Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Bachelor Degree in Informatics Engineering
Specialisation: Computing

Realistic face rendering for 3D mixed reality experience

Bachelor's Thesis of Tharangini Sankarnarayanan

Director: Josep Ramon Morros
Signal Theory and Communications Department

Co-Director: Javier Ruiz Hidalgo
Signal Theory and Communications Department

Tutor: Nuria Castell Ariño
Computer Science Department

Date: July 1st, 2019

Abstract

Virtual Reality (VR) has advanced significantly in the recent years. VR allows users to explore novel environments (both real and imaginary), play games, and engage with media in a way that is unprecedentedly immersive. However, compared to physical reality, sharing these experiences is difficult because the user's face is not easily observable from the outside and is partly occluded by the VR headset. Mixed Reality (MR) is a medium that alleviates some of this disconnect by sharing the virtual context of a VR user in a flat video format that can be consumed by an audience to get a feel for the user's experience.

Even though MR allows audiences to connect actions of the VR user with their virtual environment, empathizing with them is difficult because their face is hidden by the headset. Mixed reality is a way to convey what's happening inside and outside a virtual place in a two dimensional format. With this new technology, we're able to make a more complete picture of the person in virtual reality. Some of this disconnect is alleviated by Mixed Reality (MR), a related medium that shares the virtual context of a virtual reality user in a two dimensional video format allowing other viewers to get a feel for the user's virtual experience. Even though mixed reality facilitates sharing, the headset continues to block facial expressions and eye gaze, presenting a significant hurdle to a fully engaging experience and complete view of the person in virtual reality.

Google has a project to enhance the experience of virtual reality, our project aims to present a solution to address this problem by virtually removing the headset and revealing the face underneath it using a combination of 3D vision, machine learning and graphics techniques for images. We are now able to "remove" headsets and show a person's identity, focus and full face in virtual reality. The solution to this problem helps in enhancing communication through videos and video conferencing and in healthcare treatments for mental disorders.

Acknowledgement

First and foremost, I would like to thank the Almighty for giving me this opportunity and guiding me always.

I am eternally grateful to my mentors Professors Josep Ramon Morros, Professor Javier Ruiz Hidalgo and Professor Nuria Castell Ariño for letting me work on this project and their kind thoughtful guidance throughout.

I am greatly thankful to SASTRA University, for providing me this wonderful opportunity to work in Universitat Politecnica De Catalunya for my Bachelor thesis.

I would also like to thank Universitat Politecnica De Catalunya for allowing us to use the NVIDIA GPU Center of Excellence and giving us its resources and machines for developing this thesis.

Lastly, I would like to thank my parents for their eternal love, support and confidence

Contents

Index of Figures

Index of Table

Chapter 1

Introduction

1.1 Why the need to enhance the experience of Virtual and Mixed Reality?

Virtual Reality (VR) has advanced significantly in the recent years. VR allows users to explore novel environments (both real and imaginary), play games, and engage with media in a way that is unprecedentedly immersive. However, compared to physical reality, sharing these experiences is difficult because the user's face is not easily observable from the outside and is partly occluded by the VR headset. Mixed Reality (MR) is a medium that alleviates some of this disconnect by sharing the virtual context of a VR user in a flat video format that can be consumed by an audience to get a feel for the user's experience.

Even though MR allows audiences to connect actions of the VR user with their virtual environment, empathizing with them is difficult because their face is hidden by the headset. Mixed reality is a way to convey what's happening inside and outside a virtual place in a two dimensional format. With this new technology, we're able to make a more complete picture of the person in virtual reality. Some of this disconnect is alleviated by Mixed Reality (MR), a related medium that shares the virtual context of a virtual reality user in a two dimensional video format allowing other viewers to get a feel for the user's virtual experience. Even though mixed reality facilitates sharing, the headset continues to block facial expressions and eye gaze, presenting a significant hurdle to a fully engaging experience and complete view of the person in virtual reality.

Researchers in America are exploring the ways in which VR can be used across healthcare; such as for 'post-traumatic' stress disorder, for reducing amputee pain.

Participants who used VR reported reduced levels of pain and general distress and said they would like to use VR again during painful medical procedures. VR appears to trigger an array of

emotional, cognitive and attention processes that act on the body's pain modulation system, acting as a non-pharmacological pain killer.

1.2 Existing Solutions

1.2.1 State of the art

The problem to remove the headsets in mixed reality involves videos, which is an aggregation of consecutive images. We have worked to develop a prototype to remove headsets in images. The headsets can be referred to a masked region which needs to be inpainted. Image inpainting, or image completion, involves filling in missing regions of an image. It is an important step in many image editing tasks. It can, for example, be used to fill in the holes left after removing unwanted objects from an image. Humans have an uncanny ability to zero in on visual inconsistencies. Consequently, the filled regions must be perceptually plausible.

Generative adversarial networks focus on creating "deepfakes" for the facial images. Adversarial machine learning has other uses besides generative modeling and can be applied to models other than neural networks. The general idea of learning via competition between players dates back to at least 1959 with the influential work of Arthur Samuel, demonstrating that algorithms could learn to play checkers via adversarial self-play.

Ian Goodfellow is recognized by several sources as having invented GANs in 2014. This paper included the first working implementation of a generative model based on adversarial networks, as well as game theoretic analysis establishing that the method is sound.

GANs gained importance for fixing ancient paintings with missing portions. The process gained momentum in 2017, when a GAN was used for image enhancement focusing on realistic textures rather than pixel-accuracy, producing a higher image quality at high magnification for paintings. Later in 2017, the first faces were generated. These were exhibited in February 2018 at the Grand Palais. Faces generated by StyleGAN in 2019, drew comparisons with Deepfakes.

Beginning in 2017, GAN technology began to make its presence felt in the fine arts arena with the appearance of a newly developed implementation which was said to have crossed the threshold of being able to generate unique and appealing abstract paintings.

1.2.2 Related Work

Among other things, the lack of fine structure in the filled region is a giveaway that something is amiss, especially when the rest of the image contain sharp details. The work presented in a model called Edge-Connect paper is motivated by the observation that many existing image inpainting techniques generate over-smoothed and/or blurry regions, failing to reproduce fine details. The process of image inpainting is divided into a two-stage process: edge generation and image completion. Edge generation is solely focused on hallucinating edges in the missing regions. The image completion network uses the hallucinated edges and estimates RGB pixel intensities of the missing regions. Both stages follow an adversarial framework [18] to ensure that the hallucinated edges and the RGB pixel intensities are visually consistent. Both networks incorporate losses based on deep features to enforce perceptually realistic results. Like most computer vision problems, image inpainting predates the wide-spread use of deep learning techniques. Broadly speaking, traditional approaches for image inpainting can be divided into two groups: diffusion-based and patch-based. Diffusion-based methods propagate background data into the missing region by following a diffusive process typically modeled using differential operators. Patch-based methods, on the other hand, fill in missing regions with patches from a collection of source images that maximize patch similarity. These methods, however, do a poor job of reconstructing complex details that may be local to the missing region. More recently deep learning approaches have found remarkable success at the task of image inpainting. These schemes fill the missing pixels using learned data distribution. They are able to generate coherent structures in the missing regions, a feat that was nearly impossible for traditional techniques. While these approaches are able to generate missing regions with meaningful structures, the generated regions are often blurry or suffer from artifacts, suggesting that these methods struggle to reconstruct high frequency information accurately. Then, how does one force an image inpainting network to generate fine details? Since image structure is well represented in its edge mask, it is possible to generate

superior results by conditioning an image inpainting network on edges in the missing regions. Clearly, there is no access to the edges in the missing regions. Rather, an edge generator/sketch generator is used, that hallucinates edges in these areas. The approach of "lines first, color next" is partly inspired by the understanding of how artists work.

"In line drawing, the lines not only delineate and define spaces and shapes; they also play a vital role in the composition", as artist Betty Edwards said, highlights the importance of sketches from an artistic viewpoint. Edge recovery, we suppose, is an easier task than image completion. This is the base model that essentially decouples the recovery of high and low frequency information of the inpainted region.

1.3 Problem Formulation

Virtual Reality (VR) enables remarkably immersive experiences, offering newer ways to view the world and the ability to explore novel environments, both real and imaginary. However, compared to physical reality, sharing these experiences with others can be difficult; as virtual reality headsets make it challenging to create a complete picture of the people participating in the experience. The headsets obstruct virtual communication. If one is watching someone else using virtual reality, it is hard to tell what is going on and what they are seeing. This also obstructs viewing one another's facial expressions without an avatar representation.

The project aims to provide a solution to address this problem by revealing the user's face by virtually "removing" the headset and create a realistic see-through effect by the combination of 3D vision, machine learning and graphics techniques and develop a prototype to demonstrate the system using a calibrated VR setup including a headset (like the HTC Vive), a green screen, and a video camera (in this case a phone camera) combined with accurate tracking and segmentation.

1.4 Why Computer Vision and Machine Learning?

In our domain, trying to solve the problem mathematically, with statistics, will only take us so far. Making that solution of use in real life with respect to response time and scaling it to real world size would require huge computational and storage capabilities. Even if we were to use the resources of such magnitude, the solution still will not be as elegant as one involved Computer Vision because computer Vision is an emulation of the most advanced, the most powerful computer there exists: The Human Brain.

We try to solve the problem of the usage of VR headsets that makes it challenging to create a complete picture of the people participating in the experience. The face hidden revealing the user's face by virtually removing the headset and create a realistic see-through effect. A computer can be trained to do the same with the help of 3D vision, machine learning and graphics techniques. We choose Generative Adversarial Networks as the Machine Learning technique to be used, because GANs inherently by their design are similar to the visual cortex of the brain and likewise more suited to deal with the processing visual data such as images.

1.5 Methodology

This section gives a brief overview of the methodology to be followed in the project. It involves five major steps:

- Create a dataset with images of people with and without headsets
- Refine the data into a dataset suitable for training the network
- Build a convolution neural network model
- Tune the hyper-parameters of the model
- Deploy the final tuned model

Chapter 2

Scope, Stakeholders, and Challenges

2.1 Scope

The initial phase of the project will involve creating the dataset, working with the dataset to test different mapping schemes of images. This will be followed by pruning the dataset to remove unsuitable images to enhance the quality of training data.
Second, the most optimum values for the parameters of the adversarial network model will be decided by running experiments on models with different parameters to study their impact on the result.
Finally, the fully trained model allows the removal of the headsets in images in mixed reality.

2.2 Stakeholders

This project has a very diverse and large target audience as elucidated below.

- Headset removal is poised to enhance communication and social interaction in VR itself with diverse applications like VR video conference meetings, multiplayer VR gaming, and exploration with friends and family. Going from an utterly blank headset to being able to see, with photographic realism, the faces of fellow VR users promises to be a significant transition in the VR world

- VR can be used across healthcare; such as for post-traumatic stress disorder and other mental disorders, for reducing amputee pain by noticing various changes in the person's expressions and emotions.

2.3 Challenges

2.3.1 Dataset

The dataset consists of images of faces of people without the headsets and wearing the headsets. First, there is no dataset publicly available for usage. Hence, we created a dataset with frontal facial images of 13 people with and without the headsets. There are 3 images each of a person wearing the headsets and without wearing the headsets of all the people.

2.3.2 Computational Resources

The more complicated our convolutional network model is, the more layers it has and consequently the longer it takes to train. Training a simple 2-layer convolutional network for takes weeks for 1 epoch and testing for 2 days for 1 epoch takes approximately 30 minutes on a machine with Intel Core i7-3537 2GHz processor and 8 GB of RAM. Thus, to achieve good results, we would be required to train a more complex network, getting enough computational resources to run the experiments fast is a big challenge.

Chapter 3

Project Management

3.1 Initial Milestone

Before beginning the project, we estimated the temporal and financial aspects of the project which include the schedule to be followed and the expected budget of the project. We present those estimates in this section. After the completion of the project, we re-evaluated the temporal and financial aspects of the project to a finer precision and compared it to the initial estimates. The comparisons are presented in the next section (4.2 Final Milestone).

3.1.1 Temporal Planning

The project begins on the 11th February. The hard deadline is on the 24th of June, a week before the final defense (Expected to be on the 1st of July). Thus, a total of 18 weeks is at our disposal for the completion of the project. We will be undertaking the project by splitting it into different smaller tasks and executing the tasks in an ordered manner. The tasks have been ordered in a sequenced manner for execution. They have been described in detail in the following section.

3.1.1.1 Tasks

- Initial Preparatory Research
  This, being the first task, involves obtaining the background knowledge required to embark on the project. The main workers on this task will be the software designer and the project manager. It can be subdivided into three tasks as follows:
  Learning about Generative Adversarial Networks and Convolutional Neural Networks
  Learning with PyTorch
  Reading related Research Papers

Time: A duration of 2 weeks (50 hours) is both necessary and sufficient for this task.

Resources: Working Laptop with Internet Connection, Web Browser, PDF Viewer

- Experiments

  - Refining the training data

    One of the primary tasks of the tasks of the project. This involves working with the dataset to find out the most optimal organization of the training data. The main refinements to consider are:

    - Images are focused on the frontal face
    - Images are then resized to size 256*256

  - Deciding the model architecture

    A Generative Adversarial Network (GAN) is used as the base model. Given a training set, this technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics. A known dataset serves as the initial training data for the discriminator. Training it involves presenting it with samples from the training dataset, until it achieves acceptable accuracy. The generator trains based on whether it succeeds in fooling the discriminator.

    The image inpainting network that consists of two stages:

    1) Edge Generator,
    2) Image Completion Network

    Both the stages follow an adversarial model. The first stage consists of a generator/discriminator pair. Let G1 and D1 be the generator and discriminator for the edge generator, with the generator being executed in a GAN, and the discriminator is a CNN network for the image completion network. Let G2 and D2 be the generator and discriminator for the image completion network, with the

generator being executed in a GAN, and the discriminator is a CNN network for the image completion network.

This task deals with the building a prototype for the problem. Specifically, the generators consist of encoders that down-sample twice, followed by eight residual blocks and decoders that up sample images back to the original size. The discriminator consists of dilated convolutions with a dilation factor of two are used instead of regular convolutions in the residual layers. Instance normalization is used across all layers of the network.

- Simulation

  This task can be divided into three parts:

  Running trial simulation based on current model with the current dataset

  Analyzing the results

  Based on the results, we go back to Task 2.A to further refine the dataset for better accuracy. Then execute Task 2.B again with to update the parameters based on the obtained results. Then run Task 2.C a new simulation.

  Thus, Tasks 2.A, 2.B and 2.C form a continuous cyclic process of experimenting with the goal of bettering the model as much as possible. All three subtasks require an equal distribution of time and resources allocated for the collective task. Also, all these sub-tasks would require the major involvement of the project manager, software designer and software developer.

  Time: Thus, this task of Experiments is the major and the most important part of this project and will require the most amount of time. Hence, a total amount of 12 weeks has been allocated for this task, which is approximately 70% of the total duration of the project.

  Resources: Python IDE, Working Laptop with Internet connection

- Testing

  One of the final stages of the project. This task will involve conducting a thorough test of the entire project. The main objectives are testing the GAN prototype. It will encompass testing the sub-components too. The software tester is responsible for this task throughout under the supervision of the project manager.

  Time: A continual testing occurs as according to the agile methodology to completely verify all the components of the project.
  Resources: Python IDE, Working Laptop with Internet connection

- Documentation

  Last, but not the least part of the project. The final phase of the project where the thesis is written. As, the thesis requires a lot of time to write correctly and thoroughly, a dedicated time of a week has been allocated for the same. Within this period, the presentation for the final defense is also scheduled to be prepared. The project manager takes care of this task, with inputs from the software designer, software developer and software tester.

  Time: Again, this task has been allocated 2 weeks for its completion as it involves writing the thesis and an additional task of preparing the presentation for the defense.
  Resources: Microsoft Word, Working Laptop with Internet connection

- Project Management Course

  The Project Management Course (GEP) is an important task that will run in parallel with the project till its completion (1st April). There is a total of 6 deliverables as part of this course, which aims to teach the basics of project management. It helps us plan our project and has the added benefit of ensuring a good plan for our project.

Time: 75h (37.5 hrs of Guided Learning + 37.5 hrs of Self Study)

Resources: Web Browser, PDF Viewer, Microsoft Word, Microsoft Excel

Summary

Tasks	Start Date	End Date	Duration (Weeks)	Duration (Hrs)
Initial Preparatory Research	Feb 11,19	Feb 25, 19	2	50
Experiments	Feb 26,19	May 23, 19	12	250
Testing	May 24, 19	June 13,19	2	50
Documentation	June 14, 19	June 24, 19	1	50
Project Management Course	Feb 13, 19	April 01,19	6	75

Table 1: Summary of Schedule

3.1.1.2 Action Plan

We propose to execute the project as per our plan. Table 1 and Figure 1 give the tabular and Gantt chart representation of the schedule respectively. We will finish the tasks one by one in the listed order. The given order is logical and sequenced in such a way that there will be no scenario wherein a task has to be started without its preceding requirement's completion.

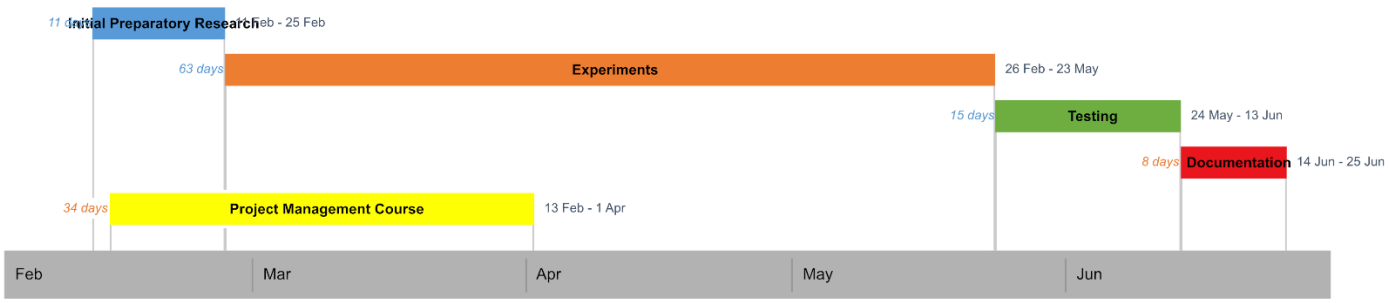

Figure 1: Gantt Chart of Schedule

Also, we have distributed our available period of 18 weeks among the different tasks with a reasonable estimate, accounting for 30h of work per week for a total of 540h for the project. Optimistically, this is a definite and attainable schedule, as delays due to holidays and other unforeseen circumstances leading to losses in the number of hours per week can be compensated by putting in more hours in the following weeks. Thus, in extreme cases, most likely a workload of 40h/week can be present. However, since this amount is also an easily surmountable figure, we hope to complete the project in time.

We also realize that as in the case of any other projects, there might be unexpected obstacles during each task, especially beginning from the Experiments phase. Thus, if we run out of time during each task, we will go forward with the best version we have then. Of course, there will be priorities that will be met such as more than 50% accuracy of the adversarial network. In case, pessimistically, we are faced with a severe need for more time to deploy the project, we can cut down on the extent of the Experiments phase. Planned weekly meetings with the director and the co-director of this project will further help us to track, expedite and follow our proposed plan to ensure deadlines are met.

3.1.2 Financial Planning

The budget of the project has been estimated in the following section. We have divided the total budget into three main categories of Hardware, Software, and Human Resources.

3.1.2.1 Hardware Costs

Amortization has been using the formula:
Amortized Cost = Actual Cost * (No of years in Use/ Useful Years)

No of Years in Use is the length of the project in years, which is, 0.42 years (5 months)

Table 2(Fig. Below) shows the required calculation for the Hardware Costs, which amounts to a total of Euros

Product	**Price**	**Units**	**Useful life**	**Amortization (in Euros)**
Dell Inspiron 15	715 €	1	5 years	74.47
HTC Vive	710 €	1	5 years	74.00
Google Nexus 41	299 €	1	5 years	33.56
Total estimated	1699 €			181.03

Table 2: Hardware Resources

3.1.2.2 Software Costs

All the software we need and will use for the project are available free of cost. They are listed as below.

- Ubuntu 16.10
- PyCharm Community Edition
- Google Chrome
- Sublime Text Editor
- Libre OpenOffice Writer
- Libre OpenOffice Calc
- Various Python Packages (listed in Requirements Section)

3.1.2.3. Human Resources Costs

Role Estimated hours	**Euros/Hours**	**Hours**	**Total (in Euros)**
Project Manager	50.00	135	6750
Software Designer	35.00	195	6825
Software Programmer	25.00	100	2500
Software Tester	20.00	110	2200
		Grand Total	18275.00 €

Table 3: Human Resources Budget

Table 3 gives a detailed distribution of the Human Resources budget, which is a total of 18,275 Euros.

The human resources budget has been estimated by considering the different roles required to complete the project. We wish to bring to notice that there is only one person working on the project and he will adopt different roles as and when required. They are:

- Project Manager

  She oversees the entire project planning. She must work during all tasks and phases of the project ensuring the project is proceeding as per plan.

- Software Designer

  She is responsible for the design of the program we are going to develop. She will have a major portion of her work during the Experiments phase, when she should redesign the project model according to the results of the experiments.

- Software Programmer

  She is the actual coder. She writes code for the software designed by the Software Designer. Mainly, she must work during the Experiments phase alongside the Software Designer.

- Software Tester
  As the name suggests, she takes care of testing the workings of the entire project, detecting bugs, and reporting it to the Project Manager. She will be continuously involved in the initial stages. However, she will have a dedicated period as the final phase for thorough testing of the developed project.

3.1.2.4 Indirect & Unforeseen Costs

In this section, we have considered costs incurred from other categories than Hardware, Software, or Human Resources. It considers indirect costs and unforeseen costs. This accounts for possible deviations in the course of the project.

Under Indirect Costs, we have Internet. We use eduroam Wi-Fi for the project which is free. Thus, we have not considered expenses due to Internet. However, in case we are unable to use eduroam, then we must spend money for Internet. This is considered under unforeseen costs. Also, we need not consider office spaces as an expense, as the physical material constituting office space will be intact even after out project completion for further use.

The reason for allocating a sum of money to unforeseen costs is to prepare for contingencies. We hope the project will proceed per plan, but there might be sudden and unforeseen changes in the plan. We should be prepared to handle these as well.

Some situations that might crop up are:

- Human resource estimation might exceed our budget. For example, if during the testing phase, an error is detected and some part of the code must be modified, we should use the Software Designer and Software Programmer during the testing phase also which we have not accounted for in the calculation. Such instances will come under unforeseen costs.

- We don't expect to use any more hardware than we have listed but there might be two cases in which we might have to.

- Any of our hardware gets corrupt or needs replacement

- Out laptop isn't computationally sufficient to train our network. In that case, we hope to use the supercomputing cluster present at Barcelona Supercomputing Center (BSC), which is free of cost to academicians.

3.1.2.5 Task-based distribution

	Task 1	Task 2	Task 3	Task 4	Task 5	Grand Total (in Euros)
Hardware Costs						
Laptop	14.90	14.90	14.90	14.90	14.90	
HTC Vive	14.00	14.00	14.00	14.00	14.00	
Mouse	0.33	0.33	0.33	0.33	0.33	
Total Hardware Costs	29.23	29.23	29.23	29.23	29.23	146.15
Human Resources Costs						
Project Manager	30.00	60.00	15.00	10.00	20.00	
Software Designer	30.00	120.00	25.00	0.00	20.00	
Software Programmer	0.00	80.00	10.00	0.00	10.00	
Software Tester	0.00	40.00	10.00	50.00	10.00	
Total Human Resources Costs	2550.00	10000.00	2075.00	1500.00	2150.00	18275.00
Total Costs	2565.23	10015.23	2101.69	1526.69	2165.23	18374.06

Table 4: A Task based distribution of the Resources

A task-based view of the different facets of our budget has been presented in Table 4.

3.1.2.6 Budget Control

Seeing, that we have accounted for almost all possible deviations for the budget in the project, we have a strong belief that the project will not exceed the proposed budget. Our consideration of unforeseen costs and possible deviations is an extensive one and is a robust budget control measure. Thus, it is an indicator that the proposed budget is definitely an upper limit on the actual budget. All considerations in case the project requires more resources, including time, have been dealt with under unforeseen costs. In case, the project finishes earlier than expected or requires fewer resources, then we will have a lower actual budget.

3.1.2.7 Total Budget

	Cost (in Euros)
Hardware	1400.00
Software	0.00
Human Resources	18275.00
Contingencies (Unforeseen Costs)	0.00
Total	19675.00

Table 5: Total Budget

Thus, as apparent from the Table 5 (Fig. Above), the total estimated budget for this project is 19675.00 euros.

3.1.3 Sustainability

We analyze the sustainability of the project under three main categories: Economic, Social and Environmental Sustainability. They have been described in detail, respectively in the following sections.

3.1.3.1 Economic Sustainability

Our project is economically a very sustainable one, as apparent from the budget we have proposed. We have assessed all costs (material and human) that will be incurred during the project. We also have accounted for indirect and unforeseen costs that may have escaped our initial estimate.
All the software used in the project is free and there is a very minimal requirement of hardware as well. With respect to human resources, we have a reasonable and viable estimate, which can exceed only in highly unrealistic circumstances. It is also impossible to reuse code of any kind completely. It cannot be done with fewer resources of any kind. Thus, we are confident that this is economically the most efficient budget possible for this project.
Next, for the realistic rendering of images for enhancing communication through virtual reality, there is a need to record images before the beginning of the usage that is used for training and then there is testing that occurs.
Finally, the potential return of investment of this project when implemented on videos, is huge as it enhances the communication interaction in virtual reality. Thus, there is a huge scope for a large-scale implementation of this project with corporate investments. We think this project qualifies as an 8, on a scale of 1 to 10, in the Economic Sustainability analysis.

3.1.3.2 Social Sustainability

Social Sustainability is where our project scores the highest because it has a huge positive impact on society. This project does not have any regional, political or social barriers to its successful implementation because it deals with the problem of enhancing the communication through virtual reality and using to various aspects of other fields like healthcare.
Also, it offers a solution that is much of a less hassle of pre-captured images for training. This could ease the communication levels and enhance aspects of gaming and healthcare.
Considering the usefulness and the potential and efficiency of our project as a social contributor, we award it a 9 on a scale of 1 to 10, in the Social Sustainability analysis.

3.1.3.3 Environmental Sustainability

Under environmental sustainability, we analyze the project during the course of development. During the development, the only environmental concern for the project is the consumption of electricity. The main contributor for electricity consumption is our Laptop. However, since laptops have become a necessity in today's world and doing a project without a laptop is next to impossible. We chose to ignore the electricity factor. Even if we had considered it, we require 540 hours for the project. Assume out laptop consumes 200 W while working, which gives us a total of 108 KWH, which is equivalent to 41.58 kg of CO2, which is well within the permissible amount for any project. A positive feature of this project is that the project is also entirely paper-free. We only use computers for analysis, design, development and implementation.

Considering all the points mentioned in the above discussion, we grade the Environmental Sustainability of this project as 8, on a scale of 1 to 10.

3.2 Final Milestone

After the completion of the project, we compared the initial milestone estimates with the actual temporal and financial outcomes. The main goal of the project and the methodology to be followed did not change from the inception of the project throughout its course. Thus, there are no major changes to temporal and financial aspects of the project as well. The comparisons are presented in the following sections.

3.2.1 Temporal Planning

The initial schedule was an inaccurate estimate of the timeline of the project. All the tasks roughly took the same amount of time as expected.
Some minor changes were observed in the first task of Initial Preparatory Research, which took longer than the allocated 2 weeks. It took a week extra, owing to the fact that the project developer

needed more time to understand the technologies used in the project and familiarize herself with the technical environment, apart from gaining background knowledge on the related subjects.
However, in a compensated manner, the task of Implementing the code a week less than the initial estimate. With the thorough understanding of the project, obtained from the extensive phase of running experiments, the task of implementations was easier than expected.
Thus, the overall schedule and deadlines remain unaffected at the final milestone.

3.2.2 Financial Planning

With respect to the budget of the project, all expected expenses were accurate. No extra resources in terms of hardware, software or time were necessary for the completion of the project. Thus, there were no deviations from the initial estimate of the budget.

Particularly, we needed more hardware resources because the laptop did not have enough computational capacity to run the experiments in a timely and efficient manner. So, we used the servers in Barcelona Supercomputing Center to remotely run our experiments and get the results in less time. Since, we had already considered this expense under Contingencies (Table 5), for which we had allocated 1000 euros, renting the extra computing resource did not deviate the budget.
To be precise, we had allocated an amount that was more than required for Contingencies. The only contingency that had to be handled was there rent of the extra hardware resource, which costs only 300 euros.

3.2.3 Sustainability

The sustainability factor of the project remains the same across economic, social and environmental dimensions as the project proceeded according to the plan without any major deviations in terms of time, effort, money, or the result. The deliverables that were promised were

given within the expected temporal, financial, social and environmental boundaries. Thus, there are no changes in the sustainability of the project.

3.2.4 Legal Implications

The images we are using are from the dataset created by us. Since, this is a dataset created with images of 13 people, their privacy has been taken care of by not making it public. Thus, the project does not infringe any law or regulation.

Chapter 4

About the Data

This section describes the dataset that is utilized for the project.

4.1 Creating the Dataset

The dataset consists of training and testing images. The training images consists of 3 images of 13 people each, without wearing the headsets. The testing images consists of 3 images of 13 people each, without wearing the HTC Vive headsets. The images are captured using Google Nexus 41. Hence, there are 39 training images and 39 testing images. The images are focused on the frontal image of the face. The images are then resized to 256*256.
There is a masked image to mask the headsets of size 256*256, as part of the dataset.

Training Images:

Figure 2: Training Images

Testing Images:

Figure 3: Testing Images

Chapter 5

About the Model

This section describes the generative adversarial network models that we have used as the network model. We first describe the architecture of the model in detail and then discuss the different experiments carried out in hopes of improving the accuracy.

5.1 Architectures

The various architectures considered are described in this section.

5.1.1 Basic Layout of GAN

Generative adversarial networks (GANs) are deep neural net architectures comprised of two nets, pitting one against the other thus are termed adversarial.

GANs were introduced in a paper by Ian Goodfellow and other researchers at the University of Montreal, in 2014. Referring to GANs, Facebook's AI research director Yann LeCun called adversarial training "the most interesting idea in the last 10 years in Machine Learning."

GANs' potential is huge, because they can learn to mimic any distribution of data. That is, GANs can be taught to create worlds eerily similar to our own in any domain: images, music, speech, prose. They are robot artists in a sense, and their output is impressive.

Generative algorithms are favored over discriminated algorithms for this problem. Discriminative algorithms try to classify input data; that is, given the features of an instance of data, they predict a label or category to which that data belongs.

So discriminative algorithms map features to labels. They are concerned solely with that correlation. One way to think about generative algorithms is that they do the opposite. Instead of predicting a label given certain features, they attempt to predict features given a certain label. The basic model used is a Generative Adversarial Learning (GAN).

Given a training set, this technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics. Typically, the generative network learns to map from a latent space to a data distribution of interest, while the discriminative network distinguishes candidates produced by the generator from the true data distribution. The generative network's training objective is to increase the error rate of the discriminative network i.e., "fool" the discriminator network by producing novel candidates that the discriminator thinks are not synthesized (are part of the true data distribution).

The generator is creating new, synthetic images that it passes to the discriminator. It does so in the hopes that they, too, will be deemed authentic, even though they are fake. The goal of the generator is to generate passable hand-written digits: to lie without being caught. The goal of the discriminator is to identify images coming from the generator as fake.

The simplest type of GAN consists of generator and discriminator. In this case the generator and the discriminator are just simple multi-layer perceptrons. GANs simply just seek to optimize the mathematical equation using stochastic gradient descent. The generator here takes in a noise vector ‘z’, usually 100-dimensional) and produce an image G(z), which is just a flattened vector of all the pixels in the image. This image is used in the equations we saw previously to simply update the weights of the generator and the discriminator by computing gradients through backpropagation. Here are the steps a GAN takes:

- The generator takes in random numbers and returns an image.
- This generated image is fed into the discriminator alongside a stream of images taken from the actual, ground-truth dataset.

- The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

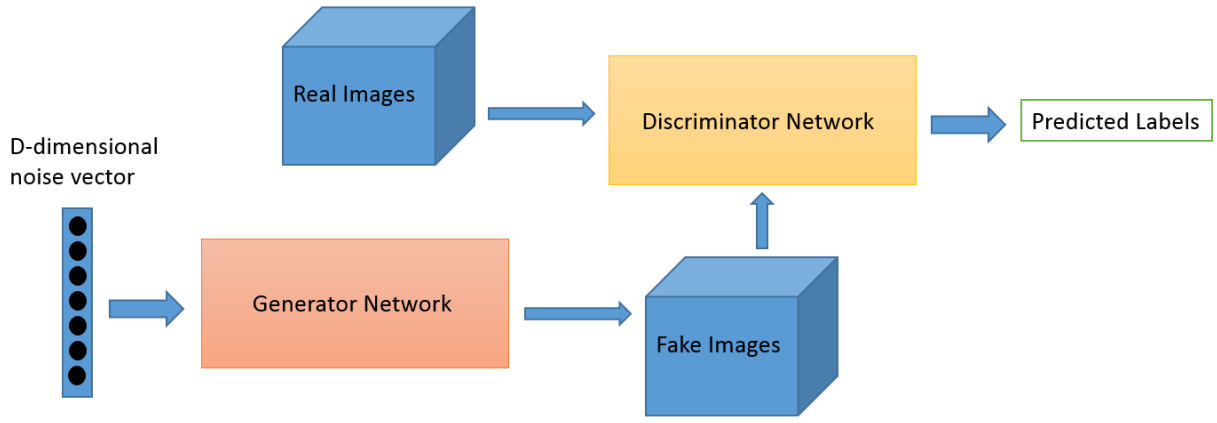

Figure 4: Basic Generative Adversarial Network Architecture

When the discriminator is trained, the generator values are constant; and when the generator is trained, the discriminator is constant. Each should train against a static adversary. For example, this gives the generator a better read on the gradient it must learn by.

Each side of the GAN can overpower the other. If the discriminator is too good, it will return values so close to 0 or 1 that the generator will struggle to read the gradient. If the generator is too good, it will persistently exploit weaknesses in the discriminator that lead to false negatives. This may be mitigated by the nets' respective learning rates.

A known dataset serves as the initial training data for the discriminator. Training it involves presenting it with samples from the training dataset, until it achieves acceptable accuracy. The generator trains based on whether it succeeds in fooling the discriminator. The basic components of every GAN are two neural networks - a generator that synthesizes new samples from scratch, and a discriminator that takes samples from both the training data and the generator's output and predicts if they are "real" or "fake". The generator input is a random vector (noise) and therefore its initial output is also noise. Over time, as it receives feedback from the discriminator, it learns

to synthesize more “realistic” images. The discriminator also improves over time by comparing generated samples with real samples, making it harder for the generator to deceive it.

1. We take some noise from random distribution, then we feed it to the Generator G to produce the fake x (label y=0) → (x,y) input-label pair.
2. We take this fake pair and the real pair x (label y =1) and feed it to the Discriminator D alternatively.
3. The discriminator D is a binary classification neural network so it calculates the loss for both fake x and real x and combine them as the final loss as D loss.
4. The generator G also calculates the loss from its noise as G loss since each network has a different objective function.
5. The two losses go back to their respective networks to learn from the loss (adjusting the parameters w r t the loss)
6. Apply any optimization algorithm (Grad descent, ADAM, RMS prop, etc..) Repeat this process for certain no of epochs or as long as you wish.

Each network has goals so these two networks pit against each other during the training.

The generator G gets stronger and stronger at generating the real type of results and the discriminator D also gets stronger and stronger at identifying which one is real, which one is fake.

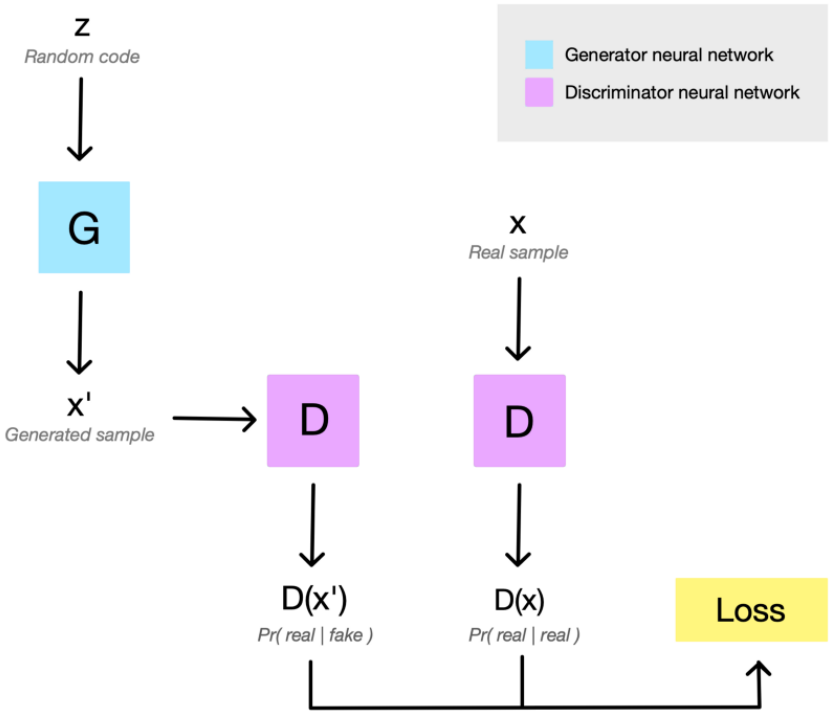

Figure 5: Comparison of Discriminative and Generative Neural Network

In a typical GAN, a random code z is fed into the generator G to produce a 'fake' sample. A discriminator network D is then (separately) fed both the generated sample x', and a real sample x from the training set. It assigns a probability of being 'real' to each, which depends on how convincing the fake is, and how sophisticated the discriminator has become. Both these probabilities are then used to compute the adversarial loss, from which we train both D and G via backpropagation. As the training iterations proceed, the discriminator is updated by ascending its stochastic gradient and the generator is updated by descending its stochastic gradient.

5.1.2 Edge-Connect Model

Along with the basic layout as described in the previous section, Edge-Connect divides the process into 2 stages. The two stages are edge generator and image completion. There is basic model of GAN utilized here. Here with the two steps as edge generator and image completion network, there are 2 GAN structures for each of the stage.

As in the considered base model, for the process of edge detection, the HED [16] is used to generate the sketch data, which corresponds to the user's input to modify the facial image. After that, the curves are smoothened and the small edges are erased. To create color domain data, blurred images are created by applying a median filtering with size 3 followed by 20 application of bilateral filter. GFC is used to segment the face, and each segmented parts were replaced with the median color of the corresponding parts. When creating data for the color domain, histogram equalization was not applied to avoid color contamination from light reflection and shadowing. Then, the curve is smoothened and the small objects are erased. Finally, the mask is multiplied, adopting a process similar to the previous free-form mask, and color images and get color brushed images.

The network architecture is based on encoder-decoder architecture like the U-net [13] and SN-patchGAN is used as the discrimination network. The network structure produces high-quality synthesis results with image size of 256×256 while achieving stable and fast training. The network also trains generator and discriminator simultaneously like the other networks. The generator receives incomplete images with user input to create an output image in the RGB channel, and inserts the masked area of the output image into the incomplete input image to create a complete image. The discriminator receives either a completed image or an original image (without masking) to determine whether the given input is real or fake.

5.1.3 Final Model

The final model works on the base model as GAN. In our network, as synonymous to the Edge-Connect model, there are two stages of edge generation and image completion network.

Edges are defined as an abrupt change in some low-level image feature such as brightness or color. Edge generation is the process of detecting the edges of the object in the image. The intensity changes can be detected by either finding the maxima or minima of the first derivative or finding zero crossings in the second derivative of the image. Gradient based detects edges by looking for

the maxima or minima of the first derivate of the image. Laplacian based model detects edges by searching for the zero crossing of the second derivate of the image.

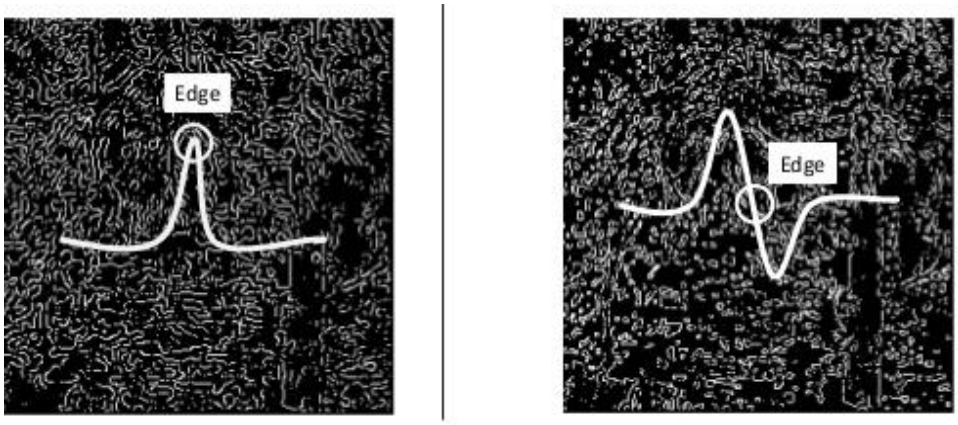

Figure 6: Gradient Based Detection (left) and Laplacian based Detection (right)

Canny Edge Generator is used for the process as used in the Edge-Connect. The process begins with the removal of noise using Gaussian filter. The first order derivatives of the image using operators like the Sobel operator. For every pixel, the non-maximal suppression is calculated and hysteresis thresholding is performed to get the edge model. The difference from the Edge-Connect model is that the process of edge filtering happens by determining conditional probability of a point being an Edge. The softmax function allows a probabilistic output, allowing multiple outputs to be generated as edge models.

The GAN network consists of a generator and discriminator. The generator and discriminator are separated by a structure termed as the self-attention structure of a convolutional neural network. The network trains the generator, self-attention layer and the discriminator simultaneously like the other networks. The generator receives incomplete images, images with the masked headsets as the user input to create an output image in the RGB channel, and inserts the masked area of the output image into the incomplete input image to create a coarse/incomplete image. This image is then passed through the self-attention layer.

The **self-attention layer** learns where to borrow or copy feature information from known background patches to generate missing patches. It is differentiable, thus can be trained in deep models, and fully-convolutional, which allows testing on arbitrary resolutions. We consider the

problem where we want to match features of missing pixels (foreground) to surroundings (background). We first extract patches (3 × 3) in background and reshape them as convolutional filters. To match the foreground patches f(x, y) with background ones b(x', y'), we measure with normalized inner product (cosine similarity) as below.

$$s_{x,y,x',y'} = \langle \frac{f_{x,y}}{||f_{x,y}||}, \frac{b_{x',y'}}{||b_{x',y'}||} \rangle$$

where s(x, y, x', y') represents similarity of patch centered in background (x', y') and foreground (x, y). Then we weigh the similarity with scaled softmax along (x', y') dimension to get attention score for each pixel s(x, y, x', y') described as s*(x, y, x', y') = softmax(λs(x, y, x', y')) where λ is a constant value. This is efficiently implemented as convolution and channel-wise softmax. Finally, we reuse extracted patches b(x', y') as deconvolutional filters to reconstruct foregrounds. Values of overlapped pixels are averaged. Firstly we use convolution to compute matching score of foreground patches with background patches (as convolutional filters). Then we apply softmax to compare and get attention score for each pixel. The softmax function The main advantage of using Softmax is the output probabilities range. The range will 0 to 1, and the sum of all the probabilities will be equal to one. Finally we reconstruct foreground patches with background patches by performing deconvolution on attention score. The contextual attention layer is differentiable and fully-convolutional. The structure of the self-attention layer is described below.

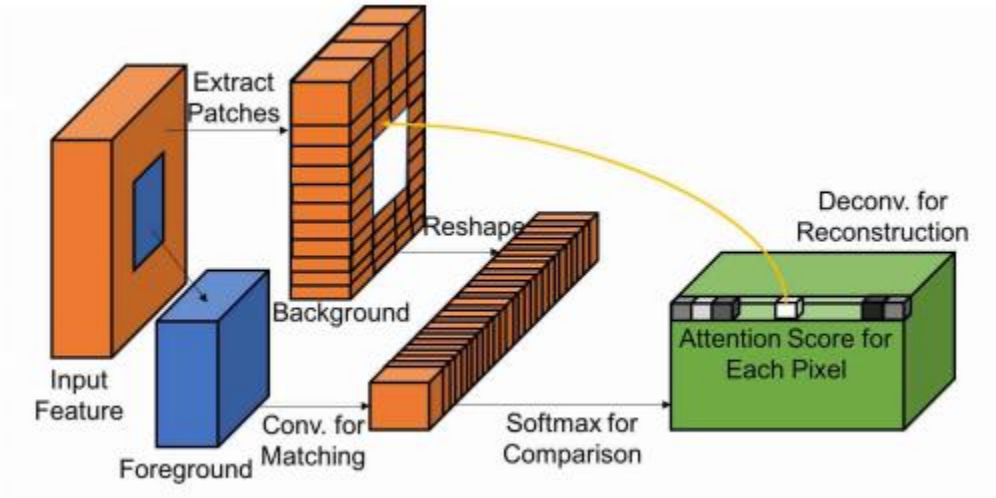

Figure 7: Self-Attention Layer

The discriminator then receives either a completed image or an original image (without masking) to determine whether the given input is real or fake. SN-PatchGAN is used as the network architecture of the discriminator here.

We use a combination of Mean Squared Error, L1 loss and general GAN loss is effective to restore large erased portions. In this case, the masked headset corresponds to a large portion of the missing in the image. The generator network takes an image with white pixels filled in the holes and a binary mask indicating the hole regions as input pairs, and outputs the final completed image as part of the input and output configurations. The input also has a mask with a corresponding binary mask to handle holes with variable sizes, shapes and locations. The input to the network is a 256 × 256 images sampled randomly during training, and the trained model can take an image of different sizes with multiple holes in it for testing.

Suitable training data is a very important factor for increasing the training performance of the network and increasing the responsiveness to user input. To train our model, we used the dataset we have created, after several pre-processing steps as described as following. We resize the images to 256×256 pixels before attaining the trained dataset. Since the headsets are focused to cover the eyes in the face image, we used a free-from mask based on the eye position to train network. Using free-form mask and face segmentation GFC as used in the base model, various options for the sketch image are generated. This is a crucial step which enabled the system to produce persuasive results for the user input case. However, when training on the facial images, a free draw mask is randomly applied with eye-positions as a starting point in order to express complex parts of the eyes.

As in the considered base model of Edge-Connect, for the process of edge detection, the HED [16] is used to generate the sketch data, which corresponds to the user's input to modify the facial image. After that, the curves are smoothened and the small edges are erased. To create color domain data, blurred images are created by applying a median filtering with size 3 followed by 20 application of bilateral filter. GFC is used to segment the face, and each segmented parts were replaced with the median color of the corresponding parts. When creating data for the color domain, histogram equalization was not applied to avoid color contamination from light reflection

and shadowing. Then, the curve is smoothened and the small objects are erased. Finally, the mask is multiplied, adopting a process similar to the previous free-form mask, and color images and get color brushed images.

5.1.3.1 Generator

The generator is based on U-net [10] and all convolution layers use gated convolutions, by using a using 3x3 size kernel like the self-attention layer. The generator differs from that of the Edge-Connect Model, by using dilated convolutions instead of a residual blocks model. Local signal normalization (LRN) is applied after feature map convolution layers excluding other soft gates. LRN is applied to all convolution layers except input and output layers. The encoder of our generator receives input tensor of size 256×256×9; an incomplete RGB channel image with a removed region to be edited, a binary sketch that describes the structure of removed parts, a RGB color stroke map, a binary mask and a noise. The noise is the gaussian noise that acts as the first layer. The encoder downsamples input 7 times using 2 stride kernel convolutions, followed by dilated convolutions before upsampling. The decoder uses transposed convolutions for upsampling. Then, skip connections were added to allow concatenation with previous layer with the same spatial resolution. We used the ELU activation function after each layer except for the output layer, which uses a tanh function.
Overall, the generator consists of 16 convolution layers and the output of the network is an RGB image of same size of input (256×256). We replaced the remaining parts of image outside the mask with the input image before applying the loss functions to it. This replacement allows the generator to be trained on the edited region exclusively. The generator is trained with losses as: MSE, L1 loss, per-pixel losses, perceptual loss, style loss and total variance loss. The generic GAN loss function is also used in the network.

5.1.3.2 Discriminator

The discriminator has SNPatchGAN [17] structure. Also we use 3×3 size convolution kernel and applied gradient penalty loss term. A convolutional neural network is used as the discriminator where the input consists of image, mask and guidance channels, and the output is a 3-D feature of

shape R(h×w×c), where h, w, c representing the height, width and number of channels respectively. Six strided convolutions with kernel size 5 and stride 2 is stacked to captures the feature statistics of Markovian patches. Below describes the discriminator.

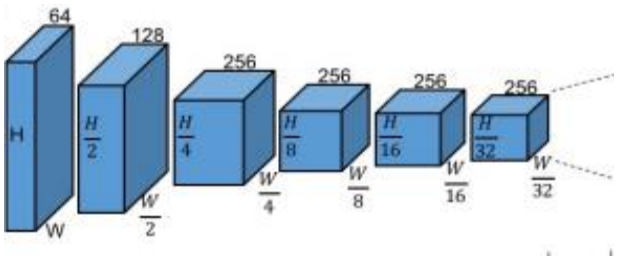

Figure 8: Structure of Discriminator

5.2 Experiments with Models

The training dataset consists of 39 images of people without wearing the virtual reality headsets (here HTC Vive glasses). The process begins with edge detection and edge generation. The edge generation is preceded with a probabilistic model, which helps in the generation of multiple images corresponding to one masked image. The model is able to generate photo-realistic results with a large fraction of image structures remaining intact. Furthermore, by including style loss, the inpainted images lack any “checkerboard” artifacts in the generated results. As importantly, the inpainted images exhibit minimal blurriness.

The generator uses a kernel of 3 and stride 2. The discriminator uses a kernel of 5 and stride 2. The model was initially trained with images of one person which resulted in blurred images. Training with 39 images betters the results, reducing the blurriness. Hence the more the training images, the better the network.

5.2.1 Tuning the Learning rate

The learning rate is probably the most important parameter in any network. It determines the rate at which the weights of the network are modified and is the key variable for the Gradient Descent Algorithm, which is the fundamental optimization algorithm. We used a momentum based approach. Momentum is a technique for accelerating Gradient Descent Algorithm. Using momentum, we can achieve faster learning rates and better our chances of converging to a global minimum. We also used a learning decay. It is the measure by which the learning rate will be reduced in successive epochs. This is done in order to ensure continuous learning even at later epochs. Otherwise, the learning will satiate after some epochs because the learning rate will become too high. Thus, it needs to be gradually reduced for optimal learning. 5.2.2 Tuning the dropout rate

To reduce overfitting, we include a dropout layer after each of the layer and the fully connected layers. The earmark of dropout layers is that there is only one parameter, the dropout rate is the probability with which neurons are dropped in the next layer.

5.2.2 Tuning the Number of Layers

The number of convolutional layers and fully connected layers in our basic model was decided to be 2 each. This was fixed at 2 because having more layers would create more parameters and thus exponentially increase the size of our model for every layer that was added. The main objective of this model was to achieve results close to the original image.

5.2.3 Batch Normalization

We used Batch Normalization as suggested in [19] to improve the basic model. Batch Normalization draws its strength from making normalization a part of the part of the model architecture and performing the normalization for each training mini-batch. Normalization is often used as a pre-processing step to make the data comparable across features. As the data flows

through a deep network, the weights and parameters adjust those values, sometimes making the data too big or too small again - a problem the authors of [19] refer as “internal covariate shift”. By normalizing the data in each mini-batch, this problem is largely avoided. Batch Normalization allows us to use much higher learning rates and to be less careful about initialization. It also acts as a regularizer.

5.2.4 Exponential Linear Units

Rectified Linear Units (ReLUs) have been the standard activations used in Neural Networks. Their performance has been proved empirically. However recently, many improvements to ReLUs have been suggested and we used the most recent and advanced one, which were Exponential Linear Units (ELUs).

In contrast to ReLUs, ELUs have negative values which allows them to push mean unit activations close to zero. Zero means speed up learning because they bring the gradient closer to the unit natural gradient. Like batch normalizations, ELUs push the mean towards zero, but with a significantly smaller computational footprint. ELUs saturate to a negative value with smaller inputs and thereby decrease the propagated variation and information. Consequently, dependencies between ELUs are much easier to model and distinct concepts are less likely to interfere.

5.2.5 Noise

In the model, we also tried adding a Gaussian Noise Layer as the first layer in order to add noise to the input data. Adding noise to the training input will help in making the model generalize to other inputs better.

5.2.6 Optimizers

The most common choice for optimizers for neural networks in the Stochastic Gradient Descent (SGD) Optimizer. We used the same optimizer for all the experiments. SGD requires us to manually tune the learning rate, decay and momentum. The methods in which they were tuned in Section 5.2.1 ‘Tuning the learning rate’.

Chapter 6

Experimental Results

6.1 Procedure

In this section, we present ablation studies with comparisons to recent related works, followed by face editing results. All the experiments were executed on NVIDIA(R) Tesla(R) V100 GPU and Power9 @ 2.3GHz CPU with Tensorflow v1.10, CUDA v8, Cudnn v7 and Python 3. For testing, it takes 54s on GPU for resolution 256×256 on average regardless of size and shape of inputs.

The training dataset consists of 39 images of people without wearing the virtual reality headsets (here HTC Vive glasses). The process begins with sketch development and color filling. The generator is a generative adversarial network and the discriminator a convolution neural network. Our model is able to generate photo-realistic results with a large fraction of image structures remaining intact. The model was initially trained with images of one person which resulted in blurred images. Training with 39 images betters the results, reducing the blurriness. Hence the more the training images, the better the network. As importantly, the inpainted images exhibit minimal blurriness.

The sketched models, that are formed by the Canny Edge-Detector are then passed through color generation layer. Histogram equalization is used to create sketches and HED is then used for smoothing and erasing the image for enhancing the quality of the output image. The Canny Edge detector has a probabilistic function to increase the number of outputs for a corresponding input image. Each segment is then replaced with the corresponding median color.

The testing results below indicate the masked image with the corresponding 5 outputs.

Figure 9: Test Results 1; Input (first image), outputs (unmasked images of first and second row), true image (third row)

The testing results show the original image and the 6 corresponding outputs after masking and inpainting processes.

Figure 10: Testing Results 2; Input (true image), outputs (unmasked images of second and third row)

Chapter 7

Conclusion & Future Work

7.1 Conclusion

We successfully developed a network with Generative Adversarial Network, which can remove the headsets from the images to enhance the experience of Virtual Reality. The advantage of this model is that there is no current method publicly available to solve the problem, hence this serves as an initial prototype for the problem using images. Once the images are aggregated together with a timer, the system has scope to be implemented in videos for virtual reality.

This technology enables viewing changes in facial expressions and emotions which enhances the communication through videos and video conferences. This has benefits for mental disorders and for turning VR into a viable short term treatment for pain that patients can use at home as an alternative to drugs. These benefits indicate the importance of this problem and the corresponding benefits.

7.2 Future Work

The project focuses on removal of headsets in images. This can further be enhanced to videos by aggregating the images to form the video and keeping track of time to implement the communication effectively. What we have developed is a prototype for the initialization of the project, which I hope will enhance and develop effectively to achieve the goal of real time headset removal.

While we have shown the potential of our technology, its applications extend beyond Mixed Reality. Removal of headsets in Headset removal is poised to enhance communication and social interaction in VR itself with diverse applications like VR video conference meetings, multiplayer

VR gaming, and exploration with friends and family. Going from an utterly blank headset to being able to see, with photographic realism, the faces of fellow VR users promises to be a significant transition in the VR world, and we are excited to be a part of it.

7.3 Personal Conclusion

At the outset, I would like to place on record my gratitude to Professor Ramon Morros, Professor Javier Ruiz Hidalgo and Professor Nuria Castell for giving me the opportunity to work on such an exciting and enriching project and to help me throughout the duration. This project was just the right difficulty for me, wherein, neither was it too easy being a walk nor too hard, seemingly impossible. It was challenging and piqued my curiosity to put my best efforts into it, since there was no available solution to the problem. Being my first foray into the domain of Image Processing and Inpainting, I learnt a lot thanks to this project, both academically and practically. The background knowledge that I had to acquire before beginning the project gave me a thorough theoretical exposure to generative adversarial networks and convolutional neural networks. Reading through so many research papers and articles in this domain, I obtained a very good awareness of the latest advances, best practices and conventions in the field. After starting the work on the project, I was exposed to the practical side of implementing adversarial networks and along with it, I was got a good experience of Image Processing and File handling in Python as well. Working with such a comprehensive codebase taught me how important modularity and writing clean code was, when building software. More importantly, the work that I did has inspired me to delve deeper into the domain and pursue higher studies in the same domain. Apart from these technical insights, I gained a lot of valuable non-technical skills too. Running experiments overnight and waiting for results, taught me how important patience and discipline were in research. Also, being an exchange student in Barcelona, I was able to live and relish a beautiful city with a great culture. The completion of this project, as a one-man team, in a new environment, in a new city has boosted my confidence and made me ready to take on bigger challenges. However, this would not have been possible without the constant support and expert guidance of Professor Ramon Morros and Professor Javier Ruiz Hidalgo, to whom I am eternally grateful. So, on the whole, this project has given me a wonderful experience which has improved my skills and character. I take with me a lot of knowledge and cherishable memories, as I move forward in life.

Chapter 8

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In OSDI, volume 16, pages 265–283, 2016. 6

[2] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. Stargan: Unified generative adversarial networks for multidomain image-to-image translation. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018. 3

[3] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2414–2423, 2016. 1

[4] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In Advances in Neural Information Processing Systems, pages 5767–5777, 2017.

[5] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. ACM Transactions on Graphics (TOG), 36(4):107, 2017. 1, 3, 4

[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. CVPR, 2017. 2

[7] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In European
Conference on Computer Vision, pages 694–711. Springer, 2016. 5

[8] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196, 2017. 3, 4

[9] Y. Li, S. Liu, J. Yang, and M.-H. Yang. Generative face completion. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 1, page 3, 2017. 1, 3, 4

[10] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. arXiv preprint arXiv:1804.07723, 2018. 4

[11] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957, 2018. 5

[12] T. Portenier, Q. Hu, A. Szabo, S. Bigdeli, P. Favaro, and M. Zwicker. Faceshop: Deep sketch-based face image editing. arXiv preprint arXiv:1804.08972, 2018. 2, 3, 4, 6, 7

[13] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015. 2, 3, 4

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3):211–252, 2015. 5

[15] M. Weber. Autotrace, 2018. http://autotrace.sourceforge.net. 3, 7

[16] S. ”Xie and Z. Tu. Holistically-nested edge detection. In Proceedings of IEEE International Conference on Computer Vision, 2015. 3

[17] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Free-form image inpainting with gated convolution. arXiv preprint arXiv:1806.03589, 2018. 2, 3, 4, 6, 7

[18] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. arXiv preprint arXiv:1801.07892, 2018. 3, 6, 7, 9

[19] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros. Real-time user-guided image colorization with learned deep priors. arXiv preprint arXiv:1705.02999, 2017. 2, 3

[20] Y. Zhao, B. Price, S. Cohen, and D. Gurari. Guided image inpainting: Replacing an image region by pulling content from another image. arXiv preprint arXiv:1803.08435, 2018. 2

[21] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired imageto-image translation using cycle-consistent adversarial networkss. In Computer Vision (ICCV), 2017 IEEE International Conference on, 2017. 2

[22] Nazeri, Kamyar and Ng, Eric and Joseph, Tony and Qureshi, Faisal and Ebrahimi, Mehran, EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning, arXiv preprint, 2019.

Appendix A: Requirements

The following tools are required for the project.

Software

- Python was chosen as the language of implementation as it is one of the best and most famous programming/ scripting languages, aptly suited for the project. Python enables us to use a highly modular framework with a huge library of packages which we can use for the deployment of the project.

  Python Packages

  - To get the data from the server we use Requests, which is a HTTP API built for python.
  - To process the data, we use PIL (Python Imaging Library).
  - To store the processed data as datasets, we use h5py.
  - To implement the generative adversarial network model, we use pytorch. PyTorch is a high-level neural networks library, written in Python. Scalable distributed training and performance optimization in research and production is enabled by the torch due to distributed backend. Deep integration into Python allows popular libraries and packages to be used for easily writing neural network layers in Python. A rich ecosystem of tools and libraries extends PyTorch and supports development in computer vision, NLP and more.

- We also use **Git**, the best way for project management and version control. Github gives us a git repository on the cloud, which can be used for synchronization of different files and security of the project.

- **PyCharm**, a Python IDE developed by JetBrains is used for the working of the project. It has automatic synchronization with Git and helpful features like code completion, automatic error checking, suggestions, package control, logging etc.

Hardware

- **Laptop**, our primary workstation. This is bundled with basic software (Ubuntu 16.04, Google Chrome, Libre OpenOffice, Sublime Text Editor) and internet connection.
- **HTC Vive**, The HTC Vive is a virtual reality headset developed by HTC and Valve Corporation. The headset uses "room scale" tracking technology, allowing the user to move in 3D space and use motion-tracked handheld controllers to interact with the environment. They are used for creating our dataset.

Appendix B: Background Knowledge

Generative Adversarial Networks

Generative adversarial networks (GANs) are deep neural net architectures comprised of two nets, pitting one against the other thus the “adversarial”.

GANs were introduced in a paper by Ian Goodfellow and other researchers at the University of Montreal, in 2014. Referring to GANs, Facebook’s AI research director Yann LeCun called adversarial training “the most interesting idea in the last 10 years in Machine Learning.”

GANs’ potential is huge, because they can learn to mimic any distribution of data. That is, GANs can be taught to create worlds eerily similar to our own in any domain: images, music, speech, prose. They are robot artists in a sense, and their output is impressive.
Generative algorithms are favored over discriminated algorithms for this problem. Discriminative algorithms try to classify input data; that is, given the features of an instance of data, they predict a label or category to which that data belongs.

So discriminative algorithms map features to labels. They are concerned solely with that correlation. One way to think about generative algorithms is that they do the opposite. Instead of predicting a label given certain features, they attempt to predict features given a certain label. The basic model used is a Generative Adversarial Learning (GAN). Given a training set, this technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics. Typically, the generative network learns to map from a latent space to a data distribution of interest, while the discriminative network distinguishes candidates produced by the generator from the true data distribution. The generative network's training objective is to increase the error rate of the discriminative network i.e., "fool" the

discriminator network by producing novel candidates that the discriminator thinks are not synthesized (are part of the true data distribution).

The generator is creating new, synthetic images that it passes to the discriminator. It does so in the hopes that they, too, will be deemed authentic, even though they are fake. The goal of the generator is to generate passable hand-written digits: to lie without being caught. The goal of the discriminator is to identify images coming from the generator as fake.

Convolutional Neural Networks

Convolutional networks (LeCun, 1989), also known as convolutional neural networks or CNNs, are a specialized kind of neural network for processing data that has a known, grid-like topology. Examples include time-series data, which can be thought of as a 1D grid taking samples at regular intervals, and image data, which can be thought of as a 2D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simple neural networks that use convolution in place of general matrix multiplication in atleast one of their layers.

Convolution leverages three important ideas that can help improve a machine learning system: sparse interactions, parameter sharing and equivariant representations.

- Traditional neural network layers use matrix multiplication by a matrix of parameters with a separate parameter describing the interaction between each

input and each output unit. This means every output unit interacts with every input unit. Convolutional networks, however, typically have sparse interactions (also referred as sparse connectivity or sparse weights). This is accomplished by making the kernel smaller than the input.

- Parameter sharing refers to using the same parameter for more than one function in a model. In a traditional neural net, each element of the weight matrix is used exactly once when computing the output layer. It is multiplied by one element of the input and then never revisited. As a synonym for parameter sharing, one can say that a network has tied weights, because the value of the weight applied to one input is tied to the value of a weight applied elsewhere. In a convolutional neural net, each member of the kernel is used at every position of the input.
- A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.