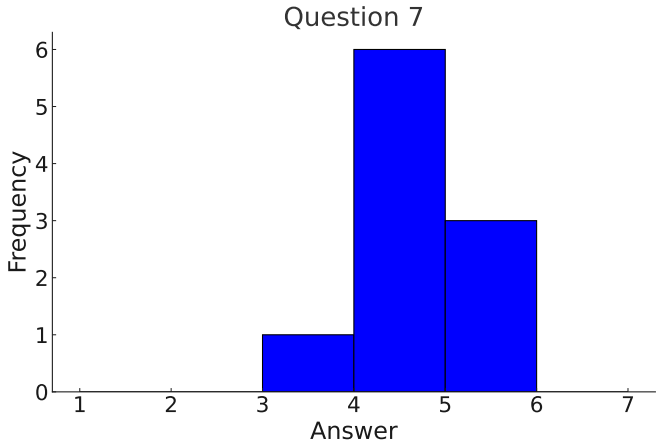


Figure 4.7.: Distribution of Question 7 “I was completely captivated by the virtual world.”

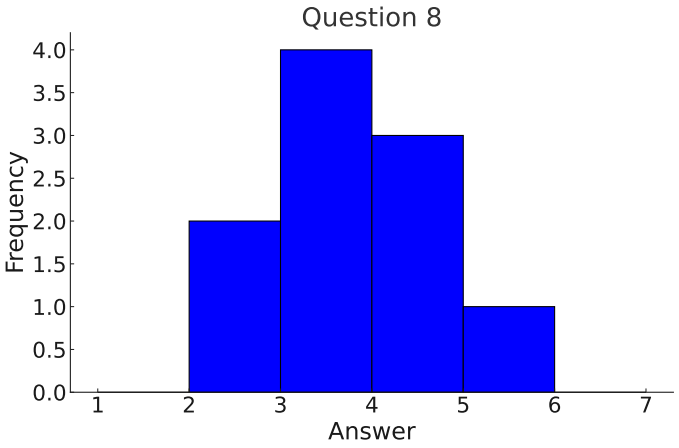


Figure 4.8.: Distribution of Questions 8 “How well could you read the emotions of the other users avatar?”

the generated avatars, there were instances where this was challenging, indicating a need for improvement in this area. The corresponding results can be seen in Figure 4.8.

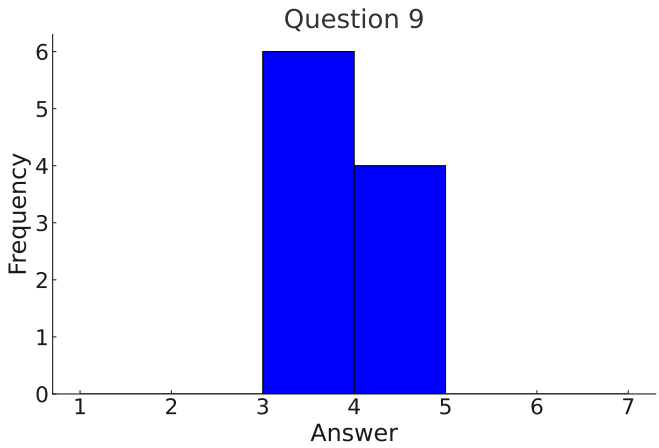


Figure 4.9.: Distribution of Question 9 "How much did your experience in the virtual environment seem consistent with your real-world experience?"

Question 9, "How much did your experience in the virtual environment seem consistent with your real-world experience?", also received mediocre responses, with a similar average score of 3.4. The results, illustrated in Figure 4.9, suggest that while the virtual experience was somewhat aligned with real-world expectations, inconsistencies were noted. For instance, some testers pointed out a lack of interaction with the environment, such as missing hand gestures or inconsistencies in object design, like a door lacking a doorknob.

The final question, Question 10, "The virtual world seemed more realistic than the real world," received the lowest scores, with an average of just 1.8, as shown in Figure 4.10. This result, along with the other findings in the Experienced Realism category, clearly indicates that the overall realism of the application needs significant improvement. Enhancing the realism of objects, interactions, and the environment is crucial for creating a more believable virtual experience. In Table 4.4 you can see the distribution of all the answers from all surveys.

For a more detailed analysis, a one-sample t-test was conducted to determine whether the mean scores significantly differed from 3. The results, presented in Table 4.5, provide interpretations of these findings. Additionally, a binomial test was performed to assess whether 50 % or more of the responses were positive. The p-value obtained from this test was approximately $p = 0.0066$, indicating that the proportion of responses greater than 3 is statistically significantly different from 50 %. This result confirms that more than half of the responses exceeded a score of

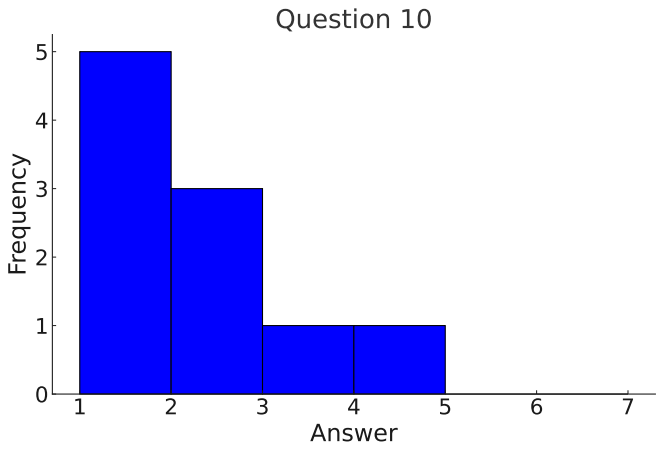


Figure 4.10.: Distribution of Question 10 "The virtual world seemed more realistic than the real world"

Answer	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	0	0	0	0	0	0	0	0	0	5
2	1	3	0	1	1	0	0	2	0	3
3	1	3	0	0	1	3	1	4	6	1
4	5	2	6	6	3	5	6	3	4	1
5	3	2	4	3	5	2	3	1	0	0

Table 4.4.: Distribution of Responses

3, reinforcing the overall positive reception of the application.

Question	Test Statistic	P-Value	Statistical Interpretation
Question 1	3.35	0.008	Significantly different from 3; mean significantly greater than 3.
Question 2	0.81	0.43	Not significantly different from 3; similar to 3, no evidence to reject the hypothesis that mean is 3.
Question 3	8.57	1.268	Significantly different from 3; mean significantly greater than 3.
Question 4	3.97	0.0032	Significantly different from 3; mean significantly greater than 3.
Question 5	3.67	0.0051	Significantly different from 3; mean significantly greater than 3.
Question 6	3.85	0.0038	Significantly different from 3; mean significantly greater than 3.
Question 7	6	0.0002	Significantly different from 3; mean significantly greater than 3.
Question 8	0.999	0.343	Not significantly different from 3; similar to 3, no evidence to reject the hypothesis that mean is 3.
Question 9	2.44	0.0367	Significantly different from 3; mean significantly greater than 3.
Question 10	-3.67	0.0051	Significantly different from 3; mean is much lower than 3

Table 4.5.: One-sample t-tests

5. Conclusion and Future Work

5.1. Summary

The research and development of the VR application proved to be a very time consuming task, demanding a significant investment of effort and dedication. A substantial portion of this time was devoted to precisely examining state-of-the-art applications, frameworks, and libraries, carefully assessing what was currently available in the field. This thorough investigation was necessary to ensure that the most suitable tools and technologies were identified and selected for the project. However, this process was not without its challenges. The exploration of different libraries and frameworks often proved to be a slow process, as many of them were incompatible with each other, leading to frustrating dead ends where certain combinations simply did not function as expected.

Throughout the extended period during which this thesis was developed, numerous ideas underwent significant transformations. Some concepts that initially seemed promising were either modified to better fit the evolving requirements or were entirely discarded in favor of new, more viable alternatives. This iterative process of refining and rethinking ideas was crucial, albeit challenging, as it demanded constant adaptation and flexibility. One of the most formidable challenges encountered was the task of identifying libraries that could seamlessly integrate with both the Unity Engine and Python. After identifying potential candidates, it was necessary to conduct extensive testing to determine which libraries were most effective and capable of supporting the desired functionalities.

The literature review conducted as part of this thesis provided invaluable insights and had a notably positive impact on the project. The diverse range of approaches discussed in the literature offered a broad spectrum of possibilities, enabling the exploration of various methods and techniques. This diversity was particularly beneficial when certain approaches did not yield the expected results, as it allowed for the quick pivot to alternative strategies, minimizing potential setbacks.

The prototype developed for this thesis has emerged as a robust and stable foundation, providing a solid platform for future expansions and the implementation of additional features. This foundation holds significant potential for establishing an alternative method of interacting with virtual reality environments using relatively simple means, such as images. The experiments and surveys conducted as part

of this research demonstrate that the prototype already exhibits a commendable level of spatial presence and user involvement, contributing to a strong sense of immersion within the virtual environment. However, there remains room for improvement, particularly in terms of enhancing the realism of the experience. Future iterations of the prototype could benefit from the incorporation of more sophisticated interaction possibilities and the refinement of generated images, with a focus on better recognizing and conveying emotions within the virtual space.

Overall, the journey of developing this VR application has been marked by both challenges and triumphs, with each obstacle serving as a stepping stone toward the creation of a more immersive and effective virtual reality experience.

5.2. Outlook and possible improvements

As previously mentioned, the surveys conducted have revealed that the current prototype is already a commendable achievement, particularly in the realms of spatial presence and user involvement. These findings underscore the success of the initial efforts to create a virtual reality experience that effectively immerses users within a spatially coherent environment, fostering a strong sense of being present within the virtual world.

Given the encouraging results of the current prototype, it is my hope that other researchers will continue to build upon the foundation established by this thesis. One of the key areas for future development lies in the enhancement of facial generation and the accurate depiction of emotions. The surveys highlighted a clear opportunity for improvement in this regard, particularly in the domain of Experienced Realism. Many participants noted that the emotions displayed by the generated avatars were either difficult to discern or not sufficiently expressive. Addressing this issue would be a significant step forward, and it would be highly beneficial for future work to place greater emphasis on refining the realism of these generated images. This could involve developing more sophisticated algorithms that better capture the nuances of real facial expressions, thereby making the avatars' emotions more immediately recognizable and authentic.

In addition to enhancing facial expressions, there is also potential to explore the integration of gesture recognition within the virtual environment. By developing methods to accurately transfer gestures, such as hand movements or pointing, the virtual experience could be made even more immersive and interactive. This would allow users to communicate and interact within the virtual space in a more natural and intuitive manner, further bridging the gap between the virtual and real worlds.

Moreover, the survey methodology itself could be expanded and refined. Incorporating additional questions, particularly those focused on the nuances of experiential realism, could provide deeper insights into the areas that require fur-

ther improvement. This expanded survey approach would help future researchers better understand the specific elements of the virtual experience that still fall short of achieving true realism, guiding them in their efforts to create an even more lifelike and engaging virtual reality environment.

In conclusion, while the current prototype represents a good foundation, there is still considerable room for growth and enhancement. By focusing on improving the realism of facial expressions, integrating gesture recognition, and refining survey methodologies, future researchers can build upon this work to create even more immersive and realistic virtual experiences. It is my hope that this thesis will serve as a stepping stone for further advancements in this exciting and rapidly evolving field.

Bibliography

[1] Mel Slater and Martin Usoh. "Representations Systems, Perceptual Position, and Presence in Immersive Virtual Environments". In: *Presence* 2 (Jan. 1993), pp. 221–233. DOI: 10.1162/pres.1993.2.3.221.

[2] Thomas Schubert, Frank Friedmann and Holger Regenbrecht. "Embodied Presence in Virtual Environments". In: (Dec. 1998). DOI: 10.1007/978-1-4471-0563-3_30.

[3] Bob G. Witmer and Michael J. Singer. "Measuring Presence in Virtual Environments: A Presence Questionnaire". In: *Presence: Teleoper. Virtual Environ.* 7.3 (June 1998), pp. 225–240. ISSN: 1054-7460. DOI: 10.1162/105474698565686. URL: https://doi.org/10.1162/105474698565686.

[4] Thomas Schubert, Frank Friedmann and Holger Regenbrecht. "The Experience of Presence: Factor Analytic Insights". In: *Presence* 10 (June 2001), pp. 266–281. DOI: 10.1162/105474601300343603.

[5] Holger Regenbrecht and Thomas Schubert. "Real and Illusory Interactions Enhance Presence in Virtual Environments". In: *Presence* 11 (Aug. 2002), pp. 425–434. DOI: 10.1162/105474602760204318.

[6] Thomas Schubert and Holger Regenbrecht. "Wer hat Angst vor virtueller Realität? Angst, Therapie und Präsenz in virtuellen Welten". In: (Jan. 2002).

[7] Thomas Schubert. "The sense of presence in virtual environments: A three-component scale measuring spatial presence, involvement, and realness". In: *Zeitschrift für Medienpsychologie* 15 (Apr. 2003), pp. 69–71. DOI: 10.1026//1617-6383.15.2.69.

[8] Ian Buck. "Gpu computing with nvidia cuda". In: (Aug. 2007), p. 6. DOI: 10.1145/1281500.1281647.

[9] A. Surendernath et al. "Mixing real and virtual conferencing: Lessons learned". In: *2012 IEEE Virtual Reality Workshops (VRW)*. 2012, pp. 79–80. DOI: 10.1109/VR.2012.6180891.

[10] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[11] Hadar Averbuch-Elor et al. "Bringing portraits to life". In: *ACM Trans. Graph.* 36.6 (Nov. 2017). ISSN: 0730-0301. DOI: 10.1145/3130800.3130818. URL: https://doi.org/10.1145/3130800.3130818.

[12] Joon Son Chung, Amir Jamaludin and Andrew Zisserman. *You said that?* 2017. arXiv: 1705.02966 [cs.CV]. URL: https://arxiv.org/abs/1705.02966.

[13] A. Nagrani, J. S. Chung and A. Zisserman. "VoxCeleb: a large-scale speaker identification dataset". In: *INTERSPEECH*. 2017.

[14] Kfir Aberman et al. *Deep Video-Based Performance Cloning*. 2018. arXiv: 1808.06847 [cs.CV]. URL: https://arxiv.org/abs/1808.06847.

[15] Jiahao Geng et al. "Warp-guided GANs for single-photo facial animation". In: *ACM Trans. Graph.* 37.6 (Dec. 2018). ISSN: 0730-0301. DOI: 10.1145/3272127.3275043. URL: https://doi.org/10.1145/3272127.3275043.

[16] Hyeongwoo Kim et al. *Deep Video Portraits*. 2018. arXiv: 1805.11714 [cs.CV]. URL: https://arxiv.org/abs/1805.11714.

[17] Patrick David Pazour, Andreas Janecek and Helmut Hlavacs. "Virtual Reality Conferencing". In: Dec. 2018, pp. 84–91. DOI: 10.1109/AIVR.2018.00019.

[18] Albert Pumarola et al. *GANimation: Anatomically-aware Facial Animation from a Single Image*. 2018. arXiv: 1807.09251 [cs.CV]. URL: https://arxiv.org/abs/1807.09251.

[19] Nataniel Ruiz, Eunji Chong and James M. Rehg. *Fine-Grained Head Pose Estimation Without Keypoints*. 2018. arXiv: 1710.00925 [cs.CV].

[20] Ting-Chun Wang et al. *Video-to-Video Synthesis*. 2018. arXiv: 1808.06601 [cs.CV]. URL: https://arxiv.org/abs/1808.06601.

[21] Olivia Wiles, A. Sophia Koepke and Andrew Zisserman. *X2Face: A network for controlling face generation by using images, audio, and pose codes*. 2018. arXiv: 1807.10550 [cs.CV]. URL: https://arxiv.org/abs/1807.10550.

[22] Caroline Chan et al. "Everybody Dance Now". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[23] Kuangxiao Gu, Yuqian Zhou and Thomas Huang. *FLNet: Landmark Driven Fetching and Learning Network for Faithful Talking Facial Animation Synthesis*. 2019. arXiv: 1911.09224 [cs.CV]. URL: https://arxiv.org/abs/1911.09224.

[24] S. N. B. Gunkel et al. "360-Degree Photo-realistic VR Conferencing". In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2019, pp. 946–947. DOI: 10.1109/VR.2019.8797971.

[25] Lingjie Liu et al. *Neural Rendering and Reenactment of Human Actor Videos*. 2019. arXiv: 1809.03658 [cs.CV]. URL: https://arxiv.org/abs/1809.03658.

[26] Aliaksandr Siarohin et al. *Animating Arbitrary Objects via Deep Motion Transfer*. 2019. arXiv: 1812.08861 [cs.GR]. URL: https://arxiv.org/abs/1812.08861.

[27] Aliaksandr Siarohin et al. "First Order Motion Model for Image Animation". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/31c0b36aef265d9221af80872ceb62f9-Paper.pdf.

[28] Yang Song et al. *Talking Face Generation by Conditional Recurrent Adversarial Network*. 2019. arXiv: 1804.04786 [cs.CV]. URL: https://arxiv.org/abs/1804.04786.

[29] Ting-Chun Wang et al. *Few-shot Video-to-Video Synthesis*. 2019. arXiv: 1910.12713 [cs.CV].

[30] Egor Zakharov et al. *Few-Shot Adversarial Learning of Realistic Neural Talking Head Models*. 2019. arXiv: 1905.08233 [cs.CV]. URL: https://arxiv.org/abs/1905.08233.

[31] Yipin Zhou et al. *Dance Dance Generation: Motion Transfer for Internet Videos*. 2019. arXiv: 1904.00129 [cs.CV]. URL: https://arxiv.org/abs/1904.00129.

[32] Egor Burkov et al. "Neural Head Reenactment with Latent Pose Descriptors". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. DOI: 10.1109/cvpr42600.2020.01380. URL: http://dx.doi.org/10.1109/CVPR42600.2020.01380.

[33] Egor Zakharov et al. *Fast Bi-layer Neural Synthesis of One-Shot Realistic Head Avatars*. 2020. arXiv: 2008.10174 [cs.CV]. URL: https://arxiv.org/abs/2008.10174.

[34] Ting-Chun Wang, Arun Mallya and Ming-Yu Liu. *One-Shot Free-View Neural Talking-Head Synthesis for Video Conferencing*. 2021.

[35] Lalit Kumar and Dushyant Kumar Singh. "Comparative analysis of Vid2Vid and Fast Vid2Vid Models for Video-to-Video Synthesis on Cityscapes Dataset". In: *2023 International Conference on Computer, Electronics & Electrical Engineering & their Applications (IC2E3)*. 2023, pp. 1–5. DOI: 10.1109/IC2E357697.2023.10262586.

[36] sky24h. *Face Animation in Real Time*. https://github.com/sky24h/Face_Animation_Real_Time [Accessed: (10.05.2024)]. 2023.

[37] Alex Trevithick et al. "Real-Time Radiance Fields for Single-Image Portrait View Synthesis". In: *ACM Transactions on Graphics (SIGGRAPH)*. 2023.

[38] *batch-face*. https://github.com/elliottzheng/batch-face [Accessed: (28.05.2024)].

[39] CVZone. *CVZone*. https://github.com/cvzone/cvzone [Accessed: (21.05.2024)].

[40] hecomi. *uChromaKey*. https://github.com/hecomi/uChromaKey [Accessed: (21.05.2024)].

[41] *HTC Vive System Requirements*. https://www.vive.com/us/support/vive/category_howto/what-are-the-system-requirements.html [Accessed: (18.07.2024)].

[42] ipgroup. *Presence Questionnaire 1*. www.igroup.org [Accessed: (21.05.2024)].

[43] *Nvidia CUDA Toolkit*. https://developer.nvidia.com/cuda-toolkit [Accessed: (28.05.2024)].

[44] *NVIDIA cuDNN*. https://developer.nvidia.com/cudnn [Accessed: (22.06.2024)].

[45] *OpenCV - About*. https://opencv.org/about/ [Accessed: (28.05.2024)].

[46] *PyTorch - About*. https://pytorch.org/docs/stable/index.html [Accessed: (28.05.2024)].

[47] *PyTorch - Nvidia Description*. https://www.nvidia.com/en-us/glossary/pytorch/ [Accessed: (28.05.2024)].

[48] *Unity Engine*. https://unity.com/products/unity-engine [Accessed: (28.05.2024)].

[49] *Unity Manual*. https://docs.unity3d.com/Manual/VROverview.html [Accessed: (28.05.2024)].

[50] zhanglonghao1992. *One-Shot Free-View Neural Talking Head Synthesis*. https://github.com/zhanglonghao1992/One-Shot_Free-View_Neural_Talking_Head_Synthesis [Accessed: (28.05.2024)].

A. Appendix

SURVEY ON EXPERIENCES IN VIRTUAL WORLDS

1. *In the computer generated world I had a sense of "being there".*

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Not at all — Very much

2. *How responsive was the environment to actions that you initiated (or performed)?*

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Very bad — Very good

3. *Somehow I felt that the virtual world surrounded me?*

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Fully disagree — Fully agree

4. *I felt like I was just more than perceiving pictures?*

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Fully disagree — Fully agree

5. *I felt present in the virtual space.*

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Fully disagree — Fully agree

6. *I had a sense of acting in the virtual space, rather than operating something from outside.*

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Fully disagree — Fully agree

7. *I was completely captivated by the virtual world.*

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Fully disagree — Fully agree

8. *How well could you read the emotions of the other users avatar?*

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Very bad — Very good

9. *How much did your experience in the virtual environment seem consistent with your real-world experience?*

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

not consistent — very consistent

10. *The virtual world seemed more realistic than the real world.*

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Fully disagree — Fully agree

Figure A.1.: Print version of the Presence Questionnaire

Figure A.2.: Link to Video Demonstration https://youtu.be/Tv2CMzFceew