Cairo University
Faculty of Engineering
Department of Computer Engineering

Replica

A Graduation Project Report Submitted
to

Faculty of Engineering, Cairo University

in Partial Fulfillment of the requirements of the degree

of

Bachelor of Science in Computer Engineering.

Presented by

Radwa Samy Khattab
Salma Ibrahim Ismail
Mariam Hesham Heiba
Nehal Abdelkader Bayoumi

Supervised by

Prof. Nevin Darwish

26/07/2021

Abstract

Deep fake is one of the rapidly growing technologies, due to the growth in the hardware resources and the advances of GPUs architectures. Deep fake is used for different purposes like motion transfer. Motion transfer can improve the production of games, movies, and the entertainment industry. Currently, most applications that use these technologies either work on images, or simply capture the motion of a person in a video to be applied on a 3D model on animation platforms like Unity.

Replica is a project that uses deep fake technology in motion transfer that could be applied to videos directly. Given two videos of two persons or their 3D models, the motion of one person (source) is captured to produce another video replicating this motion to be performed by the second person (target), producing a new manipulated video as output.

This process is done in a way to allow the generation of multiple manipulated videos of the same target given multiple sources, without the need to go through the whole workflow again. The normal workflow of the process includes technologies such as image segmentation, pose estimation, and generative adversarial networks (GAN). The first version of Replica has been successfully implemented and tested. Results are promising and competitive to similar projects.

الملخص

التزييف العميق احد التقنيات التي نمت بشكل كبير مؤخرا نتيجة لتطور معدات الحاسوب والتقدم فى وحدة معالجة الرسوميات (جي بي يو). تقنية التزييف العميق تستخدم لعدة أغراض مثل نقل الحركة. نقل الحركة يساعد على التحسين من انتاج الالعاب الالكترونية والافلام وصناعة الترفيه. حاليا معظم التطبيقات التي تستخدم هذه التقنيات تعمل على الصور لالتقاط حركة الإنسان في الفيديو ليتم تطبيقها على نموذج ثلاثي الأبعاد على منصات الرسوم المتحركة مثل يونيتي.

ريبليكا مشروع يستخدم تقنية التزييف العميق في نقل الحركة لتطبيقها على الفيديو مباشرة. باستخدام فيديو من المستخدم وفيديو آخر الذي يستخدم لنقل الحركة منه, سيتم التقاط الحركة من الفيديو ليتم نقلها الى الفيديو الخاص بالمستخدم.

العملية تتم بطريقة تسمح بإنشاء عدة مقاطع فيديو مزيفة لنفس المستخدم باستخدام مقاطع فيديو أخرى دون الحاجة الى اعادة جميع الخطوات مرة أخرى. سير العملية يتضمن تقنيات تقطيع الصورة واستخراج الوضع واستخدام تقنية "ج.ا.ن". تم تنفيذ نسخة اولية من ريبليكا و تم تجربتها بنجاح, و النتائج كانت واعدة و منافسة للمنتجات المشابهة.

ACKNOWLEDGMENT

First and foremost, we'd like to thank our supervisor Prof. Nevin Darwish for her incredible support, mentorship, interest and patience. We would also like to thank Innolabs for the technical support. Last but not least, thank you to our family and friends for being supportive of us throughout this journey.

Table of Contents

List of Figures

List of Tables

List of Abbreviation

AR	Augmented Reality
BCE	Binary Cross Entropy
cGAN	Conditional GAN
Conv2D	2D Convolution Layer
CovNet	Convolution Networks
CNN	Convolution Neural Networks
DL	Deep Learning
GAN	Generative Adversarial Networks
MAE	Mean Absolute Error
NN	Neural Networks
SRGAN	Super Resolution GAN

List of Symbols

e	Exponential Term
σ	Standard Deviation
f	flow threshold

Contacts

Team Members

Name	Email	Phone Number
Radwa Samy Khattab	RadwaSM@hotmail.com	+2 01028256013
Salma Ibrahim Ismail	SalmaIbraheem44@gmail.com	+2 01016721744
Mariam Hesham Heiba	MariamHeiba@gmail.com	+2 01026632117
Nehal AbdelKader Bayoumi	NehalAbdulkader@gmail.com	+2 01142890015

Supervisor

Name	Email	Number
Prof. Nevin Darwish	ndarwish@eng.cu.edu.eg	+2 01222247364

This page is left intentionally empty

1 Chapter 1: Introduction

Entertainment business has become one of the largest business industries worldwide. With the advances in technology, comes advances in the entertainment business. More specifically, the advances in GPU's and hardware, which lead to better performances, better accuracy, more parallelism and faster processing.

Deep Fake is one of the technologies that advanced due to these reasons. It is a way of manipulating data in a way to produce a fake media in a realistic way, which could fool the viewer, but better, could be used in entertainment applications.

Deep Fake helped advance old methods of motion capture into motion transfer. Instead of capturing motion of a moving person, either using a camera only or using sensors, and then applying this motion onto a pre-made 3D model on one of various platforms, like Unity. We now could transfer the motion from one video to the other. This is the main technology behind Replica.

Replica is a platform on which you could upload a video of yourself making some random motion, and then choose one of our multiple videos on the platform, which includes different dances from different sources. Using this chosen video and your video, you would result with a video of yourself making the dance moves of the other video you chose, which you can share with your friends or on social media applications for entertainment.

1.1 Motivation and Justification

We all know how big the entertainment industry is these days, and getting even bigger. There are movies, games, social media applications, and much more. In 2020, the United States entertainment and media market was estimated to be worth 720 billion dollars. This is just one country. Movies alone are worth $136 billion as estimated in 2018. Social media is worth $94.83 billion in 2020, and estimated to be $102.62 billion in 2021.

Many of these entertainment markets need to do some process of manipulation of media, for one purpose or the other. For example, in movies they need a stunt double to make hard scenes instead of the real actor or actress. They use motion capture methods to act in an animation movie, and apply the captured motion from the actor to a

3D model on computers instead. They use many hard and pricey methods for the sake of producing a good realistic movie, even if all done by animations.

In social media, there are many applications that are based mainly on manipulation of data. There is B612 and capturing applications that apply some edits on your face while taking the photo, like applying fake make-up, or a sticker, or giving you bear ears…etc. There are applications like ReFace that take a photo of you and capture your facial features then apply these features on faces of other actors in other videos you choose, which makes it look like you are the one acting in the movie.

There are many examples for how media manipulations could improve the entertainment industry, or produce a new product that would be appealing to teenagers and social-media users.

In Replica, we wanted to produce a new way of using technology to produce a fun and entertaining output, but also that once is complicated and accurate enough, could be used in movies to replace the need of using a stunt duple or capturing the motion of an actor and applying it on someone else on a computer. In Replica, we would make these processes much easier, when you could choose the motion you need to transfer onto your video, and this motion could be done by someone else before, or simply made by a stick-man on a simple media application, without the need of hard processing of moving a complicated model in each frame.

1.2 Project Objectives and Problem Definition

Motion Transfer needs a complicated and accurate process to be done right. Our problem relies on transferring this motion accurately from the source video—dancer video for example—to the target (user) video, in a realistic enough way.

Starting with two videos, both taken by a single camera video, one is for the user and the other is the source. Using the source video, we need to transfer its motion to the target subject, with all its details, considering outfit, background, body shape and proportions.

1.3 Project Outcomes

The project outcome is a platform that lets any user upload a video of himself making any random motion, and then using one of various available source videos on the

platform and some processing, the output video is a manipulated video of the user making the moves of the chosen source video.

1.4 Document Organization

The document will be organized in 6 chapters along with the appendices, the first chapter is an introduction to the project showing us the motivation for the project and the expected outcome, then in chapter 2 we explain the market need for this kind of solution, the targeted customers along with a detailed market survey on the types of competitors in this market. The last detail is the financial analysis of our project along with the expected stream of revenues we can think of.

The next chapter goes through the literature survey of all the tried methods, most of them are academically published papers as the methods used by private companies are not disclosed out to the public. This then goes through all the necessary background needed to understand the rest of the report. After explaining the background, chapter 4 explains the system design and architecture, first laying out the block diagram of the solution then going in detail through every module, explaining the concepts behind the module and the relevant implementation details.

Chapter 5 explains the testing done to the system and the integration along with any comparative work to previously known solutions. Finally, chapter 6 goes through the conclusion, the challenges we faced during the implementation, our gained experience through this whole process and the future work to create an even better product.

2 Chapter 2: Market Feasibility Study

Motion Transfer is a market that has been massively growing in the past couple of years and is going to continue growing in the following years along with the advances of AR applications and the need to ease the production of movies, animation movies, games and even for entertainment. We can all notice how big social applications can get, and the amount companies can profit from entertainment services, like the Reface app.

Currently, the market is relatively new, and papers covering this topic are recent. However, with the advances in technology, more specifically in GPU's which can ease the building of motion transfer projects, the field would only get more accurate and better, and there would be more demand for it.

2.1 Target Customers

Our target customers are movie makers, games developers and users. To users, we will provide our tools to them in a software as a service platform. Considering that motion transfer can be applied to humans or 3D models, we can produce a new animated video with the targeted motion.

Movie makers can benefit from it in generating motion for animated characters or for generating a specific difficult motion in movies that will need the help of a stunt double of an actor or an actress.
In game development, we can use it in the motion of characters and achieving the most accurate reaction for each movement of the characters is the most expensive part in the development, along with the environment of the game itself. We may help reduce this cost by producing the motion ourselves that can be easily added to the model in the game environment. Since game development is a massively growing field and becoming more challenging specially for the indie game developers, this may increase our customers.
Also, as for users, we target their usage in the entertainment field, such that we will offer the service on a platform, on which any user can use it with a variety of options for the source video.

Currently, our initial customers would be the users and the entertainment field. For future work, once our project is sophisticated enough, we can include other customers in game development with a subscription or project-based plan.

2.2 Market Survey

While we present a Replica application, we face multiple competitors in similar fields. Motion transfer is a widely-growing market. Many existing apps have different approaches to accomplish it in various ways. We show similar products to our app in the following sections, mentioning the pros and cons of each.

2.2.1 Example on Motion Transfer on a Person

One of our main competitors is the “Do as I Do: AI Puppet-Master” application. It was created only recently, in December 2020, by Lugo Okeke. It is an entertainment iOS application which applies motion transfer to create a manipulated video of the user mimicking the motion in one of the sample videos provided.

Figure 2.1: Example of the input image, a capture from the source video in the middle, and the equivalent capture of the target after motion transferred

One of the advantages of “Do As I Do”, is that it only requires an input image of the user where his full body is visible in the image. However, the image should be taken somewhere without much noise or shadows, and even if, as you can still see in figure 2.1, the accuracy of the result is not that good, and you can see that it is fake.
Another main downside to the app is its cost, which is 3$ a week to use the app, and it is only available on iOS.

2.2.2 Example on Motion Capture to a Model

Other applications, like Radical [1], sample output shown in figure 2.2, process the user—target—video to capture motion, then apply the human motion on the pre-made 3D model. Radical offers uploading video that includes the source motion to their cloud to be processed and extract the motion into the FBX file, which can be later added to any 3D model you have on Blender, Unity or any of the games' development environments.

Figure 2.2: A shot taken from Radical demo, showing the model mimicking the person's movement

One of the advantages of Radical is that the resulting motion of the model could be more accurate than that of a video of another person. This is due to the fact that there is no generation of fake frames. The motion is captured and this motion animation is transferred to a model. On the other hand, the limitations of Radical is that it can only capture the motion and extract it to an FBX file. So, it can not be applied directly to a human being, or produce a video. In order to use Radical, one must have a good understanding of dealing with models, and their animations and how to apply one. Although they offer the service, it does not come for free. Their prices depend on the length of the video you upload.

2.2.3 Example on Motion Capture in AR apps

There is more than one product that offers similar service that is used in AR applications, for entertainment mainly. One of these applications is DeepMotion [2], sample output is in figure 2.3. DeepMotion provides multiple services. The closest one to our service is applying it on an animated model in an AR application, such that it can animate any person walking into an avatar. Of course, this application too depends more on motion capture and applying it on an avatar but in real-time, which is an advance to Radical.

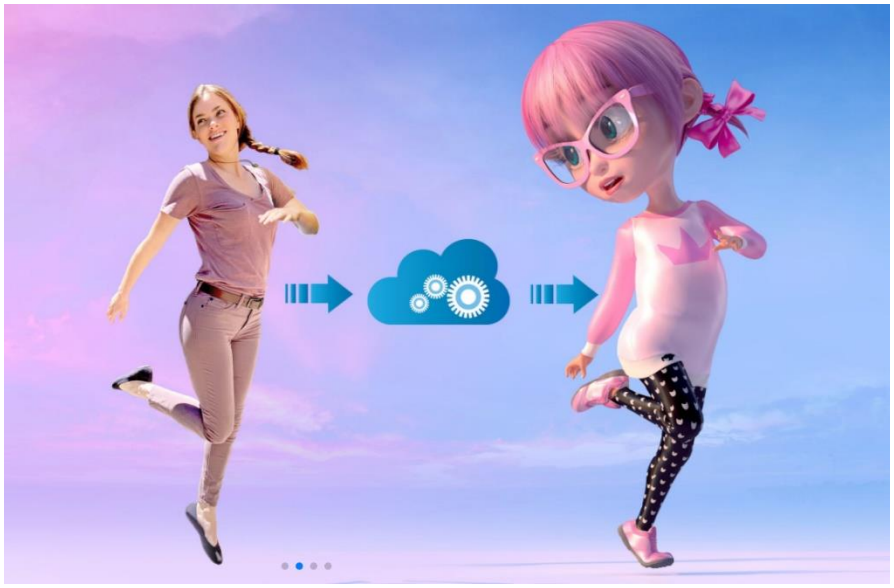

Figure 2.3: Example on what DeepMotion AR app does

The advantage of this application is working in real-time, and the entertainment in seeing the motion on an avatar similar to the user or from his choosing. However, again, this product is limited to applying it on an avatar and is mainly for entertainment, so it can not be used by all of our potential users. Also, the application is focused on motion capture rather than motion transfer, then applies this motion to a pre-made avatar they have on their cloud.

2.3 Business Case and Financial Analysis

2.3.1 Business Case

Based on our previous market analysis, we are going to offer an initial free trial for each new user, such that the user can upload one video of himself (target video) and can choose a number of source videos from a limited available options (for example 5 free source videos) on the website to have the motion transferred from either of them to his target video. After this free trial, if the user wants to buy another source video that is not free, each source video would cost 50 cents only. If the user wants to change the target video to another one, he can upload another video for three dollars. Prices will be reconsidered after one year and then on bi-yearly basis, based on analysis of our customers. Promotions are also planned at times of holidays. Incentives will also be offered so that the source video cost could be less around 30% if the user would buy more than one. The target at this point is 10 videos at a time in order to achieve our first million users in five years.

2.3.2 Financial Analysis

The financial analysis for our project based on the previous business case is presented in table 2.1 below. Shown in the table are the Capex which are the one-time spending such as the costs of purchasing computers for our employees, which would be in the first year, and then again after two years when we increase our number of employees from four to ten. The table also includes the Opex which are the recurring payments such as salaries for employees, rent and website hosting costs.

It is expected that the advertising would help in making the application more popular and used, but more importantly the propaganda on social media, as most of the well-known applications now started by getting known among teengers and social media users. From the statistics on similar applications on deep fake released in the past couple of years, as the ones mentioned in the marketing survey, and our plan for advertisement on different social media platforms, and our local distribution of the application, we can have a minimum of 500 users by the first year. Suppose that in the first year, the 500 users for the application each paid $20 on the application, then the total subscription profit in 1st year could be $1000. As years go by, more users would subscribe and more profit could be made. In addition, ads profit. On the other hand, the cash out would include the cost of the cloud server that would be used in training of GAN, and the website host to put it online. Also, there are the salaries of the employees and office rent and bills. Lastly, the cost of the advertising on other websites. More details are shown in table 2.1.

Table 2.1: Financial Analysis of Replica

Measure	Year 1	Year 2	Year 3	Year 4	Year 5
Computers	$4,000		$6,000		
Website Host & Cloud Server	$500	$1,000	$1,500	$2,000	$3,000
Salaries	$24,000	$24,00	$60,000	$60,000	$60,000
Office Rent & Bills	$2,400	$2,400	$6,000	$6,000	$6,000
Advertising	$120	$120	$150	$150	$200
Total Cash Out	$31,020	$27,520	$67,650	$68,150	$69,200
Subscriptions Profit	$1,000	$5,000	$30,000	$70,000	$100,000
Website Ads Profit	$3,000	$10,000	$30,000	$50,000	$50,000
Total Cash In	$4,000	$15,000	$60,000	$120,000	$150,000
Revenue	- $27,020	-$12,520	-$7,000	$51,850	$80,800

3 Chapter 3: Literature Study

This chapter discusses the main topics that are of interest for our project. Each section will be dedicated to detail one of these topics along with reviewing the research papers that have been published in relation to the topic. Next, a detailed comparison between different approaches is given in support of building the complete project. Finally, the chapter ends by discussing the specific approach that has been chosen for implementation along with the reasons for its choice.

3.1 Background on Segmentation

Video object segmentation is a wide area of research as it aims to mimic the properties of Human visual system. These properties allow humans to determine the object of interest on their visual sight and give attention to it by tracking it over time. The rest of the details in the scene that do not belong to the object of interest goes in the background. A lot of techniques have been presented for segmentation using a classical perspective using image processing techniques or a deep learning perspective.

One of the popular classical techniques is using conditional random fields to accomplish video segmentation. [3] used a graph-based superpixels' segmentation to perform a classification technique in each pixel based on its feature, it would be classified as a foreground pixel taking label 0 or a background pixel taking label 1. They get this done in four main steps, first step is to extract 8 features that can represent each pixel using CIELAB color space representation ,spatial coordinate, the apparent motion of the backward/forward optical flow and spatiotemporal saliency then in the second step they propose a super-pixel segmentation as each frame is segmented multiple times with different numbers of super-pixels by varying input parameters, then merging this segmentation into a single over-segmentation by solving the graph using spectral clustering. Then in the third step using a foreground separation model that uses super-pixel segmentation from the previous step, and features from the first step to obtain an initial segmentation map for each frame of the input video to be used to calculate the likelihood of a pixel belonging to a particular class. At the final step they construct and solve a conditional random field to get the final segmentation results using energy minimization to obtain a map comprising segmented primary objects.

Paper [4] proposes two methods for detection and segmentation of moving objects in videos. First method is for object detection using background subtraction where the object is determined by taking the difference between the background image and input image then using morphological operations to remove noise regions. The close

operation eliminates background noise and the open operation eliminates noise in the object region itself. After the morphological operations there are still holes in the interior of the object and to solve this problem, the area enclosed by the boundary is checked. If the area is greater than 40% of the total area then the algorithm will merge this area with the total enclosed area by the boundary. Second method for segmentation using two approaches i.e thresholding and edge detection. At first the image is converted into binary images and then thresholding and edge detection is performed to segment the object. In thresholding, pixels are assigned to categories based on the range of values in which a pixel lies. In edge-based segmentation, an edge filter is applied to the input image, pixels are labeled as edge or non-edge based on the filter output and pixels which are not separated by an edge are assigned to the same category.

In recent years, they were focusing on making deep learning solve segmentation problems in terms of extracting features of each scene by the network using convolution neural networks that include the temporal changes in video. The easiest way to include the temporal information is to concatenate multiple frames into a single input for this network [5]. They succeeded in doing that with two branches in the network: the temporal coherence branch which is designed to capture the dynamic appearance and motion cues of video sequences and the spatial segmentation branch which segments objects based on the learned appearance and motion cues.

3.2 Background on Pose Estimation

Pose estimation is an important part of our project. In pose estimation, we want to know the pose of a person—his limbs locations and angles—given an image of the person.
You may have seen this technology used in XBox games or Wii, in which the device automatically captures your poses and tracks your movements so you can control your character in the game without the use of a joystick.

There are multiple approaches to perform pose estimation, one approach is a classical approach without the use of deep learning. The other approach uses machine learning or deep learning models, given a proper dataset. However, these approaches are more appropriate if you need to classify your poses to one of some poses you have. For example, you need to know if the person is standing, sitting, or jumping, without tracking every movement they make.

In [6], the authors were concerned with human pose estimation and tracking using Markov Chain Dynamics. In this paper they propose an efficient approach to human pose estimation in static images and human pose tracking in video sequences. They

present the human body as a three-level tree structure and the estimation process is formulated as a Bayesian inference problem.

They designed two Markov Chain Dynamics; which are diffusion and jump. These two respectively correspond to a local search operation, and a switch to a new local optimization process. Both will be explained more further on.

They started their approach by creating a human model that represents a tree composed of 10 parts as shown in figure 3.1 (a). Where they made the torso the root of the tree, and then the limbs connected to it as its children, which formulated a tree as in figure 3.1 (b). Each of these nodes/parts are represented by their x and y coordinates, and their orientation relative to their parent as shown in figure 3.1 (c).

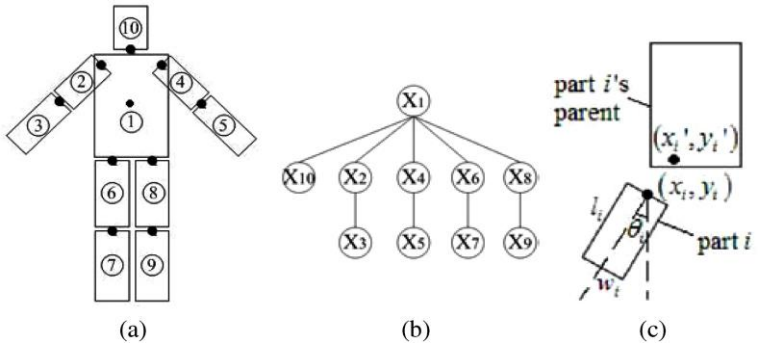

Figure 3.2: (a) Body Model, (b) Body Tree, (c) Body part orientation

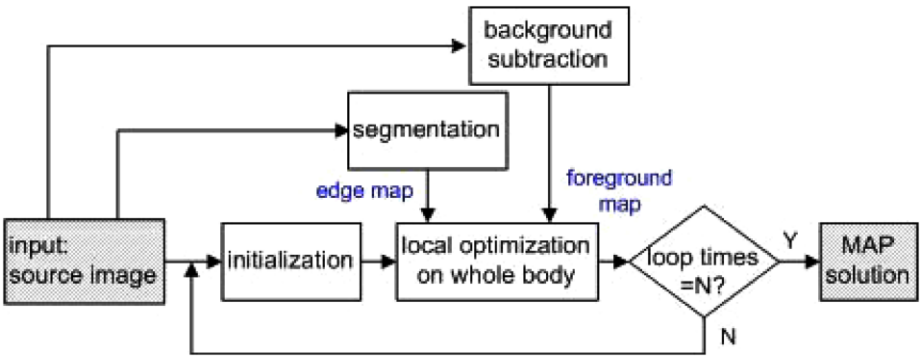

Figure 3.1: Flow chart of the loop iterations

As shown in figure 3.2, given a source image, they assume their input image is already segmented to a foreground image having the person in it in white and the background in black. They get the edge map image of this foreground image. The algorithm works iteratively, starting with initial values for the body parts using prior information of body constraints—like body proportions. After the initial values, in each iteration, a local

optimization is done on the parsed human body parts to find the best possible fit for the given body configuration in such a way to achieve a local maximum in the logically-right joints intersection.

During these iterations, their algorithm chooses which body part to modify depending on an importance equation, taking the body part with the highest importance. If a body part's importance is less than some threshold, they jump to another body part. This iterative process is repeated N times till the best result is reached.

In [7] pose estimation is done using deep learning approaches dependent on capturing information at every scale, with the help of a special type of Fully Convolutional Network called hourglass structure. Hourglass is a simple and minimal design that can capture all features, then takes this feature matrix and combines it with earlier layers which have a higher spatial understanding than the feature matrix, this allows to understand a lot about the input, then generates a heat map of where the joints are.

The hourglass structure consist of two sections; encoding and decoding, each section has four cube, each cube is a bottleneck layer then use the pooling layer,but with different first layer as it consist of 7x7 convolution layer while other cubes consists of 3x3 convolution layer.

At the last layer in the decoder we have the highest feature info and the lowest spatial info, this is passed to the encoder that consists of another four cubes, and starts resampling. At the end of the encoder the output is passed through a convolution layer then duplicates the layer to produce a set of heatmaps. Then perform an element-wise addition between the inputs of the network, this will give us the required output.

3.3 Background on Generative Adversarial Networks

As explained in [8], Generative Adversarial Networks (GAN) consist of a generative model G that captures data distribution and a discriminative model D that estimates the probability that a sample came from the training data rather than the generator. The generator has no direct access to the real data it learns by its interaction with the discriminator. The discriminator has access to the real data and the synthetic samples. An input noise variable is used to learn the generator's distribution over data x and uses the discriminator error signal to generate better quality forgeries.

In training GANs, it is important that both models should improve together and should be kept at similar skill levels from the beginning of training the reason behind this is if we had a discriminator that is superior than the generator we'll get predictions from it telling us that all the fake examples are 100% fake which is not useful for the generator as it does not know in which direction to go to improve meanwhile if we had a superior generator the predictions will indicate that the generated data are a 100% real which will lead the generator to map too many values of the noise vector to the same value of x.

One of the main improvements in the training of GANs for generating images was the DCGAN architecture offered by [9] it proposes a set of instructions to overcome the instability of training GANs. The first is using the LeakyRELU activation function in the discriminator to achieve superior performance over using ReLU. Second is Batch Normalization in both the generator and discriminator, which stabilizes learning and helps prevent the generator from collapsing all samples to a single point. Third is replacing pooling layers with strid-ed convolutions and fractionally strid-ed convolutions which lets both the generator and the discriminator learn good up-sampling and down-sampling which may help in enhancing the quality of image synthesis.

There are different types of GAN's that are used for different purposes. For example, there is conditional GAN (cGAN) that is used to generate a fake image with specific conditions, like if we are generating fake frames of a person's face, we could specify that the person wears glasses, or has long hair…etc. Another example of GAN is used to improve the resolution of an image. This GAN is called super resolution GAN (SRGAN) [10]. In other words, it could convert a 1024x512 image to 2048x1024. This technology is used in recent GPUs by Nvidia to improve the frames of a game on a computer or playstation, without needing to have a large resolution frame converted through the internet, which made games run faster on the internet with better resolution and graphics. Another example of GAN is image-to-image translation, which we used in Replica, to convert one image to another.

GAN models are based on many concepts that must be understood in order to make the generator and discriminator models. The discriminator model is basically a deep learning model, which relies on many concepts as in [11], which are explained in the following sections.

3.3.1 Neural Network Architecture

Any neural network consists of a large number of artificial neurons or units arranged in a group of layers as shown in figure 3.3.

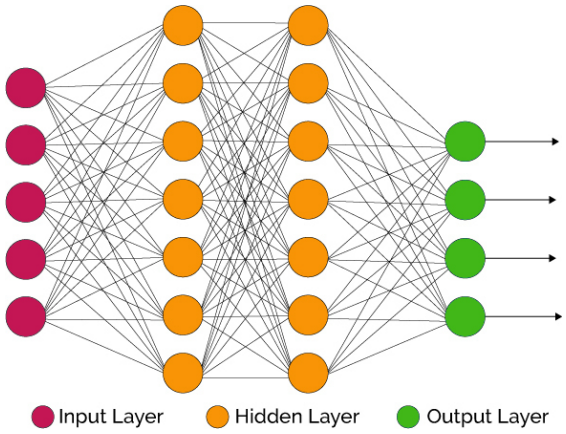

Figure 3.3: Sample of Artificial Neural Network Architecture

In figure 3.3, it is denoted the first layer of neurons with pink color, which represents the input layer. The input layer is responsible for taking the input image or tokens, depending on the project at hand. For example, when working on an image, the input layer consists of neutrons equal to the number of pixels in the image. In the hidden layers in the middle, these layers are responsible for the transformation that happens from the input to a meaningful output, whether it is a classification or a predicted value or probability. The output layer consists of neurons having the meaningful output we need. If the network makes a binary classification, then only one neuron is needed, resulting in 0 or 1.

3.3.2 Activation Functions

Activation functions are functions applied on the output of each neuron in a NN, as shown in figure 3.4. The value of any neuron in the NN is a value ranging from -infinity to infinity, given the function:

$$Y = \sum(weight * input) + bias$$

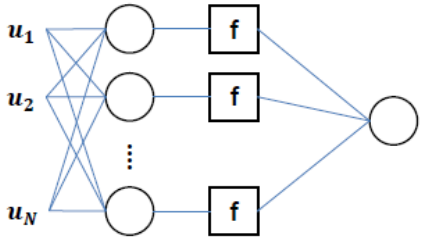

Figure 3.4: Figure showing where activation functions are located in the NN

The activation function maps this input to a different range depending on the used activation function. Some of the most popular ones are:

1. Tanh Function
   The Tanh function, shown in figure 3.5, is represented with the equation:

$$f(X) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

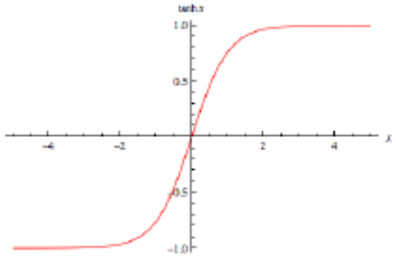

Figure 3.5: Tanh Function

2. ReLu Function
   The ReLu function, shown in figure 3.6, is represented with the equation:

$$f(X) = \max(x, 0)$$

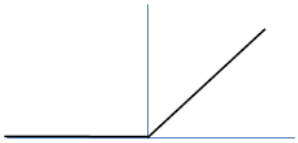

Figure 3.6: ReLu Function

3. Leaky ReLu Function
   The Leaky ReLu function, shown in figure 3.7, is represented with the equation:

$$f(X) = \begin{cases} x & if\ x > 0 \\ ax & otherwise \end{cases} \qquad where\ 0 < a < 1$$

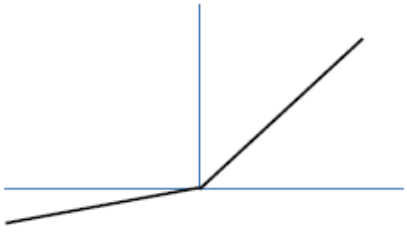

Figure 3.7: Leaky ReLu function

4. Sigmoid Function
   Sigmoid function, shown in figure 3.8, is represented with the equation:

$$f(X) = \frac{1}{1 + e^{-x}}$$

Figure 3.8: Sigmoid Function

The choice of these activation functions depend on the purpose of the NN. For example, Sigmoid is used in the output layer for binary classification, giving the probability of a classification whether it is closer to 1 or closer to 0. Tanh is better with sequence problems, like speech recognition. ReLu and Leaky ReLu are better with images.

3.3.3 Convolution Neural Networks

Convolution Neural Networks (CNN) is a specific type of NN that is used in problems concerning image classification and images recognition. CNN's main ability for recognition and dealing with images comes from its ability to capture the spatial and temporal dependencies in an image by applying filters or kernels. These filters are applied to an image by convolution. The convolution operation is a mathematical

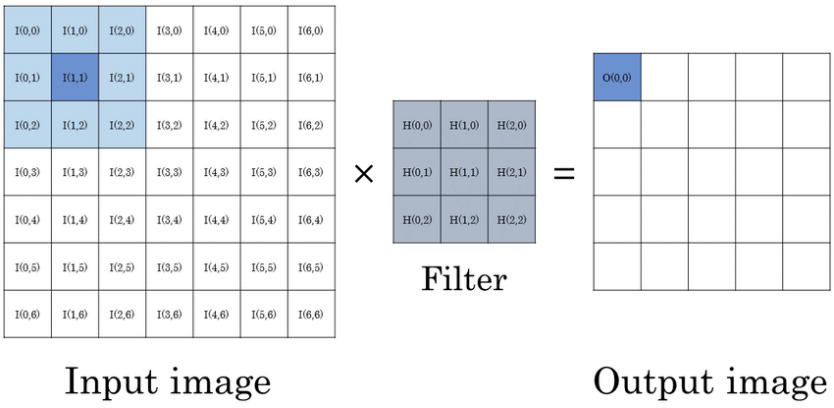

Figure 3.9: example for convolution operation with filter of size 3x3

operation that takes an image along with a filter of a specific dimension, mostly an odd number, and applies this filter along the image. The output of the operation is another image, whose dimension may differ depending on multiple factors, like strides, padding and the filter dimensions. An example of convolution is shown in figure 3.9.

Another version of convolution is called transposed convolution. The only difference between it and the original convolution layer is an applying upsampling on its output, which enlarges the output again to be of the same size as the input.

Stride is the number of movements per convolution operation, whether the right or

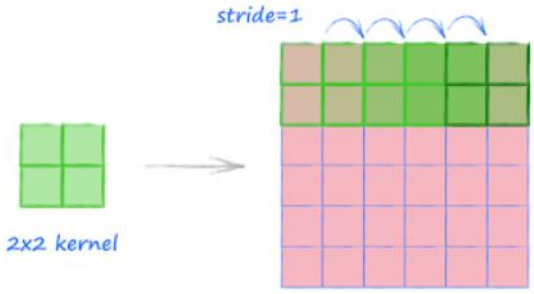

Figure 3.10: example that shows movement of kernel with stride = 1

downwards. An example for stride value of 1 is shown in figure 3.10.

Padding is adding a number of pixels to the input image, such as zeros around the image pixels. An example for it is shown in figure 3.11.

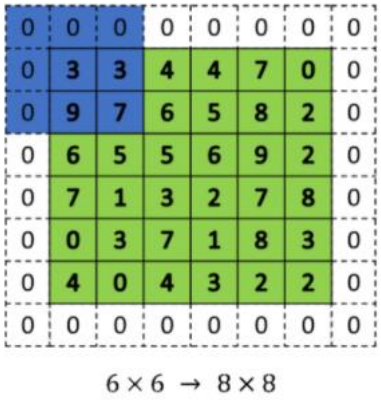

Figure 3.11: Example of padding with zeros

The size of the output image is calculated using an equation:

$$W_{output} = \frac{W_{input} - F_w + 2P}{S} + 1$$
$$H_{output} = \frac{H_{input} - F_h + 2P}{S} + 1$$

Where W_{output}, H_{output} are the width and height of the output image, W_{input} and H_{input} are the width and height of the input image, F_w and F_h are the filter (kernel) width and height, S is the stride value, and P is the padding size.

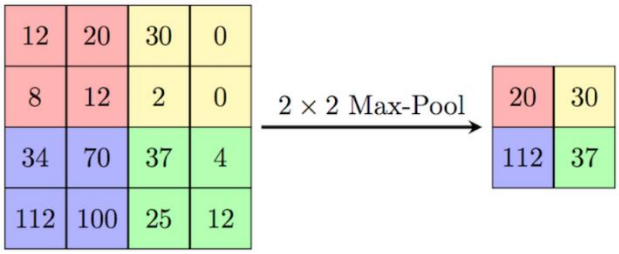

Figure 3.12: Example on Max Pooling with kernel size 2x2

Last important topic inside CNN is pooling layers. Pooling layers are layers inserted in the CNN to reduce the size of the input image so that we can get to one final output in the end, and to learn the important features more than other features in an image, to help in its classification or recognition. There are multiple types of pooling depending on the operation you choose. For example there is Max Pooling. In max pooling, you choose the maximum pixel value inside a kernel, and only take this value instead of the values of the whole kernel, as shown in figure 3.12.

There is also average pooling, which takes the average of pixel values inside the kernel. There is also sum pooling, which sums the pixel values inside the kernel.

3.3.4 Batch Normalization

Batch Normalization is a way of normalizing the output of any layer in the NN to have more of a normal distribution, with mean 0 and standard deviation of 1. The output of the normalization is then sent through a linear function with parameters gamma and beta which are learnable parameters that adjusts the distribution with the data. This helps in making the model more generic to data, regularizing it and preventing overfitting of the data.

3.3.5 Dropout Regularization

Dropout is another method for regularizing the model by randomly selecting some nodes in the NN and ignoring them during training with a given probability.

3.3.6 Optimization Algorithms

Optimization algorithm is choosing the function of learning of the parameters of the model, such as the weights of the node inputs, or the learnable parameters of layers like batch normalization. There are different functions used in different optimization methods. Each of these methods are preferable with an application.

In Replica, we used “Adam” optimization. Adam optimization is a fast learnable algorithm, whose function is based on stochastic gradient descent, which computes the gradient of the weights and tries to minimize the loss between the generated output and the real data. The advantage of the Adam algorithm is its adaptive estimation of the initial values depending on the data given, which are later learnt with the training.

3.3.7 Loss Functions

Last thing that needs to be addressed in NN is the loss functions. Loss functions are used in stochastic gradient descent to choose which function we get a gradient for and try to minimize its loss while training. The most famous losses that represent useful values are binary cross entropy and mean absolute error (MAE).
Binary cross entropy function is:

$$BCE = \frac{1}{N}\sum_{i=1}^{N} -(y_i \log(p_i) + (1 - y_i) * \log(1 - p_i))$$

Where N is the number of data of the batch, y_i is the real classification of the data, and p_i is the probability of y_i classification. The $(1 - y_i)$ represents the other prediction in the binary classification.

Mean Absolute Error function is:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - y_i'|$$

Where N is the number of data of the batch, y_i is the real classification of the data, and y_i' is the prediction of the model.

3.4 Comparative Study of Previous Work

We read about multiple approaches that apply motion transfer in their paper and describe how it was done. Two approaches specifically we found more interesting and clearer enough to understand. We will describe each paper's approach and the differences between both.

In [12] paper, which we chose to implement later on, the authors suggest using two videos one by the user—here called the target—and another video that could be from the internet, of a person dancing, for example a ballet dancer—called the source. Their target is to transfer motion from the source to the target, in their exact background and outfit, just making new moves he never did.

To do so, they first started their algorithm by taking the video of the target—the user—in which they are making any random moves in front of a camera, apart by a couple of meters approximately to capture their whole body clearly, and with a 60fps camera. The video is approximately 20 minutes, so multiplying by the fps, that means they have approximately 72,000 frames. Then they get the poses of each of these frames, which looks like a 2D stick-man in a white background with the same resolution and frame size. After having 72,000 frames of the target and another for their poses, they build a Generative Adversarial Network (GAN) which learns to generate the frames from the poses using this data. This GAN model is discriminant for this specific target.

On the other hand, using the source video—say of a ballet dancer—they divid it into frames, as well. Then get the pose estimated from all these frames. Now, since they have poses of the source, and a GAN trained to generate frames of the target using just poses frames, they used the poses of the ballet dancer as input for the GAN to generate fake frames of the target doing the moves of the source. After that, they could simply convert these frames back into a video. Now, you have the target making the moves of the source, and the motion transfer is done.

In [13], they performed motion transfer between two videos, but using a different approach. One of the main significant things in this paper is depending on reconstructing 3D models to get an accurate pose frame that is used as the conditional of the GAN later on. Another significant thing is using a details enhancement mechanism that refines details such as the hands' fingers, background details and other similar details, to make the frames look more realistic.

In the training part, GAN is trained in a similar approach to the paper we discussed before, but with the help of both videos by using an appearance encoder to extract features of each frame with its corresponding 2D projected pose. Using the mistaken image that is classified as fake by the discriminator of the GAN to train the details enhancement network to observe the enhancements the generator does to end up with more detailed frames of the resulting video.

For the generation of the resulting video, they first needed to reconstruct a 3D mesh of the target. To do so, they extracted the shape information—like height and body proportions—from the target frames and posed information from the source frames using a pretrained model, and then applied both sets of parameters—shape and poses—on a 3D human model. This 3D model is then projects to a 2D frame image, that would be much more accurate now, taking the perspective of the target video camera—which they got beforehand—to obtain a labeled image that is used as the conditional for the GAN to generate fake images of the target in the pose given to the GAN. After generating all these resulting frames, using the detail enhancement network they already trained during training the GAN, they refine the details of the frames to look more realistic. Finally, they combine these frames back into one video.

Others accomplish motion transfers in [14], they decompose the problem of human pose synthesis into simpler modular sub tasks that are trained jointly as one neural network. The first is source segmentation where the foreground is separated from the background then divided into 10 body parts. It takes the source image and source pose then outputs sample masks describing the rough boundaries between the body and background. Second is foreground transformation which applies a separate geometric transformation to every foreground layer. Third is foreground synthesis where it refines the transformed body parts and outputs the target foreground and target mask. Fourth is background synthesis where it initializes the foreground with noise and then assigns it realistic values consistent with the background.

3.5 Implementation Approach

After reviewing different papers addressing the problem of motion transfer, as previously mentioned, we decided to follow the approach of [12], with some modifications, as it was the most clear and precise way from what we have read. We will follow their approach in taking two videos, one of the targets and the other of the source. For the first step we are going to segment the person from both of the video with a classical technique in [3] and pass it to the pose estimation module, we decide to use a classical approach that depend on extracting feature using image processing technique then classify pixels of the frame and not a deep learning approach. As our project does not rely on real time, using a classical technique would not affect it. Also, we are going to follow a tracking technique to avoid segmenting each frame so we could segment one frame and track this segmentation for a while then start segmenting again to work in decreasing time.

In pose estimation, we want to estimate the poses of both video frames, we will use classical approach methods rather than using a deep learning model, as it has a lower cost, by using the method proposed in [6]. After getting poses of both the source and the target, as in [12], we would give poses and the target frames to a generative adversarial network to train to generate a target frame from its pose. Then using the poses we extracted from the source, giving it to the already-trained GAN, we could manipulate frames of the target, making poses of the source. Then merge these frames together to get one manipulated video of the target. Hence, the motion transfer would be made.

Note that we reverted from using approaches that rely on 3D models as it has much more complicated aspects and problems to face, and would require more time and money to implement, and to buy a human model we could use in one of 3D software.

4 Chapter 4: System Design and Architecture

In this chapter, there is a full description of each module design and architecture in our project. First, an overview and any needed assumptions we used will be explained. Then the overall system architecture and block diagram. Then for each module, there is a functional description, modular decomposition, design constraints and any other needed descriptions.

During development and implementation of our project, we made sure the project code is modular and clear enough to be understood, in case of any future need for the code itself.

4.1 Overview and Assumptions

Replica project consists of 3 main modules that need to be completely understood before starting implementing such a thing. These three are image segmentation, pose estimation, and generative adversarial networks (GAN). Apart from that, there is the connection between each module and the other, the research and modularity that needs to be considered. Each of these modules are described in detail in the following section, and how they all connect to each other, and how they represent the system architecture.

Some assumptions are also considered in order to simplify some of the hardships of the implementation. In the GAN, due to the complication of its design and architecture and its need for a heavy-work machine to train on, we simplify a few things in it. The GAN's input needs a high-resolution image, with good clarity, which is why we use a dataset provided by [12], which has full HD, which is of size 1024x512, images of a video shot by a 120fps professional camera for 20 minutes on average. In some trials of the GAN, we try decreasing this resolution to half HD, which is of size 512x256. Other modules have some assumptions and constraints described in each subsection below.

4.2 System Architecture

The system main modules could be explained in this flow:

1. Preprocessing (Segmentation)
2. Extracting Pose
3. Generating Manipulated Frames
4. Merging Modules

Before preprocessing, there is a “converting videos to frames” block, which takes two input videos for the project. One video is of a user (target) doing random motion, as shown in figure 4.1, which shows a random motion or pose from the user video. The other video is of a dancer or whichever the source video is, as shown in figure 4.2.
This block takes both videos and converts them into frames, which are the output of the block.

Figure 4.1: Frame taken from a user video showing a random motion by the user

Figure 4.2: Frame taken from a source video of Bruno Mars singer

The merging modules is the process of applying the flow of the project till the final output, such that the generated frames from the previously explained part are given to the preprocessing module, and it outputs segmented image of the frames, which are given to the extracting pose (pose estimation) module. Noting that in case of a deep learning approach for pose estimation, we could take the frames directly without segmenting them. Then the extracted poses of the user (target) frames are given to the GAN module, while the poses of the source are saved for use in generating the manipulated (fake) frames, which takes this input along with the output model from the GAN module. The output, which completes the motion transfer operation, is then merged back into one video and is the final output of the project.

Each of preprocessing, pose estimation (classical and deep learning approaches), and GAN are explained in detail in following subsections. GAN module is divided into two subsections later, where the first explains building the GAN model, and the other explains training and testing it (generating manipulated frames), which are the two steps applied in the process of motion transfer for each new user and source videos. The model architecture is built only once.

4.2.1 Block Diagram

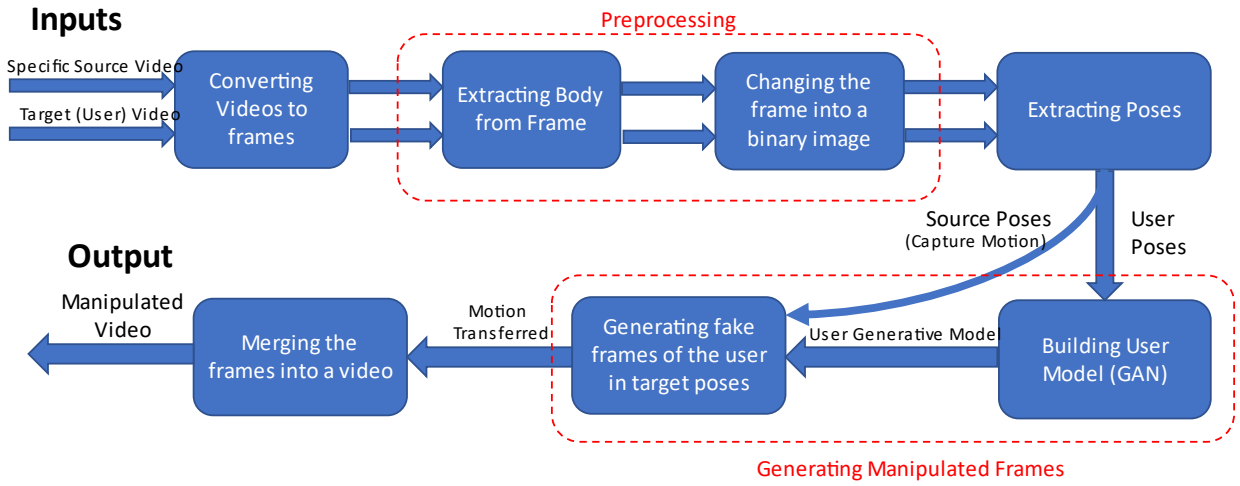

Figure 4.3: Replica Architecture Block Diagram

4.3 Preprocessing (Segmentation)

Segmentation is a technique for dividing the image into two regions, the foreground region that contains the object of interest and the background region that contains any other objects. In image segmentation it's hard to extract the human body depending on tracking motions as we have only one scene, but working on video segmentation is easier as it is accomplished by extracting the closest moving object as foreground. The objective of this module in our project is to extract the human body as a foreground region in both of the input videos; the source video and the target video.

This module's input is a video of a person performing some motion with different poses; the person has to be the closest object to the camera and has to move through the whole scene to be able to capture his motion. The output is a new binary video that has the white pixels represent the human as foreground and black pixels represent another thing in the scene.

4.3.1 Functional Description

As shown in figure 4.1, segmentation step is applied on the target and source videos where each frame in the two videos is segmented and the moving object is detected. This modules has three main stages, first is constructing the background by taking the median across multiple groups of frames to detect the fixed parts of the frames through the video, second is then background subtraction where the obtained background is subtracted from each frame, Third is an enhancement step, as it modifies mislabeled foreground areas in each subtracted frame using a color similarity measurement by comparing their average color to adjacent foreground clusters.

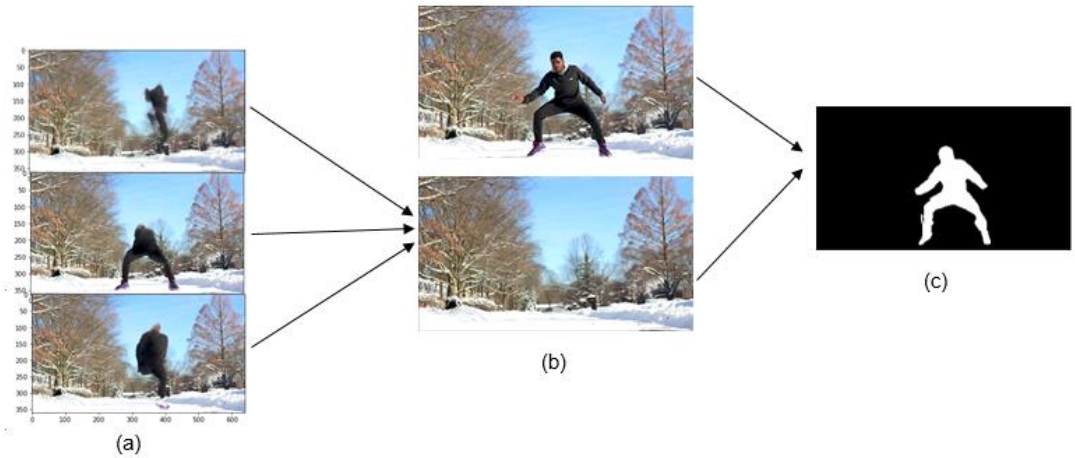

Figure 4.4: Flow of Segmentation; (a) a group of medians (b) extracted background and the frame (c) after background subtraction

4.3.2 Modular Decomposition

This module could be decomposed into four modules; background construction, background subtraction, superpixels, and color similarity feature.

4.3.2.1 Background construction

This module is responsible for extracting the background of the video which is used to segment the moving person through each frame. Background is usually obtained by taking the median of the video frames which produces relatively accurate results in most situation where the person is moving a lot in every part of the environment where the video is being shot, but this method faces a challenge when the person stays in a fixed position for a long period through the video and the generated background will contain a large part of the person which is mislabeled as background. In our approach we try to overcome this problem by dividing the video into smaller groups of frames of fixed sizes relative to the video duration, we take the median of each group separately and store them. As mentioned above the median frame of each group depends on the person's movement, if the person changes his position the median frame will be more indicative of the background otherwise a large part of the person will be mislabeled as background

We iterate through the median frames and select two of them randomly at each iteration and subtract them, the subtracted frame is a binary image where white pixels represent the areas where the person was standing and black pixels represent the fixed areas of both frames which is the background so at each iteration we discover new parts of the background that were hidden by the person in previous medians frames. We Iterate through the medians which will eventually construct the whole background.

Dividing the video into groups of fixed size frames where each of them have equal importance in the process of background construction produces better results than taking the median of all video frames which gives higher importance to the part of the video where the person stays in a fixed position for a long period. This approach is not completely robust when the person remains in a fixed position for a long time, the generated background still has some mislabeled parts which we try to modify in the color similarity feature.

4.3.2.2 Background subtraction

This module is responsible for separating the foreground from the background in each frame. It takes the background, generated by the background construction submodule and subtracts it from each frame in the video, then we apply a series of morphological operations mainly; opening and closing to remove any noise resulted from the subtraction like random noise in the background or holes in the segmented body. This module doesn't give us the final result as it has some noise from a different source; we could have noise from bad background construction, or we could have some noise from the users that are wearing cloth with the same color as the background or other noises, so we do the next step to enhance these noises.

4.3.2.3 SuperPixels

Using superpixels in segmentation is a widely used technique as it divides the image into a group of pixels that share common characteristics; these groups are called SuperPixels. With these superpixels, we just need to identify which cluster belongs to the foreground and which cluster belongs to the background, which will conclude the segmentation problem. In this module, we tried two approaches using Kmean and SLIC (Simple Linear Iterative Clustering). With the Kmean technique, we used several features to cluster the similar pixels together. We used saliency maps, CIELAB colors and optical flow. Saliency shows each pixel's unique quality, while CIELAB color shows the brightness as it consists of 3 channels L, a, and b; L* for perceptual lightness, and a* and b* for the four unique colors of human vision: red, green, blue, and yellow, while optical flow shows the motion and tracks it through the frames, these features help to group human pixels as shown in figure 4.5.

Figure 4.5: frame with super pixels

But using SLIC gave us a better result for clustering as it is faster than using Kmean. It depends on the distance of colours between pixels. Also, Kmean performs well only for a small number of clusters between 10 and 15, so we choose to use SLIC to generate superpixels. After getting the superpixels, we assign each one with a label as foreground or as a background, depending on the previous results of background subtraction. This assignment depends on the majority of the labels of each pixel in the superpixels. If the number of white pixels in this superpixel is bigger than half of the size of the cluster we assign it as a foreground cluster, otherwise we assign it as background. However, in some videos due to the closeness of the color between background and foreground, maybe there are mislabeled clusters that will be enhanced in the next step.

4.3.2.4 Color similarity

This module is responsible for modifying mislabeled clusters due in the generated background by computing a color similarity measurement between adjacent clusters. It takes each segmented frame and locates adjacent pairs of foreground and background clusters then computes the color difference between them using the euclidean distance equation. The color difference helps identify background clusters that are close in color to foreground clusters and were mislabeled in the background subtraction step.
We relabel a misclassified cluster if its color difference does not exceed a certain threshold.

4.3.2.5 Foreground Separation model

This is a different approach to classify superpixels rather than using background subtraction method then applying color similarity. This approach did not produce good results, so we followed the approach mentioned above using background subtraction .
This module uses superpixels from the previous step, saliency, CIELAB color, and optical flow as features to obtain the segmentation of each frame.

First, we get the saliency map of the frame, then apply saliency-based thresholding to label all input pixels. We labeled the entire segment to the label of the majority of pixels in that frame. After saliency thresholding there are misclassified background segments. We used optical flow thresholding to relabel these segments. For every pair of foreground and background segments we compute their Euclidean distance of their average CIELAB optical flow color values, then apply thresholding using a defined flow threshold f.

Then we apply color thresholding which consists of two parts; superpixel-level color thresholding and pixel-level color thresholding, after flow thresholding some background segments remain mislabeled as foreground due to color similarity with foreground segments. To relabel such segments, we define a superpixel-level color threshold C_s:

$$C_s = \frac{1}{T} \sum_{i,j \in B_s} \sqrt{\sum_{x \in \{l,\alpha,\beta\}} (x_i - x_j)^2}$$

Where l, α, β are the CIELAB color values for each adjacent segments (i, j) from a set B_s where one segments i labeled as foreground and the other segments j labeled as background and T is the total count of such segment pairs. We compute Euclidean distance between the average CIELAB color values for each background and foreground adjacent pairs and compare it to C_s, if the Euclidean distance is less than the threshold C_s then we relabel the pair as background, otherwise we relabel them as foreground. The relabeling process is repeated until convergence. The second stage in color thresholding is pixel-level color thresholding which applies the same steps as the previous stage except at pixel-level.

4.3.3 Design Constraints

Using classical approaches in segmentation impose some constraints on the video. As classical approaches do not depend on features in the frame such as the human anatomy which complicates segmentation, deep learning techniques exploit such features greatly.

The input video for this module has to contain only one person where he has to be the closest object to the camera so that he is completely visible and each part of him is not hidden by some object. Also, he has to be the only moving object in the video therefore no other object would be detected by background subtraction. It is also required that the person does not stay in a fixed position for a long period so that it does not affect background construction where a large part of the person is mislabeled. Additionally, the person's outfit color has to be different from the background color so background subtraction would work most efficiently.

4.4 Pose Estimation (Classical Approach)

This module is responsible for estimating the 2D pose of a human. It extracts the places of the joints of the human body. This module uses generative methods to extract the pose of the human as it depends on a human body model to model the pose.
In this module, a classical approach is followed to achieve the pose estimation which doesn't use any dataset to extract the pose but it does require a prior knowledge of general human body constraints and it applies these constraints to the human body model to accurately estimate the pose.

4.4.1 Functional Description

This module takes the segmented image of a human coming from the segmentation module where the background is in black and the human in the image is segmented in white .the module initializes the body model of the human based on prior knowledge of general body constraints using the code file written in python then it starts optimizing the body model to fit the segmented image. The output is an image of the pose which consists of x,y,theta coordinates of the head, torso, arms and legs of the human as shown in figure 4.6.

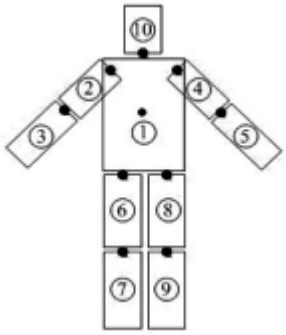

Figure 4.6: Body Model

4.4.2 Modular Decomposition

This module could be separated into a number of distinct sub-modules each responsible for a different task.

4.4.2.1 Body Model Initialization

The first step in this module is to initialize the body model used to estimate the proportions of the person in the image. We model the body as a tree structure where the torso is the root of the tree and the head,upper arms and upper legs are the children of the torso as shown in figure 4.1 . For each body part we need to calculate an initial length and width based on the foreground image and make an initial guess of the x,y and theta coordinates of each body part given that those initial assumptions don't violate normal human body constrains such that the top of the shoulders are attached at the top of the torso and that upper legs are attached to the bottom of the torso and so on.

So first we initialize the root of the tree which is the torso. Its height and width are calculated based on the foreground region of the image which gives us an estimate of the height of the whole body and we model the height of the torso based on the "Vitruvian Man" [15] which represents the ideal human body proportions so it makes a good initialization for the body model tree. The next step after obtaining the height and width of the torso is to get the x, y and theta of the torso which we obtain by running a distance transform on the foreground image to get the body distance map which gives a good assumption that the larger the pixel's value is the more likely it is to be in the center of the torso. Through this distance map we obtain the proposed torso center (x,y). Then to obtain the angle theta of the torso we use the foreground principal axis to obtain an initial proposal of the angle of the torso.

After initializing the torso, we initialize the rest of the body parts based on the same prior knowledge of body parts constraints used in the torso. The angle of the rest of the body parts are sampled randomly for the initial state using a normal gaussian distribution to sample from. Each body part is represented using a rectangle.

As for pose estimation in a video sequence there are usually small changes in the pose between consecutive frames so the body model is initialized using the above steps only in the first frame of the video and then the initial for the rest of the frames is the output of the previous frame.

4.4.2.2 Posterior Probability Calculation

This sub-module is responsible for calculating the posterior probability of the body model which will be used in the local optimization step. The calculation of the posterior

probability is an important step as it measures the degree to which the body agrees with the foreground image and therefore our objective is to increase the posterior probability. The posterior probability is formulated by the following equation which is used in paper [6].

$$P_f(I|X) \propto \exp\left(\frac{S_i - \beta S_o}{S_u}\right)$$

Where I is the foreground image, X is the current synthesized body model , S_i is the area of intersection between the body model and the foreground image, S_o is the area of the overlapping regions among the body parts of the body model, S_u is the area of the union between the body model and the foreground image and β is a penalty factor for overlapping body parts. After several trials we found that setting beta to 0.65 yields best results as the overlapping between different body parts happens when a body part is occluded by another body part. This likelihood is maximum when the model body agrees with the foreground best. Our goal is to make the model body cover as much foreground as possible and cover the background as little as possible.

4.4.2.3 Local Optimization

This sub-module is responsible for finding a local optimum in the state space of the tree structure. First step in local optimization is to choose a body part to optimize. The choice of the body part is based on two factors. The first factor is that the body part that has the least agreement with the foreground is given higher priority. We apply this factor using an importance probability calculated for each body part. The importance probability is calculated by the following equation used in paper [6].

$$P_{input}(i) \propto \frac{S_{bgIn}}{S_i} + w\frac{S_{fgOut}(S_i + S_o)}{S_i^2}$$

Where S_i is the area of the body part in the body model, w is the weight, S_{fgOut} is the nearby foreground area not covered by the body model and S_{bgIn} is the background contained in the body part, as shown in figure 4.7.

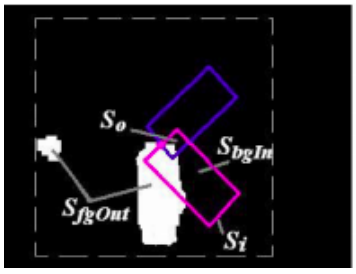

Figure 4.7: Importance Priority

The other factor considered for the selection of the body part is the human body topology meaning that the priority of each body part decreases from the root node to the leaf node.

After choosing which body part to optimize based on the factors mentioned above, we begin to change the angle of the body part by a step length in the positive direction and then calculate the posterior probability for the body model and check if it has increased. If not then we change the angle of the by the same step length in the negative direction and check the posterior probability again and keep the change if the probability increases. This step is repeated several times on the same body part until the posterior probability cannot be changed or it reaches the maximum number of loops defined. In the case that the importance probability of the body part is stuck in its place and does not get better with any of the changes done to the angle then this body part is re-sampled once again with a random angle to act as a new initial for it. After optimizing the body part we recalculate the importance probability for all body parts and choose the next body part to update. All of these steps are repeated until the posterior probability of the body model converges.

In our approach we apply different random starts to the same foreground and do the whole process on all of them and select the pose that yields the maximum posterior probability. In the last step after getting the pose we change the body part model represented by rectangles to a skeletonized body model as shown in Figure 4.8.

Figure 4.8: Skeleton Pose

4.4.3 Design Constraints

Using a classical approach in pose estimation imposes some constraints on the image. The image has to contain only one segmented person and each part of him has to be

clearly visible in the foreground image as the accuracy of the pose depends highly on the accuracy of the foreground image. If some body part of the human is occluded, the accuracy of the resulting pose would decrease. For better accuracy the person has to be facing the camera so that all of his body parts are visible.

4.5 Pose Estimation (Deep Learning Model)

This module, as the previous, takes the frames of both the user (target) and the source, and outputs the poses of these frames separately.

4.5.1 Functional Description

In this approach, we use a deep learning model, OpenPose, trained on multiple images and outputs the locations of the key points of the pose skeleton, or generate the pose image right away as in figure 4.9.

Figure 4.9: Example on output of the OpenPose model

4.5.2 Modular Decomposition

After installing the module and running it, we give it a path to the frames we have from the videos, and it outputs the images and keypoints locations.

4.6 Building GAN Model

GAN module is a large part of the project, where first the GAN model architecture needs to be completely understood and chosen wisely to generate the output we need from it, considering that there are multiple types of GANs used for different purposes, as was explained in chapter 3. After building the right GAN model, it's then used with each user and source video as explained in section 4.7.

4.6.1 Functional Description

This module focuses on building the right GAN model architecture. In Replica, we found that the best architecture to be used for the motion translation is Pix2Pix model architecture, which is explained in [16]. Pix2Pix model architecture's main purpose is for image-to-image mapping, such that the GAN model is given an input image, and it generates another image. Pix2Pix model was famously used in applications like converting satellite images to Google map images, or to convert black and white images to colored one.

In Replica, we need the GAN to take input image the poses of the user and generates a real frame of the user. However, whilte training, it takes both images as inputs, and it tries to generate one like the real image, calculating the loss, and backpropagating it to modify the weights of the model to fine tune it better.

4.6.2 Modular Decomposition

Pix2Pix model consists of two main models; generator and discriminator, as any GAN model. The generator model outputs a synthetic image that could be taken as real. The discriminator model outputs whether a generated image is real or fake, depending on the provided dataset. These two models are trained simultaneously in the GAN model, such that the generator improves itself to generate more plausible fake images that can fake the discriminator. While the discriminator improves itself in determining fake images. In some aspects, Pix2Pix model could be considered as a cGAN since its condition is the input image to the generator, which represents its latent space.

The generator model architecture depends on the encoder-decoder model architecture concept, but some modifications to make the real image associated with the input of the GAN in one run, in other words, to make poses and real frames associated in one image in the decoding step, to make the output more realistic. The encoder-decoder concept depends on encoding the input image of the generator and reducing its size till some point, then decoding this back to an image of the same resolution. In other words, it tries to convert one image to another through the process of decoding and encoding, which is shown in figure 4.9. The modification on this network could be shown in figure 4.9, and this architecture is called the U-Net [17]. The generator is then trained using adversarial loss, which improves its weights to make more plausible fake images.

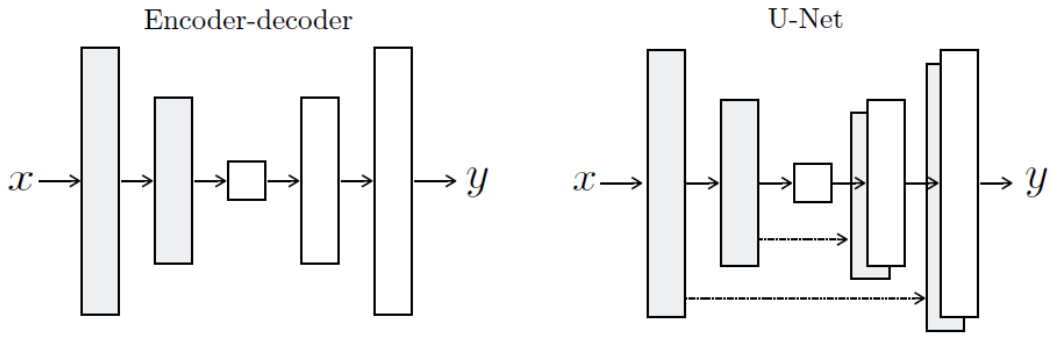

Figure 4.10: Encoder Decoder architecture is shown on the left. On the right is the U-Net architecture.

The encoding block is basically a small CNN that applies a 2D convolution layer whose input is the input layer to the encoding block, and applies a specific number of filters on the layer, given the kernel size, stride and padding. In Replica's Pix2Pix model we used kernel size (4,4), stride (2,2), and padding that persists at the same image size. Then it applies batch normalization on the Conv2D output or not, depending on which encoding block this is. Finally a LeakyReLu activation function, with *a* value equal to 0.2.

The decoding block is also a small CNN that starts with a Conv2D transpose layer whose input is also the input layer and filter number, with the specified kernel size, stride and padding. In Replica's Pix2Pix model we used kernel size (4,4), stride (2,2), and padding that persists at the same image size. Batch normalization is then applied on this output, depending whether dropout is activated or not, with probability of 0.5. Then the final output is concatenated with the skip connection, which is what we explained above in the U-Net, taking the encoding block output and concatenating it with the decoding block.. Finally a ReLu activation function is applied.

The generator model used in Replica's Pix2Pix model contains seven sequential encoder blocks (e1, e2, …, e7). The first encoder block takes the input image as its

input and applies 64 filters on it, without batch normalization activated. The following encoder blocks each take the output of the previous encoder block as input, and apply these numbers of filters respectively: 1228, 256, 512, 512, 512, 512. The final output of the 7th enocer block is taken as input in the bottleneck of the encoder-decoder architecture as in [16]. The bottleneck takes the final encoder output and applies 2D convolution on it with the same parameters as previously mentioned in both encoding and decoding blocks, with 512 filters. Then a ReLu activation function is applied to this output. After this, the decoding part starts with a decoder block with the previous output, 512 filters, and the output of the 7th encoding block for the skip connection of the U-Net, with no dropout activated. This decoder block is followed by another six decoding blocks, the first two don't have dropout activated, while the others do to generalize the final output. Each decoding block takes the output of the previous decoding block and the output of the encoding block in reverse order (e6, e5, … e1), with filter number as in the encoding model but in reverse direction too, which are 512, 512, 512, 256, 128, 64. This is logical and explains the order of the encoding block, considering it has to be of the same size, and as previously explained in chapter 3, the convolution operation output is defined by its input parameters. The output of the 7th decoding block is taken as input to a 2D transposed convolution layer with the same parameters and three filters. A Tanh activation function is applied to its output. The final generator model maps the input image to the output of the last activation function, the Tanh activation function.

The discriminator model, as previously explained, is a large CNN that classifies an image as real or fake, or in other words, tells the probability of whether the image is fake. Keeping in mind the discriminator input image is not just the generated image of the generator, but also the real image of the dataset, both concatenated into one input, called merged input. The discriminator's first layer consists of a 2D convolution layer whose parameters are kernel size of (4,4), stride of (2,2), and padding that persists at the same image size. It applies 64 filters on the merged input image. A Leaky ReLu activation function is applied to the output of this convolution layer. This layer is followed by 4 similar layers, each applying a 2D convolution, then batch normalization, then Leaky ReLu activation function with a equal to 0.2. The filter number of these convolution layers in sequence is 128, 256, and 512. The final layer of the discriminator maps the final output to one value through a convolution layer of 1 filter applied, which is taken to a sigmoid activation function to get the probability of the realism of the image. The final discriminator model maps the merget input to the output of the sigmoid activation function. With Adam optimizer of learning rate 0.0002 and beta1 or 0.5, the discriminator model is compiled to use this optimizer in training, calculated binary cross entropy, and loss weights of 0.5.

The last model that needs to be addressed is the GAN model altogether. The GAN takes the input images and applies the generator model on it. Then takes its output and concatenates it with the real images and applies the discriminator model on the merged input. The final GANmodel maps the source input to both generator and discriminator output. With Adam optimizer of learning rate 0.002 and beta1 value of 0.5, the GAN model is compiled to train using this optimization, and calculates the binary cross entropy loss and mean absolute error.

4.6.3 Design Constraints

There are always multiple design constraints in the GAN model architecture to maintain the ability of both the discriminator and generator models to train in parallel. In the Pix2Pix model, we had to use the skip connection of the encoder-decoder network to let the discriminator determine whether an image is real or fake. Another thing that needs consideration, is the reason for choosing the encoder-decoder architecture, apart from other architectures. The encoder-decoder block is the most suitable for the given problem, considering we don't start from random noise as in other GAN's. We had to ignore using a pre-trained encoder block that maps a random noise into the needed latent space, since here it is directly the input images. This made the results of the GAN better than starting with random noise, but also made the generator model slightly larger.

4.6.4 Other Description of GAN Model

It's worth mentioning that the generator model is not compiled and doesn't calculate loss or have an optimizer. This is because the generator model loss is not the most meaningful when read. It is always more meaningful to check the generated images itself and see how close they are to reality.

4.7 Generating Manipulated Frames

After the GAN model is successfully built and ready to be used, comes the part of training that is given the proper input. Then generating the manipulated (fake) frames of the user making the poses of the source.

4.7.1 Functional Description

To train the GAN model, generator and discriminator models to be able to learn to generate realistic images of the user given the poses. Everybody Dance Now [12] dataset consists of frames taken from a user video, and posees associated with each frame. This dataset could be used to test out the efficiency of the GAN. Generally though, the input to this stage is the frames of the user and the output of the pose estimation step. The output of this step is a trained GAN on the user.
After having a GAN model trained on a specific user, this GAN model given the poses of the source would be able to generate manipulated frames of the user making the poses of the source. Which would be converted back to a video, which is the final output of Replica.

4.7.2 Modular Decomposition

The training of the GAN is reflected back on both the discriminator model and the GAN model, considering that the generator model doesn't train or calculate loss, but just given the proper input, generates the proper output. The training starts by taking the input dataset and dividing it into training images and training poses. Then with a suitable batch size, considering the whole dataset would be very large, and depending on the GPU capability, we calculate the number of steps of training the multiplication of the number of epochs to train on by the batch per epoch.

For each step, we take a sample of real data separated into training images and poses., and we generate an equivalent matrix of 1's values, of the same size as the training images or poses. Considering that 1 represents real for the generator, and 0 represents fake. Another fake batch is also generated by the generator given the training poses, with an equivalent matrix of 0's values, of the same size as the training poses. After having these outputs, we train the discriminator model once on the real data with the 1's matrix, and once with the fake data with the 0's matrix. Then the GAN model is trained with the real poses, 1's matrix and real images. The returned losses of these three are what needs to be considered. Finally we summarize the performance of the model and save samples of the data and the generated data to be taken into consideration after modification.

In the testing, or generating the manipulated frames, we use the saved generator model and the testing data, or in other words, the source poses, and give them as input to the generator model to predict (genearete) the fake frames. These frames are the final output of this module, which are saved and then merged into a video.

4.7.3 Design Constraints

The training didn’t have many constraints except the fact that we needed to slow down the discriminator training to match with that of the generator, such that the discriminator model is easier to train than the generator, and if the discriminator model trains faster than the generator, then it classifies all generated images as fake, and hence stops the training of the generator.

5 Chapter 5: System Testing and Verification

Testing of the project is done to check whether the actual results match what is expected, and ensure that each module is doing what it is intended to do, and to ensure that the system as a whole is working correctly. Our project depends on the visual perspective of the output, so, most of our testing was done through testing the modules on different images and videos, and checking whether the output is satisfactory compared to the results of similar projects. In the end, we test the workflow of the project to check whether the results are clear enough to interpret or not, which is done through simple sight comparison to the source video.

5.1 Testing Setup

In order to test our project, we use several videos and frames that contain different poses for both source and target. We test each module on those videos, and the output of the previous module, then check the results based on different criterias for each module which will be mentioned in the following sections.

5.2 Testing Plan and Strategy

We follow a separate module testing to test Replica. For each module, we take samples from the previous module or frames from a video and test the output resulted from the module. Most outputs are visual, which makes the most meaningful test by plain sight. However, we used values to represent some errors and precisions compared to images with definite output we have beforehand. These values are a measure of how accurate or precise the result is.

5.2.1 Module Testing

5.2.1.1 Segmentation Testing

We implemented this module using a classical approach, we tested it manually against ground-truth segmented frames of 2 videos each of them contains around 3000 frames

and measured precision which is a measure of exactness and recall which is a measure for completeness

$$Prec. = \frac{t_p}{t_p + f_p}, \qquad Rec. = \frac{t_p}{t_p + f_n}$$

Where t_p, f_p, f_n are true positives which are foreground pixels matching the foreground pixels in the groundtruth frame, false positives which mislabeled background pixels, and false negatives, respectively. We found that our method produced precision of 0.6 and recall of 0.63.

We also tested this module on various video sequences. All sequences include several kinds of noise caused by illumination changes, small movement in the background, color similarity between foreground and background and reflection as shown in figure 5.1.

Figure 5.1: Comparison between frames in different videos

To compare between videos we use as mentioned before Precision and Recall, and we get the results as shown in table 5.1.

Table 5.1: Comparison between different videos results

Video1(perfect)		Video2(noisy back)		Video3(color similar)	
Precision	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.85	0.86	0.76	0.81	0.62	0.48

In video1 we see the precision and recall is high due to the perfect segmentation as the background is almost empty with no noise and the person moves a lot throughout the video, while in video2 there is some noise not perfect due to there being some objects moved in the background trees but with slight movements.

In video3 we can observe that the girl in the frame is wearing a color that is very close to the background color, so even that background construction gives us a good result but still the background subtraction gives us a very bad result, as it can't differentiate the foreground colours. Therefore the recall is very low due to misclassified foreground pixels that make the true positive low.

5.2.1.2 Pose Estimation Testing

In testing the pose estimation module we used different images that include various poses. The following figure 5.2 shows a sample of the results for a sequence of different poses. It shows that our pose estimation gives good results on different and complex poses. The first row shows the images of the person, the second row shows the output of our pose estimation module.

Figure 5.2: Output samples of our pose estimation

To test how accurately the algorithm performs, we assumed the results coming from the pre-trained model to be our ground truth for the body joints, then we calculated the root mean square (RMS) errors for about 50 frames in 3 different videos. We calculated the difference between the coordinates of the body joints in the ground truth and those in the poses that came from our algorithm using the following equation.

$$RMSErrors = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

The first video has simple poses and good foreground images, the second video contains more complex poses and the third video has simple poses but noisy foreground images. Table 5.2 shows the average RMS error for each of those videos.

Table 5.2: Comparison between different poses results

Video1(simple poses)	Video2(complex poses)	Video3(noisy foreground)
Average RMS	**Average RMS**	**Average RMS**
7.346	13.234	9.663

We also tested with noisy foreground images to see how well the pose estimation performs as shown in figure 5.3 if some poses had very bad results.

Figure 5.3: Pose estimation with bad segmentation

On Average the algorithm takes around 3-4 minutes to extract the pose of the human from a frame. In videos the algorithm takes the same time for the first frame in the video but for the rest of the frames it takes less than 1 minute on average to extract the pose since there are little changes in the poses between two consecutive frames.

5.2.1.3 GAN Testing

The GAN model takes a lot of time for training, depending on the capacity of the GPU and its edition. We have used a Nvidia GTX 1080 GPU with memory size 8GB in the training and testing. Since this is not the optimal GPU for testing, the results of the GAN were not as good as it ought to be, given the time we had for testing. Theoretically, though, GAN gives magnificent results with training over 500 epochs. We only were able to train it for 250 epochs tops, which took an average of 40 hours on the specified GPU.

In the first trial, we trained the GAN on a subject1 (target subject) in the dataset provided by [12] paper, but the GPU memory was only able to take 1000 images of size 512x1024 for training. The dataset included the frames of a video and their equivalent poses, as shown in figure 5.4.

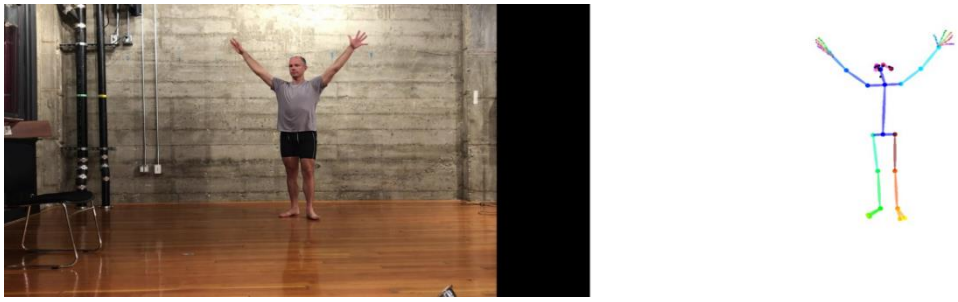

Figure 5.4: sample of subject1 in the training dataset provided by [12]

At the 250th epoch, we had discriminator loss on predicting real images of 0.216, and on predicting fake images, value of 0.204. The generator loss was 1.504, which is an indication of the difference in all pixel values in the generated and real images. This explains why the generated image is not the best just yet. In figure 5.5, while training on training dataset, in the first row are images of two different poses, the second row has the equivalent generated images, the third row has the equivalent real images.

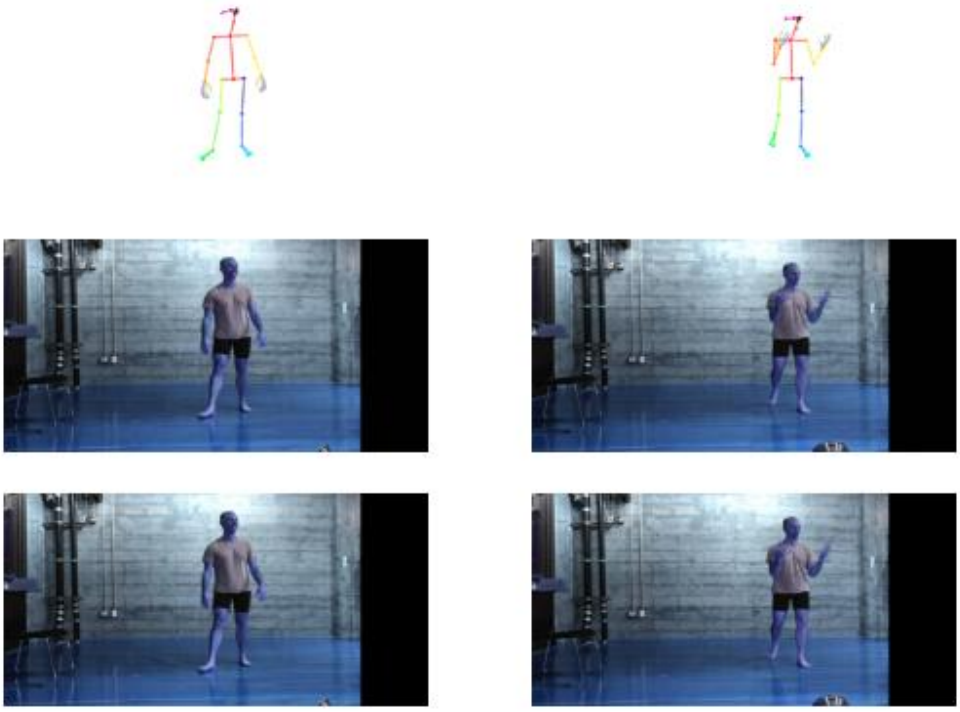

Figure 5.5: Sample of seen data generated by GAN while training.

After training is finished, we took frames from another video (source video) by the same subject, with their equivalent poses as shown in figure 5.6, and used these source poses to transfer its motion to the target user we mention above. Sample of the generated frames by the GAN are shown in figure 5.7.

Figure 5.6: Sample from subject1 validation video

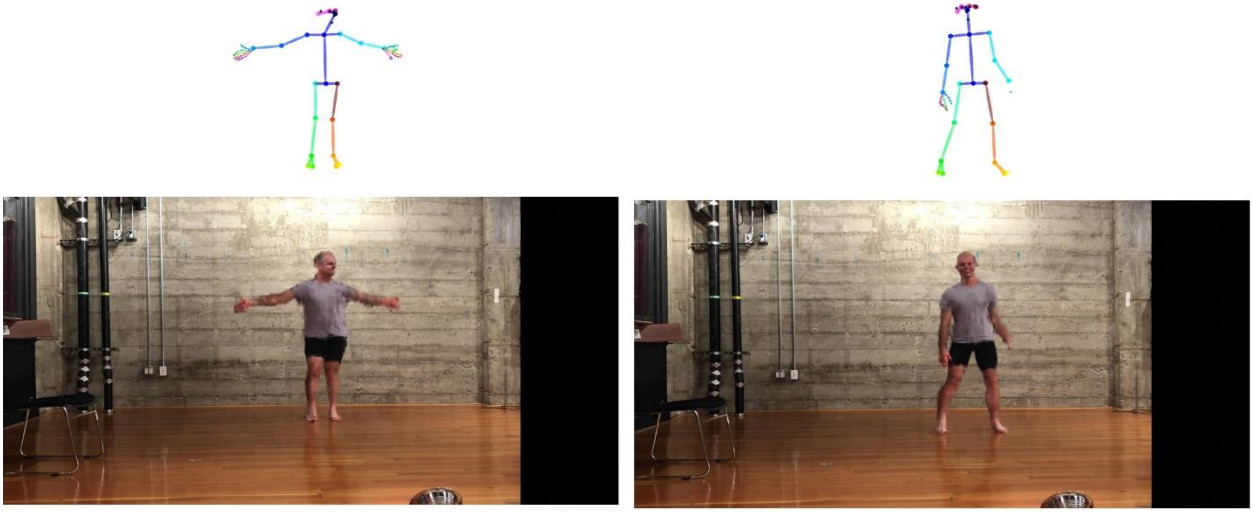

Figure 5.7: First row has sample of vlidation video frames, the second row has their equivalent poses, the third row shows the generated frames for the target user by the GAN

In the 2nd trial, we used a source subject we shot ourselves, segmented and obtained its poses. Sample frames are in figure 5.8. However, there is the problem of difference in resolution, since we do not have a camera that shoots with 512x1024 resolution. We interpolated the image in figure 5.8 to 512x1024 dimensions and tested the GAN on it, but its results were not accurate because of that.

Figure 5.8: Sample from source video we shoot ourselves

In the 3rd trial, we used a source subject from the dataset provided by [12], subject3 validation dataset. Samples from generated frames by the trained GAN are shown in figure 5.9.

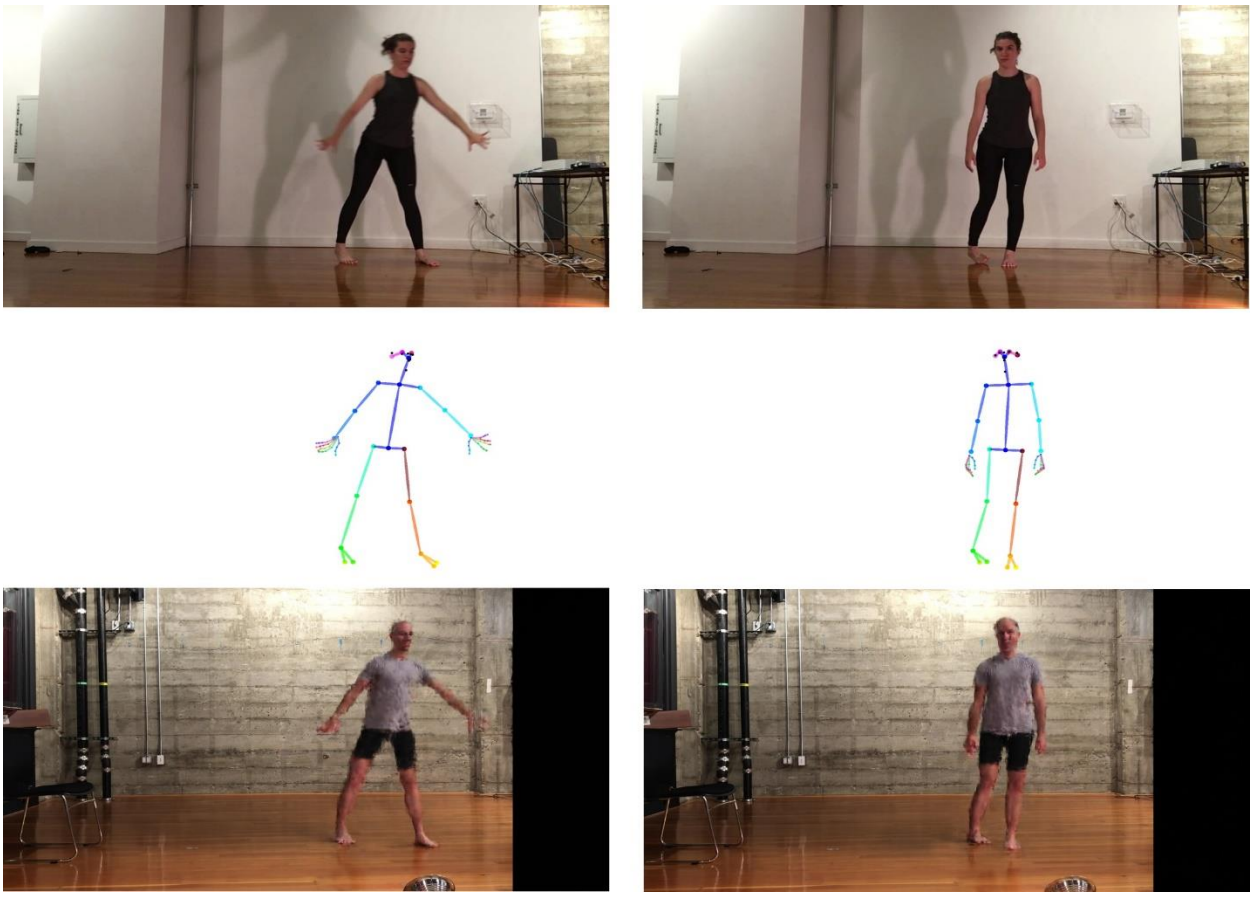

Figure 5.9: First row has sample of source subject frames, the second row has their equivalent poses, the third row shows the generated frames for the target user by the GAN

In the 4th trial, we tried training the GAN on larger number of frames of subject1, but with less resolution. We used resolution of 256x512, and trained the GAN for 180 epoch on 3000 images with their equivalent poses. However, its results were not better than with the higher resolution, but almost the same. We used a code we wrote ourselves to resize the dataset images from 512x1024 to 256x512. After training, we tested it on validation dataset of the same subject but unseen data during training. Sample of generation results are shown in figure 5.10.

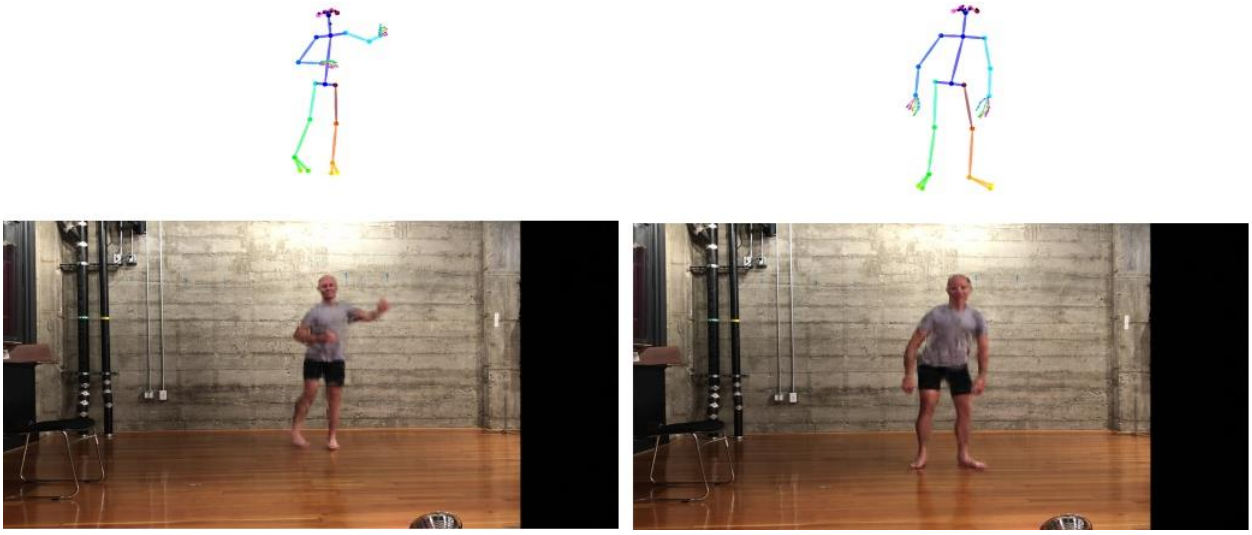

Figure 5.10: First row has sample of poses from the validation dataset, the second row shows the generated frames for the target user by the GAN

5.2.2 Integration Testing

After merging all our modules together and testing the whole project integrated, we noticed that the final results are highly dependent on three things. First is the pose frames associated with the video frames, if it is not accurate, the final output will not be accurate too. This made us notice that it was better to use the deep learning approach than the classical approach in pose estimation, as it gives better results. Second is the amount of the dataset. If our data is limited, the GAN will not be able to generate a variety of poses, and if the images are not clear and taken with a good camera, the results too are not the best. Third thing, and most important in the GAN, is the hardware resources; the GPU we train on. A fast one with a large memory size would be of a huge advantage.

Although our final project results were limited due to our resources, if the project is going for the market, we would have a paid server on a large cloud like Amazon Web Server, which would make it easier for the GAN to train on each user in a shorter time with a large amount of frames, which would give a much better results.

5.3 Test Schedule

Table 5.3: Testing Schedule

Tested Module	Testing Date
Image Segmentation	12th July 2021
Pose Estimation (classic)	May 2021
Pose Estimation (DL)	15th July 2021
GAN	9—16 July 2021
Full System	16 July 2021

5.4 Comparative Results to Previous Work

We divide our comparative results to previous testing on each module, since there are only two relatively similar applications like ours in the market we can compare to, and we compare our modules results to papers in each specified module. Finally we compare our final output to the shared results by Everybody Dance Now [12], and to iOS application Do As I Do: Puppet Master.

5.4.1 Segmentation

We compare our method with the two merged methods in paper[3], paper[12] one measured with SNR and one measured with precision and recall, we perform this test with only one video. Our results were close to both methods with their result measurements as shown in table 5.4.

Table 5.4: Segmentation Results Measurements

Paper3		Ours		Paper12	Ours
Prec	**Recall**	**Prec**	**Recall**	**PSNR**	**PSNR**
0.87	0.66	0.79	0.61	61.2702	54.381

5.4.2 Pose Estimation

We used two different approaches to implement different pose estimation modules, the first approach is the classical approach and the second is the deep learning based approach. Based on running both approaches on different sequences of videos we found that the deep learning approach yields better results especially when the person's body parts are occluded but still the classical approach has its advantages as it doesn't need to be trained on a dataset beforehand unlike the deep learning method which requires a large dataset. The deep learning method extracts the pose in a matter of seconds while the classical approach takes a couple of minutes on average to extract the pose since it uses an iterative approach to get the pose. The classical approach does give accurate results for various kinds of poses as long as different body parts are not fully occluded. We compare our results with [6] which estimates the pose using a similar method to our classical approach. We compare our RMS error and theirs and number of iterations it takes the algorithm to converge on average. Our results are close to theirs as shown in table 5.5.

Table 5.5: Comparison of pose estimation results

Our algorithm		Paper 6	
Average RMS error	**Average number of iterations**	**Average RMS error**	**Average number of iterations**
10.081	55	9.51	50

5.4.3 GAN

We compared our GAN results to the results mentioned in [16]. They had better accurate results and less loss for the generative models, however, they trained the Pix2Pix model on images of small resolution, as small as 256x256 or 128x128 or 70x70. They trained the model for 200 epochs. The small simpler images gave them an advantage in easier generation of images.

In [12], they trained the model on 1920x1280 images and on a large number of frames, like 12,000 images for the one subject. This gave them a huge advantage because the model was able to train on generating a wide variety of poses and postures, which made the results of the motion translation more realistic.

6 Chapter 6: Conclusion and Future Work

Replica project is a relatively complicated project, with the current and available resources, which still faces lots of challenges till it meets the point of perfection to be used for commercial use. It contains complicated algorithms, optimizations, machine learning, and computer vision.

6.1 Faced Challenges

6.1.1 GAN Training

The GAN model used in Replica is a large complicated one that needs high resolution data of large amounts, and these constraints make it harder to train on the available resources we have. It needs a fast GPU of large memory size, preferably 16GB, which we didn't have. We used a 8GB GPU, which still was not able to save all the data at once. We were limited by many aspects; the data size, the number of epochs to train on, and time. All these things affected the output of the GAN.

6.1.2 Classical Approaches Resources

Since our two main modules; segmentation and pose estimation, depend mainly on classical approaches techniques, it was difficult to find robust classical techniques resources that produce good quality results, due to the remarkable growth in deep learning field,which made the classical approaches resources were rare and not up to date and not always accurate or correct.

6.1.3 Performance

In order to produce a relatively good and clear output of the GAN, we were not able to always use the output of the classical approach of the pose estimation. So the performance of the GAN was limited to its input, along with the available resources. However, if the input was not accurate enough and with the limited resources, the output of the GAN would be more of a mess of pixel values than an image.

6.2 Gained Experience

- Researching and reading dozens of papers to achieve working results helped us to be exposed to multiple methods in every submodule, and to see the evolution in the techniques through time.

- We also noted the difference of the development and output between the classical approaches and deep learning approach. Despite the deep learning approach producing better results, at some points, it was more complicated to develop the classical approaches.

- Getting exposed to new technologies such as the deep fake, GAN and working on a large project with multiple modules with a clear workflow as Replica.

- Learning new tools used in development, as described in Appendix A, and learning to work on a remote server to train models on.

6.3 Conclusions

Throughout this document, we demonstrated the idea of motion transfer from one video to the other, and the steps applied to obtain a new video with totally different moves using the concept of deep fake. We also illustrated how the pose of the person is detected from the video and used it to train the GAN model, and showed a comparison between the results of classical approaches versus using deep learning techniques in pose detection.

6.4 Future Work

Replica has multiple possible future extensions that would add value to the project. One of which is adding facial expressions on the pose estimation module to produce more realistic videos. We can also create a motion synchronized dancing video with multiple subjects in the same video, using the same source video to generate motion for all target subjects, so that they all perform the same dance moves. Anothing extension could be adding the option that the detected motion of the source is extracted as an FBX animation file, that could be added on any animation model for game or animation purposes.

References

[1] Radical: https://getrad.co/, 26th March 2021

[2] DeepMotion: https://deepmotion.ai/, 26th March 2021

[3] Asma Hamza Bhatti, Anis Ur Rahman, Asad Anwar Butt, “Unsupervised video object segmentation using conditional random fields”, Signal, Image and Video Processing, Vol. 13, 9—16, June 2019

[4] Anaswara S Mohan and Resmi R, “Video image processing for moving object detection and segmentation using background subtraction,” First International Conference on Computational Systems and Communications (ICCSC), pp. 288-292, 2014

[5] Kai Xu,Longyin Wen,Guorong Li,Liefeng Bo,Qingming Huang,”Spatiotemporal CNN for Video Object Segmentation”, In proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),June 2019, pp. 1379-1388

[6] Xiaoqin Zhang, Changcheng Li, Weiming Hu, Xiaofeng Tong, Steve Maybank, Yimin Zhang, “Human Pose Estimation and Tracking via Parsing a Tree Structure Based Human Model”, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, VOL. 44, NO. 5, MAY 2014

[7] Newell A., Yang K., Deng J. “Stacked Hourglass Networks for Human Pose Estimation.” ECCV 2016, Amsterdam, Netherlands, 2016.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, “Generative Adversarial Nets”, in proceedings of 27th International Conference on Neural Information Processing System, Vol. 2, 2672—2680, Dec. 2014

[9] Alec Radford,Luke Metz,Soumith Chintala.”UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS ”, International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico,7 Jan 2016.

[10] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi, “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network” in proceedings of CVPR 2017, Hawaii, USA, 21 Nov. 2016.

[11] Sakshi Indolia, Anil Kumar Goswami, S.P. Mishra, Pooja Asopa, “Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach”, Procedia Computer Science, vol. 132, 679—688, 2018.

[12] Caroline Chan, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros, “Everybody Dance Now”, in proceedings of ICCV, South Korea, Oct. 2019

[13] Yang-Tian Sun, Qian-Cheng Fu, Yue-Ren Jiang, Zitao Liu, Yu-Kun Lai, Hongbo Fu, Lin Gao, “Human Motion Transfer with 3D Constraints and Detail Enhancement”, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, VOL. XX, NO. X, 580—592, MAY 2020

[14] Guha Balakrishnan, Amy Zhao, Adrian V. Dalca, Fre\do Durand, John Guttag, “Synthesizing Images of Humans in Unseen Poses”, in proceedings of Conference on Computer Vision and Pattern Recognition, Utah, United States, June 2018

[15] Paul J Nicholson, Leonardo da Vinci, The Proportions of the Human Figure (after Vitruvius), c 1490, *Occupational Medicine*, Volume 69, Issue 2, March 2019, Pages 86–88

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks”, in proceedings of CVPR 2017, Hawaii, USA, 21 Nov. 2016.

[17] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, in proceedings of MICCAI, Munich Germany, 201

A. Appendix A: Development Platforms and Tools

A.1 Software Tools

Programming languages and libraries:

- Python: it was our main language to create our project.
- Tensorflow: it is a symbolic math library, and is also used for machine learning applications such as neural networks.
- Keras: it is a software library that provides a Python interface for artificial neural networks.
- Numpy: it is a library for Python programming language, adding support for large, multi-dimensional arrays and matrices.
- OpenCV: it is a library of programming functions mainly aimed at real-time computer vision.
- Scipy: Open-source software for mathematics, and engineering equations and functions.
- Scikit-image: Collection of algorithms for image processing
- Scikit-learn: Tool for efficient predictive functions and data analysis
- Imutils: Collection of image processing functions
- Tqdm: A Fast, Extensible Progress Meter for processing video to frames

Tools and platforms

- Github: version control.
- Colab: Jupyter notebook environment running wholly in the cloud.

B. Appendix B: User Guide

Our final application interface is in the form of a desktop application that just requires the user to click on the .exe file to run the program. The user will find the program as shown in figure B.1.

Figure B.0.1: Replica Interface

The user is required to choose the path of his desired video, which is the target video, and the source video the user wants to transfer motion from to his video. To do either, the user clicks the “Upload” button below the video text box, and a window will open to pick the video path.

By clicking the “Transfer” button, the code will be run, it would take some time as expected because we need to train GAN for every new user video. In the near future, our code will be deployed on a server with high capability, that would make the code run much faster, and the user would not need to wait. The resulting video would be sent to his mail.

C. Appendix C: Feasibility Study

Motion transfer is a growing market in several fields such as the animation and gaming industry. The enhancement in GPU's makes motion transfer an easier process which would attract more demand in this market.

Our target customers are game developers, movie makers and users. Game development can benefit from it in the motion of characters to get precise movements. In movies, we can apply it to get a difficult motion that requires a stunt double or to generate motion for animated characters. Our tools will be provided to users in a software as a service platform. However, right now our initial customers are the users and the entertainment field.

Through our survey for sthe market on similar applications, we found applications that use deep fake technology and motion transfer in general. Application Do As I Do was able to use deep fake technology to generate fake frames using a target image and source video. Application ReFace captures target facial keypoints and maps it on source video, so it seems like the target is the one in the source video. Applications like Radical and DeepMotion simply capture the motion of the target and map it to a pre-made model.

For our business case, based on our previous market analysis, we are going to offer an initial free trial for each new user, such that the user can upload one video of himself (target video) and can choose a number of source videos from a limited available options (for example 5 free source videos) on the website to have the motion transferred from either of them to his target video. After this free trial, if the user wants to buy another source video that is not free, each source video would cost 50 cents only. If the user wants to change the target video to another one, he can upload another video for three dollars. Prices will be reconsidered after one year and then on bi-yearly basis, based on analysis of our customers. Promotions are also planned at times of holidays. Incentives will also be offered so that the source video cost could be less around 30% if the user would buy more than one. The target at this point is 10 videos at a time in order to achieve our first million users in five years.

It is expected that the advertising would help in making the application more popular and used, but more importantly the propaganda on social media, as most of the well-known applications now started by getting known among teengers and social media users. Suppose that in the first year, there are 500 users for the application, and each paid $20 on the application, then the total subscription profit in 1st year could be $1000.

As years go by, more users would subscribe and more profit could be made. In addition, ads profit. On the other hand, the cash out would include the cost of the cloud server that would be used in training of GAN, and the website host to put it online. Also, there are the salaries of the employees and office rent and bills. Lastly, the cost of the advertising on other websites. More details are shown in table 2.1.