UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Comparative study on automatic face edition

Autor: Richard Sudario Berrocal

Director: Sergio Escalera Guerrero

Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, June 11, 2022

Abstract

In 2018, a video of Barack Obama making a fake speech played by Jordan Pelee went viral [27], in the same year we have seen similar videos with different speeches and we have even seen how totally static images were animated, giving them a pattern to follow. That video and the other ones were made with Deep Learning technology which allowed to recreate the face of a person in the body of another or animate static faces.

Since then techniques such as deep fake, face swapping, or face motion became more popular [14], we have seen it both in the movies and on social media, and thanks to this we can see how this impacts us on a social level, for both good and bad.

All of this is because Computer Science has advanced exponentially lately, what we thought impossible a few years ago, is now possible and easy to do. Research in the field of computer vision has been very impressive and in a short time, we have gone from simply creating objects to generating artificial faces and making them replicate the patterns of behavior of other people.

This project will consist of a state-of-the-art review of these different techniques. Three methods will be studied, all of them to edit the image that they obtain as input. It will detail the pipelines used and will show some results, with fixed parameters, to the public to obtain a qualitative assessment to define their faults and highlight the best of each method. And finally, a conclusion on the results obtained and the current status of this technology as well as its social impact and the applications that this involves.

Resumen

Por allá el 2018 un vídeo de Barack Obama diciendo un discurso falso interpretado por Jordan Pelee fue viral [27], en el mismo hemos visto vídeos similares con diferentes discursos, todos ellos falsos, incluso hemos visto como imágenes totalmente estáticas pueden ser animadas, dando un patrón que seguir. Ese vídeo y los otros han sido hecho mediante tecnología Deep Learning la cual nos permite recrear la cara de una persona en el cuerpo de otra o animar caras estáticas.

Desde entonces estas técnicas como deep fake, face swapping o face motion se han convertido en populares [14], lo podemos ver tanto en películas como en las redes sociales y es gracias a estas que podemos ver el impacto social que tienen, tanto para lo bueno como para lo malo.

Todo esto es debido a que la informática ha avanzado de forma exponencial ultimamente, lo que

pensábamos imposible hace unos años es hoy mismo posible. De hecho, la investigación en el campo de la vision artificial ha sido impresiónate y en poco tiempo hemos pasado de crear objetos simples a crear caras de manera artificial, incluso son capaces de replicar los patrones de comportamiento de otras personas.

Este proyecto consistirá en un estudio del arte sobre estas diferentes técnicas. Se estudiaran tres metodos distintos, todos ellos con la finalidad de editar una imagen que reciben como entrada. Se detallara la pipeline que usan y se mostraran resultados, con parametros fijos, al publico para poder obtener una opinión cualitativa con el objetivo de encontrar sus fallos y destacar en lo que mejor hacen. Para finalizar se concluye con una reflexión de los resultados obtenidos y con el estado actual de esta tecnología, también se hablara del impacto social y las aplicaciones en las que se usa.

Acknowlegements

First of all, I would like to thank my tutor, Dr. Sergio Escalera, for advising me and guiding me through everything, without him I could not have completed the work on time, thank you for taking the time he had despite his busy schedule.

Also thanks to the teacher Cristòfol Garcia and Ruben Nadal, both teachers of my higher degree DAM who pushed me to start the career, and partly I am still advancing thanks to them.

I would also like to thank my family and friends, especially my friend Yaiza Cumelles, without her support I would not have been able to continue.

And finally, I would like to thank myself for the mental load that has been to carry out this work and that thanks to knowing how to deal with it I have been able to finish it with the result I expected.

Contents

Introduction **1**

1 Related work **1**

1.1 Deep Learning 1

1.2 CNN 2

1.3 GAN 3

1.4 Deep Fake 3

1.5 Autoencoder 4

2 Methods **5**

2.1 Deep Face Lab 5

2.1.1 Pipeline Definition 5

2.1.2 Post Processing 9

2.2 CoMotionSegment 10

2.2.1 Pipeline Definition 10

2.2.2 Training 13

2.3 Stitch it in Time 14

2.3.1 Pipeline 14

3 Testing and Evaluation **19**

3.1 Technical Specs . . . 19
3.2 Input . . . 19
3.3 Software . . . 21
3.3.1 DFL . . . 21
3.3.2 CoMotionSegment . . . 24
3.3.3 STIT . . . 25
3.4 Personal evaluation . . . 26

4 Survey **29**
4.1 Inputs and parameters . . . 29
4.2 Graph with results . . . 30

5 Ethics **37**
5.1 Current state of the technology . . . 37
5.2 Social Impact . . . 38

6 Conclusions and further work **39**

Parameters for training in DFL **41**

Bibliography **42**

Introduction

Automatic editions technology is currently being used a lot in the world of film, television [5] and social networks, we can see it in The Mandalorian Fig 1 or Fast And Furious and the use that they give it is to revive actors or recreate the actor younger. We can also see a great boom in applications and posts on social networks related to face swapping or face animation.

Figure 1: Luke Skywalker as young using Deep Fake.

And with its popularity has come a growing number of different implementations to achieve the same goal.
However, not all of them are correct and have some flaws, particularly in terms of quality. Some fail to recognize the face, extract it and even position it on the destination.
That is why it is necessary to study them internally and see where they fail and how they can be improved.

To compare the methods, an in-depth study of their implementation will be carried out.

We will look at the pipeline they use, the neural networks, the training they have, and the post-processing they do.

In addition, final videos will be made and a survey will be conducted to obtain a more qualitative assessment.

Main goal

The main goal will be to execute these methods, study them deeply and get inputs that will challenge the method, for example, scenes with very high illumination or images with low resolutions and reduced quality. With this, we will be able to see how the different pipelines perform and how they solve, if they do, the problems they encounter.

To reach this objective, the first thing to do will be to search the Internet for current methods that are in use and publicly available. The criteria used is that the objective they have is one of the following, face swapping, deep fake, face animation, or face edition. Then they must be recent and up to date and finally, the pipelines they use must be different from each other.

Once the methods are in place, the next step is to look for challenging inputs, i.e. high illuminance, low temporal coherence, or low resolution.

Following this, we will run the methods and obtain results to show to the public and get feedback for comparison on the quality of the resulting video, see if the change is noticeable and see which looks better.

Chapter 1

Related work

1.1 Deep Learning

Deep Learning is a set of algorithms of machine learning in which we define a model as a function with parameters as inputs to our operations. Inside we define a layer model, in which we will have three types, input layer, hidden layer and output layer Fig 1.1.

And we try to reach an optimal solution using the procedure:

1. We repeatedly enter the observations(inputs) we have through the model and save the values of the weights computed through the "forward pass" process.
2. We calculate a *loss* representing how different the predictions obtained from the model are from the outcome we expect.
3. Using the weights calculated in step 1, we compute new weights for the new inputs, which are related to the loss calculated above.
4. We update the parameter values and pass them to the next model so that it can calculate a loss that favors us.

At the beginning of our model, we start with a series of linear operations to transform them into features of our destination through iterations. Then, in our model, we also define a structure-function to apply these operations. We also apply non-linear functions

(*sigmoid*) and finally a set of linear operations for the last layers. With these modifications, we will obtain learning results closer to the expected.

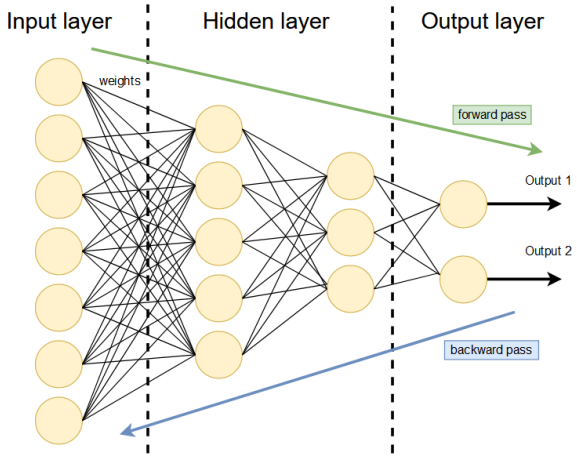

Figure 1.1: Basic Diagram of Deep Learning.

Deep learning is a fundamental part of the methods we will see ahead because it is what allows us to detect faces, recognize facial features and create masks of our destinations. And it also allows us to reassemble and join the source and destination faces together.

1.2 CNN

One thing that deep learning algorithms use a lot is neural network architectures. CNN is one of them which reads the data directly. It is especially useful for finding patterns in images to recognize objects. This network has three layers, a convolutional layer, a pooling layer, and a fully connected layer Fig 1.2.

In the convolutional network is where the greatest workload is performed, its task is to make the dot product between two matrices, the kernel and the other is a part of the input. While we do the forward pass in this layer, the kernel will go through all the parts of the image that it has as input and create a feature map as a result.

Then in the pooling layer, we reduce the data obtained as input from other layers to be able to keep the most important, in our case would be to keep the landmarks of the face.

Finally, in the fully connected layer we have neurons that help us to classify our results

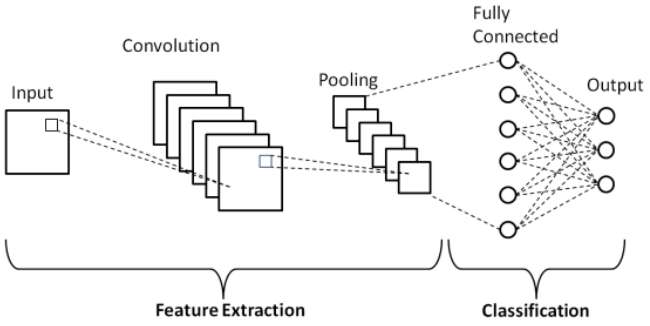

Figure 1.2: Diagram of CNN.

and with the help of these in the backward pass correct the weights of the layers and produce better results.

1.3 GAN

GAN (Generative Adversarial Networks)[8] is useful for us because it allows us to generate data in a very realistic way, always based on real data. With this, we can create realistic human faces, songs, and movies and even enhance images with the help of other images.

To achieve this we generate a structure that allows us to have as input samples to be able to train this structure and obtain an output more in line with what we want. The structure consists of a discriminator and a generator Fig 1.3. The generator's job is to replicate the input in the best possible way and pass it to the discriminator. And the job of the discriminator is to detect whether the images from the generator is the real image or a false one.

Once our discriminator does not know whether the input is real or false, we will know that we have reached our destination.

1.4 Deep Fake

Deep fake is an AI technique that allows us to replicate a person's face and blend it directly with another person's face. To do this we make use of an autoencoder, similar to GAN,

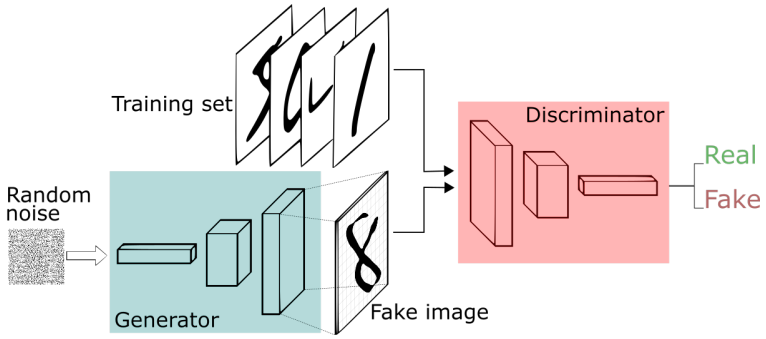

Figure 1.3: Basic Diagram of GAN.

which is two neural networks that compete with each other to adjust the result as best as possible.

1.5 Autoencoder

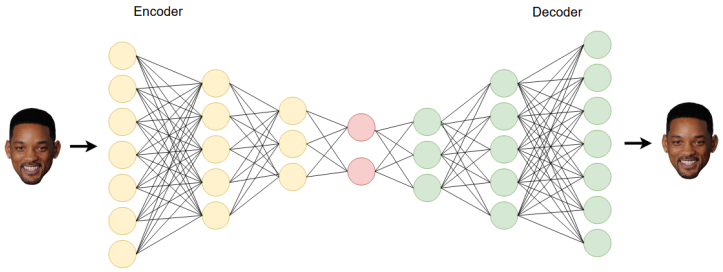

Figure 1.4: Basic Diagram of Autoencoder.

The task of the autoencoder is to have an image as input and as output, we will have the same image, the work is that the autoencoder itself has to know how to generate and replicate the image it has as input. For this purpose, two neural networks are created and positioned to create a bottleneck (encoder and decoder) Fig 1.4. The encoder part has to learn which are the characteristics of the image to be able to encode it and the decoder part has to learn how to decode the previous result to obtain an image. For the deepfake, an encoder is generated from the source face and the result is fed to the decoder of a different person.

Chapter 2

Methods

In this chapter we will see the selected methods, these have been selected because they fulfill the purpose of changing the face or editing it, besides that they are recent and popular. Those selected were Deep Face Lab [15], CoMotionSegment [22] and STIT [25].

Each of the methods is totally different and implements alternative ways of the main goal.

2.1 Deep Face Lab

Deep Face lab [15] is an open-source software that allows you to perform face-swapping. In summary, this method can detect and extract the faces of a video, then blend it into the destination video. Also, we can do a post-processing session with the results to sharpen the final video. Unlike the others, this one is more dedicated to the end-users. It is one of the most popular methods in its field and has a large community behind the project.

2.1.1 Pipeline Definition

This pipeline is divided into three phases, **extraction**, **training** and **conversion**. These parts are sequentially and the kind of data they handle is only the source and destination videos.

Extraction

On the extraction part, we have the methods to take the frames of the video and first detect the faces, get the landmarks of the face and finally extract the face.

The first step in the extraction is to detect the face, to do that is used an S3FD [31] as a default detector, later while we are executing the software we can decide which face detector we can use.

Once we have the face we can proceed to face alignment, in this process we look for facial landmarks because they are the key to maintaining stability over time. An effective algorithm is essential so this method provides two algorithms to solve this.

- Heatmap-based algorithm for facial landmark algorithm 2DFAN [2], this is useful for a face with a standard pose.
- PRNet [6] with 3D face information.

After we get the facial landmarks a function is made to smooth these landmarks in consecutive frames to ensure stability further.

And with all this is also applied a point pattern mapping and transformation method by Umeyama [26], this is used to calculate a similarity transformation matrix for face alignment. DFL provides also a facial landmark template to calculate this matrix.

With face alignment data from both sides, source, and destination, the next step is **Face Segmentation**.

A TernausNet [10] is used to successfully segment it, even if the face has hair on top or wears glasses. In addition, we can use XSeg which allows us to train the model for a specific set of faces.

We can see in the following image a diagram of the extraction phase Fig 2.1.

Training

The training phase is the most important of the whole method, it is thanks to this that we can achieve very realistic results. Once we have all the data received from the previous phase, we can proceed with the training.

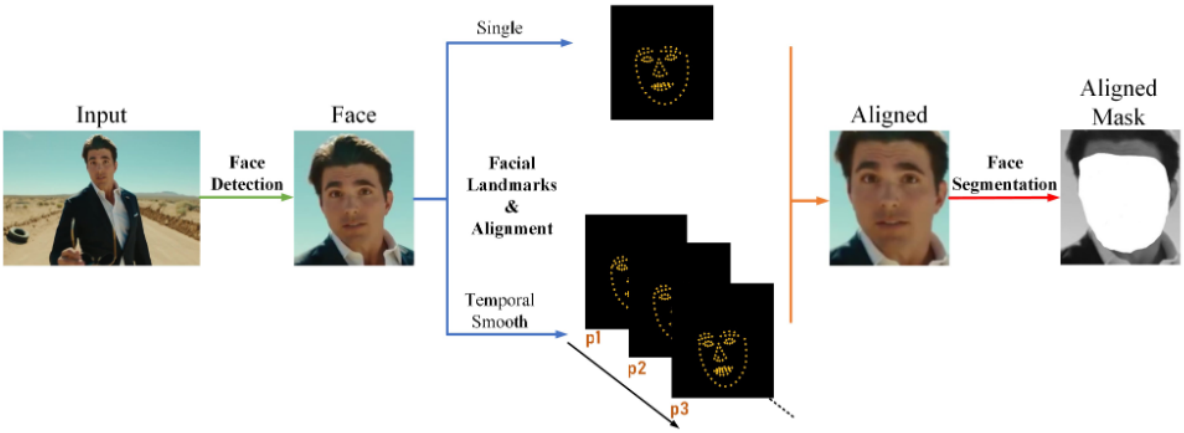

Figure 2.1: Extraction Phase

The two sides to be used don't need to be alike, DFL provides two training methods to solve this, Structure DF and Structure LIAE. Fig 2.2.

DF Structure

It is an encoder along with an intermediate layer, which shares weights between source and destination, two decoders, one for the source and one for the destination. With this structure, we can achieve a generalization of the two faces.

The algorithm can do a perfect training but in the case of having very specific things for the destination face it fails because it cannot inherit too much information, an example of this can be the external lights in the frame we are looking at.

LIAE Structure

Therefore, it is proposed to use this one, which is more complex and can collect extra information.

Shares one encoder and one decoder for both sides, in between uses one inner layer per image. From these two layers, the one that belongs to the source has as input the data of the own face but also includes data from the destination face. On the other hand, the intermediate layer from the destination image only uses its own information.

After this process, we have the combined features of each image (F_{src}^{AB}, F_{dst}^{AB}) and we will link the features respectively to the images.

To source image we get $F_{src}^{AB} \parallel F_{src}^{AB}$, we do the same for the destination $F_{dst}^{AB} \parallel F_{dst}^{B}$ and this is input to the decoder.

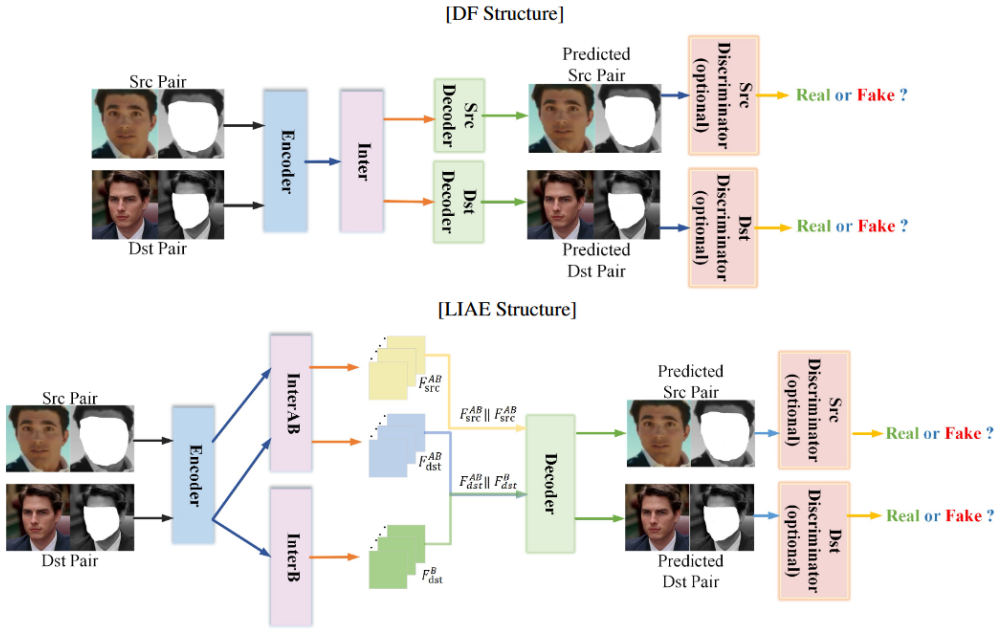

Figure 2.2: Training Phase

As a final step, it is optional to include a discriminator to help us classify the results.

As an extra, we can add SIMM [28] to set different weights depending on previous training.

Conversion

The purpose of this step is to mix the two sides, the method can blend the source face to the destination face and vice versa.

For the first case, what they proposed is to transform the source face generated, with the mask obtained, with the decoder of the destination face, then we proceed to blend it and make it look good with the rest of the photo, for which DFL provides several algorithms (RCT [17], IDT [16] , etc). For the other case, it would be the same but changing the order.

We can see a diagram about the conversion phase in Fig 2.3.

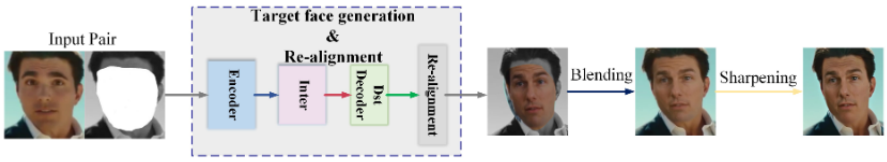

Figure 2.3: Conversion Phase

2.1.2 Post Processing

Once we have executed the code and followed the whole process, the developers of the method provide us with some useful post-processing tools to improve our result.

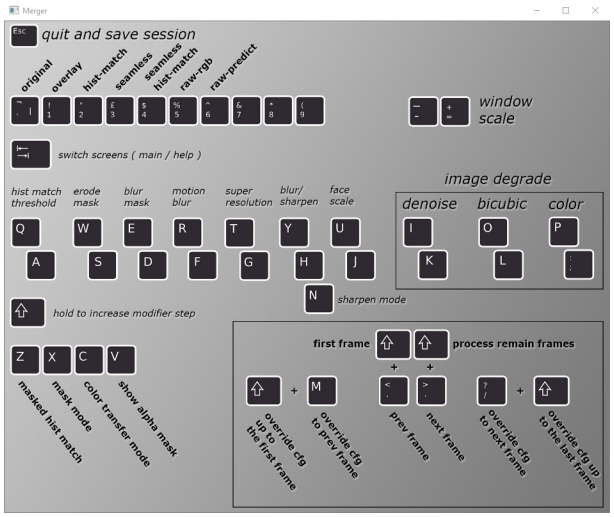

Figure 2.4: Post processing options

In Fig 2.4 we can see the different options available to us. The most useful and the ones I have personally used are a blur and motion blur. These two by setting the correct values help to improve the images and make them more realistic.

2.2 CoMotionSegment

This method is based on an extra implementation of another method, First Order Motion Model [21].

The original idea of this method is to animate an image based on a moving video destination. From a source image, we will extract a visual representation, the appearance representation of an object. And for the destination video, we extract a semantic representation, combining these two representations we will have the final video.

However, when we do this process, we may encounter a *leaking* problem. This occurs when our network does not have a good bottleneck defined. With this problem, the semantic representation will be contaminated by the appearance representation giving us very poor results.

The implemented solution of the method can predict segments associated with the destination frame and use them with the motion representation to reconstruct the resulting image.

The method is designed to have as input a single image and a video to match. Also, the model trains the data in pairs with frames from the destination video, where these frames are randomly selected.

2.2.1 Pipeline Definition

The pipeline is defined in two modules, the Segmentation Module and Reconstruction Module Fig 2.5.

For the explanation of the pipeline we define: $X_S \in \mathbb{R}^{3xHxW}$ and $X_D \in \mathbb{R}^{3xHxW}$ as images where H and W are the corresponding heights and widths.

Segmentation Module

This module is in charge of extracting the segments and the movement of the destination video. In each segment, we group the pixels of the image that correspond to the parts that move together. It is also segmented with an affine transformation on the support corresponding to each segmented part. An optical flow is represented by $\mathcal{F}$.

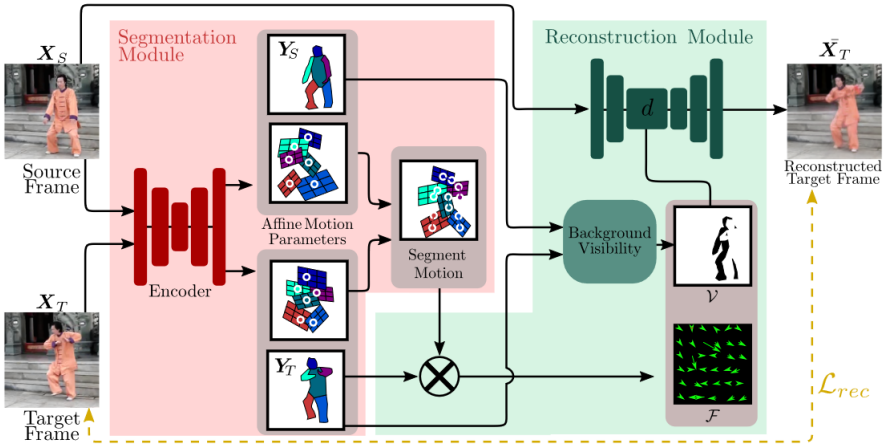

Figure 2.5: Diagram of the pipeline

The encoder task in this part will be to provide segments in which the affine model is valid.

A neural network is used to segment the frames independently from X_S and X_D into $K+1$ parts, where K represents the image segments.

From here we represent $Y_S^k \in [0,1]^{K+1xH'xW'}$ and $Y_D^k \in [0,1]^{K+1xH'xW'}$ each one associated with source and destination, k represents the segmentation channel. The values of this tensors are binary which represent, 1 the pixel is inside the segment or 0 if it is outside the segment. This is defined as $\mathcal{Y}_D^k$ where:

$$\mathcal{Y}_D^k = \{z \in [1,H']x[1,W'] | \mathcal{Y}_D^k[z] = 1\}$$

The same applies to $\mathcal{Y}_S^k$ for the source segmentation.

Taking into consideration what has been said above with the movement of each segment, it defines $\mathbf{A} \in \mathbb{R}^{2x2}$ and $\beta \in \mathbb{R}^{2x2}$ where:

$$\forall z \in \mathcal{Y}_D^k, \mathcal{F}(z) = \mathbf{A}z + \beta$$

And this formula can be rewritten for the method like this:

$$\forall z \in \mathcal{Y}_D^k, \mathcal{F}(z) = \mathcal{P}_S^k + \mathbf{A}_S^k {\mathbf{A}_D^k}^{-1}(z - \mathcal{P}_D^k)$$

where $\mathbf{A}_S^k \in \mathbb{R}^{2x2}$, $\mathbf{A}_D^{k\ -1} \in \mathbb{R}^{2x2}$, $\mathcal{P}_S^k \in \mathbb{R}^2$ and $\mathcal{P}_D^k \in \mathbb{R}^2$ are approximate data given by segmentation network.

To reduce the computation time, small resolutions are used and a U-Net [20] type network is implemented along with a softmax at the end to obtain tensors that can be interpreted as confidence maps.

Reconstruction Module

This is the second module and obtains as input the result of the previous module. The task of this module is to reconstruct the destination frame using the source frame and the segmentation motion obtained from the previous module.

With this, we obtain an optical flow between the source and destination frames. We have $\mathcal{F} : \mathbb{R}^2 \to \mathbb{R}^2$ which is a map where we see which pixel X_D corresponds to X_S

In other words, it is the estimated animation between the source and destination frames.

This value is obtained by combining the segment motion representation and the predicted segmentation, both obtained from the segmentation module. For each segment is obtained K optical flow fields $F^k \in \mathbb{R}^{2xH'xW'}$ of $\mathcal{F}$ in the respective segment $\mathcal{Y}_F^k$, the tensor obtained as $\mathbf{F}^k$ is interpreted as the partial optical flow in the pixels of the segment $\mathcal{Y}_D^k$ between source and destination frames.

It is important to know that the last optical flow we get is the one associated with the background.

With all this, we can define that each part of an object corresponds to a segment in $\mathbf{Y}_D$ and that its motion can be encoded by its corresponding motion descriptor. The following discretized estimator $\hat{\mathbf{F}}$ of $\mathcal{F}$ is obtained by:

$$\hat{\mathbf{F}} = \sum_{k=1}^{K+1} \mathbf{Y}_F^k \otimes \mathbf{F}^k$$

In this formula, we have $\otimes$ which denotes the element-wise product. To summarize, the encoder task is to obtain a segmentation mask that allows us to correctly assign the partial optical flow $\mathbf{F}^k$ to each location z, where z is $\mathbf{F}^K + 1 = z$ (background).

As a result, this segmentation mask must gather the locations that move in the same

direction.

Something else we get are the masks for the background, using the segmentation map we can differentiate between background and foreground. $\mathbf{V} \in [0,1]^{HxW}$. This would represent the background of the frame we are in and therefore, pixels to avoid. Also to estimate the locations on the foreground it is done by merging K objetcs parts $\mathbf{O}_S = \sum_{k=1}^{K} \mathbf{Y}_S^k$. The locations not included here will correspond to the background ($\mathbf{Y}_D^{K+1} = 1$).

With all this, we obtain:

$$\mathbf{V} = 1 - (\mathbf{Y}_D^{K+1} \otimes \mathbf{O}_S)$$

Finally, the destination frame is reconstructed using the source with an encoder-decoder network. For this purpose, the features of the source image are aligned using the pose of the destination object using the optical flow result.

The generator network (G) reconstructs the destination frame by warping the source frame features along with the estimated optical flow. The network incorporates an encoder-decoder with a deformation layer inside. Going deeper into this layer, it takes as input a feature map $\xi \in \mathbb{R}^{CxH'xW'}$, the optical flow $\hat{\mathbf{F}}$ and the background visibility mask $\mathbf{V}$. The use of $\mathbf{V}$ is to mask out the feature map locations that are not visible. This layer is defined by:

$$\xi' = \mathbf{V} \odot \mathcal{W}(\xi, \hat{\mathbf{F}})$$

$\mathcal{W}$ denotes the back-warping operation and $\odot$ denotes the Hadamard product. This operation is implemented using a bilinear sampler [11].

2.2.2 Training

The training is based on the perceptual loss of Johnson et al [12]. and use a pre-trained VGG-19 [23]. Takes as input a destination frame $\mathbf{X}_D$ and the reconstructed frame $\bar{\mathbf{X}}_D$. The loss is defined by:

$$\mathcal{L}_{rec}(\bar{\mathbf{X}}_D, \mathbf{X}_D) = \sum_{i=1}^{I} |\Phi_i(\hat{\mathbf{X}}_D) - \Phi_i(\mathbf{X}_D)|$$

Where Φ_i denotes the i^{th} channel feature from a specific VGG-19 [23] layer and I denote the number of feature channel in the actual layer.

2.3 Stitch it in Time

This method arises from the study and improvement of previous GAN-based methods for face edition. The method proposes a framework for semantic editing of faces in videos also producing meaningful face manipulations, maintaining a higher degree of temporal consistency, and can be applied to challenging, high-quality, talking head videos.

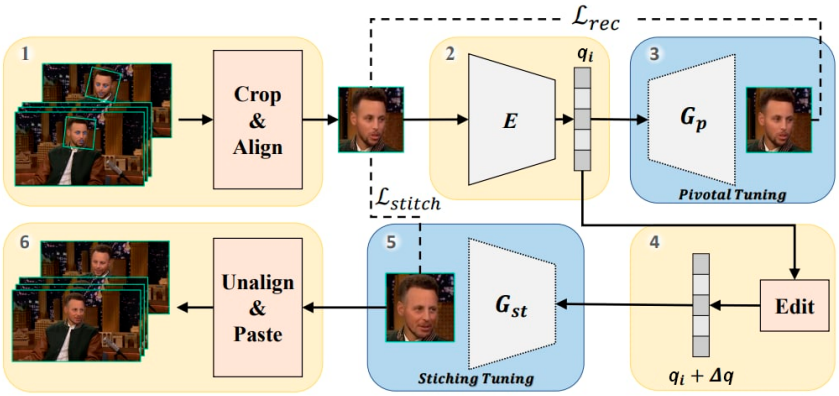

Figure 2.6: Diagram of the pipeline

2.3.1 Pipeline

The design pipeline consists of 6 steps (Fig 2.6), temporally consistent alignment, encoder-based inversion, generator tuning, editing, stitching tuning, and merge.

Alignment

A pre-trained model [13] is used, where each image was aligned and cropped around the face. It also uses a gaussian lower pass filter [7] on the landmarks, the smoothing that they implement is sufficient to overcome any inconsistencies induced by the alignment step.

Inversion

The next step will be to invert the face into the latent space of GAN. This is done using PTI [18] which discovers the first pivot of latent code that reconstructs approximately the

input image in the most editable areas. Once this is done, fine-tune the generator to the same pivot code to generate a more accurate version of the destination image. Also, PTI helps to assist to maintain temporal coherency.

In practice, however, we can see that PTI's editing will cause inconsistencies with the pivots themselves which can manifest themselves in two ways.

First, if the pivots are far away from each other in the latent space code the edition becomes less consistent. The second case is when PTI has to fix attributes of the face these fixes will not be taken into account in subsequent processes, this causes some frames to be different from each other.

This is solved by replacing the initial inversion of PTI (finding the pivot) to an encoder-based version [24]. It is expected that this includes improvements such as strong smoothness.

Formalize that given N frames from source video $\{X_i\}_{i=1}^N$, the cropped images are defined as $\{c_i\}_{i=1}^N$. First the e4e [24] encoder E is used to obtain the latent inversion $\{q_i\}_{i=1}^N = \{E(c_i)\}_{i=1}^N$. These are the latent vectors that will be used for PTI.

Therefore, we have $r_i = G(q_i; \theta)$ what are the generated images from q_i with a generator G and weights θ.

Then the PTI objective is defined by:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} (\mathcal{L}_{LPIPS}(c_i, r_i) + \lambda_{L2}^P \mathcal{L}_{L2}(c_i, r_i)) + \lambda_R^P \mathcal{L}_R$$

Where $\mathcal{L}_{LPIPS}$ is the LPIPS perceptual loss [30], $\mathcal{L}_{L2}$ is the pixel-wise MSE distance and $\mathcal{L}_R$ is the locality regularization [18]. Finally, $\lambda_{L2}^P \lambda_R^P$ are constants across all experiments.

Editing

With a consistent set of inversions, it is time to move on to editing them. The model itself is expected to apply sufficient and consistent edits near the latent codes. Applying temporally-smooth pivots using StyleGAN [13] they are waiting for this change to be implemented sequentially.

Formalizing, the semantic latent editing direction will be δq, using PTI-weights θ_p we can

obtain the edited frames $e_i = G(q_i + \delta q; \theta_p)$.

Stitching Tuning

In the final step, the edited face is inserted e_i into the original video. They do not overwrite where the face is originally located as this would be insufficient, it is in the editing process that they take care of blending the background correctly so that it does not look bad. Another observation seen with previous work is that limiting face area modifications using the segmentation mask causes Poisson blending, also we can obverse inconsistencies around the border.

So they propose to use a novel tuning technique, stitching tuning Fig 2.7.

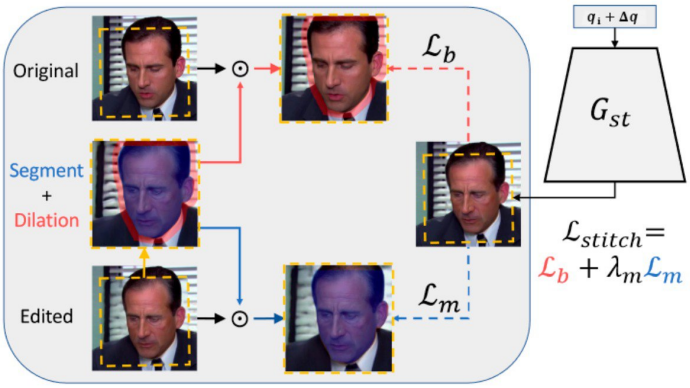

Figure 2.7: Diagram of stitching-tuning

For each edited frame, a boundary is designated at the edge of the segmentation mask. Then briefly fine-tune the generator around the edited pivot, all of this with a dual objective. The first would be to restore the boundary to pre-investment values, this is to blend it right into the source frame.

The second thing we want to do is to maintain the results of the editing, we will do this by requiring similarity to the edited frame in the area covered by the segmentation mask.

This is done by first using a off-the-shelf pre-trained segmentation network [29] to produce segmentation masks for all frames $\{m_i\}_{i=1}^{N}$. Each mask is then dilated to obtain a series of expanded masks $\{m_i^d\}_{i=1}^{N}$. Next, the boundary is defined as the element-wise xor operator

and defined by $\{b_i\}_{i=1}^{N} = \{m_i \oplus m_i^d\}_{i=1}^{N}$.

We also define as $s_i = G(q_i + \delta q; \theta_{st})$ which is the output of the stitching tuning procedure.

So, the first thing is to blend the boundary to the source image:

$$\mathcal{L}_{b,i} = \mathcal{L}_{L1}(s_i \odot b_i, X_i \odot b_i)$$

Then, we preserve the results of the edition.

$$\mathcal{L}_{m,i} = \mathcal{L}_{L1}(s_i \odot m_i, e_i \odot m_i)$$

The loss terms used to optimize the generator weights θ_{st}.

$$min_{\theta st}\mathcal{L}_{b,i} + \lambda_m \mathcal{L}_{m,i}$$

Where the weights are previously initialized with the PTI weights and λ_m is a constant.

Merge

Finally in this step is performed the inverted alignment and stitch for each frame s_i to the original image X_i using the dilated masks $\{m_i^d\}_{i=1}^{N}$.

Chapter 3

Testing and Evaluation

In this chapter, we will see how to execute the different methods, and the interface if it exists. We will also see the specifications of the computer that has been used to obtain the results and a series of problems to emphasize during the execution.

3.1 Technical Specs

The technical specifications are from my personal computer, which are as follows:

- CPU AMD Ryzen 5 3600 3.6GHz
- RAM Kingston FURY Beast DDR4 2666MHz 16GB
- SSD WD Blue SN550 1TB NVMe M.2
- GPU EVGA GeForce RTX 2060 SC OVERCLOCKED 6GB GDDR6
- SO Windows 10 pro

Thanks to this I have been able to run the methods with a few problems.

3.2 Input

To obtain the results it is first important to have the input of what we want to obtain.

For the first method, DFL, 4 clips from 4 different movies have been selected. To do a face-swapping between men I have taken a clip of Tommy Wiseau from the movie The Room and another one from the movie The Disaster Artist with James Franco.

The point of choosing these two films is because the second one, The disaster Artist, is a mockumentary about the production of The Room. With this, I want to see if it is easy for him to do his task having as a source clip with a bad resolution (480p) and with a destination of much higher resolution (1080p).

On the other hand, to see if the method may have flaws when dealing with women's faces, two clips have also been chosen, one from the series Euphoria with Zendaya and the other from the movie Kimi where we can see Zoe Kravitz.

These 4 source clips can be found at the following links.

- The Room - https://youtu.be/SXpxvsyB3gE
- The Disaster Artist - https://youtu.be/PnOS408-LTM
- Euphoria - https://youtu.be/yMJsFM5fCj4
- Kimi - https://youtu.be/K5Spd6a7xtk

Following the second method, CoMotion, a source image of the actor Robert Downey Jr. has been selected (Fig 3.1) and a video of myself making gestures so that the facial movements can be observed https://youtu.be/H61BO7HK3cM.

Figure 3.1: Input image for CoMotion method

And finally, for STIT the clips from The Room and Euphoria have been reused. This is because the first video is in low quality and I want to see if we are getting some glitches

because of that. On the other hand, in the Euphoria clip, we can see a lot of frames with so much high illuminance and that may cause trouble for the method.

3.3 Software

3.3.1 DFL

To execute the method DFL first we need to download the release that they have in their github https://github.com/iperov/DeepFaceLab.

They also let you run it on a collab but to use all the power of my computer I prefer to download it all directly.

Once we have downloaded it we can see two directories, the first one, called DeepFace-Lab..., is where all the software is stored. The other one is in case you want to use a set of pre-trained faces.

Following this, if we go into the first directory we will see a lot of bash executables. This is how developers make it easier for the user to use their program. Although, many of the executables are optional.

Another thing we also have are two folders, internal and workspace. The first one is where all the python files are located and where all are implemented, the second one is where you need to put your inputs and where you will see the output.

Now to be able to execute everything the first thing we need to do is to clean the workspace, for this we can proceed to execute the first bash file, *clear workspace.bat*.

Once we have this we can proceed to insert the videos that we will have as source and as destination inside the folder workspaces. It is important to identify which video is each one, for the source we will name it as data_src and the other as data_dst.

Having this placed in our workspace folder we can begin to perform the first step of the pipeline, the extraction.

Extraction

Here we will have to execute the following executables in order.

- *extract images from video data_src.bat*

  With this we can extract the images from the source video, the most important parameter that we find would be the *FPS* with which we can select the frame rate that we will choose, less value will mean fewer frames.

- *extract images from video data_dst FULL FPS.bat*

  Here we do not have an FPS option because we want to extract the video at the original frame rate.

- *data_src faceset extract.bat*

  This is where we extract the faces specifically. There is another file to do the same but manually, this one executes the face detector S3FD [31]. The most important parameter is the face type, which allows us to select the type of segmentation we want, Full face, Whole face, or only the head.

- *data_dst faceset extract + manual fix.bat*

  Same as before, only this one executes a manual pass where the model has not detected faces.

- *XSeg Generic) data_dst whole_face mask apply.bat and XSeg Generic) data_dst whole_face mask apply.bat*

  With these executables, we can apply the XSeg algorithm for face segmentation, although it is optional. It is also recommended to use these two *data_src mask - edit* and *data_dst mask - edit* to verify the mask and correct errors.

Once we have extracted the face of both videos we can proceed with the training.

Training

Before training, we can see where the training checkpoints and all their information will be saved. This can be found inside the workspace folder and then in the folder model.

For training, the developers allow us 3 different methods.

- SAEHD a High Definition Styled Auto Encode, useful for a user with GPUs with at least 6GB of VRAM

- AMP a model with a different architecture. This is still in development.
- Quick96 is a simple model useful for users with GPUs with 2 or 4 GB of VRAM. It has fixed parameters.

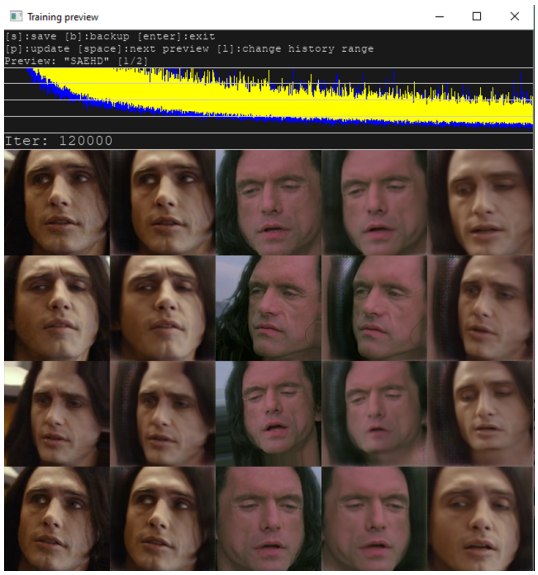

Figure 3.2: Information in real time about training.

The first method was used to obtain the results.

The most important parameter is Architecture, this allows us to select between LIAE and DF. A list of the parameters used can be seen in the appendix 6.

As a help to visualize the training we can find a window with plots telling us how it is performing in real-time Fig 3.2. At the top, we can see a graph that represents the accuracy between the two models. And at the bottom we have the following in columns: first the source face, then the synthesized version of the source image, next to the destination face, and following the synthesized version, the last one represents the synthesized output using both images.

It is important to have a fairly large series of iterations to get a decent-looking result. So I have done a training with a total of 120000 iterations, which took me a total of 12 hours.

Once we have it ready we can proceed to do the merge and get our result, for this we have to run the following script *merge SAEHD.bat*. Here we will see how we can select the

postprocessing 2.1.2 option and see how to optimize the final result.

Finally, to see the resulting video it is necessary to run *merged to mp4.bat* or *merged to avi* depending on the extension you want to use. We will get our video named result in our workspace folder.

3.3.2 CoMotionSegment

To be able to execute this method we will have to download the git version `https://github.com/AliaksandrSiarohin/motion-cosegmentation`.

Unlike the previous one, this one does not have different scripts to run the method. So we will have to install a fresh version of python and run *requirements.txt*. For this project we will use the previously trained models, these can also be found in the method repository.

Once we have everything installed we can proceed to execute the code.

The command to be used is as follows:

python part_swap.py –config config/vox-256-sem-10segments.yaml –target_video videos/video_target.mp4 –source_image videos/Robert.jpg –checkpoint checkpoints/vox-10segments.pth.tar –swap_index 2 –result_video res/out.mp4

Pre-trained models expect to receive a 256 x 256 photo and a video of a similar resolution. For this reason, the configuration to be used will be *vox-256-sem-10segments.yaml*, this means that from the 256x256 photo we will segment it into 10 parts. Therefore, we must also select the corresponding checkpoint.

Another thing to select is which part of the segmented image we want to insert in the final video. To know which segment to use, we can use the notebook provided by the developers, which can be found in their repository.

Following the steps and configuring it we can get the following image Fig 3.3, where we can see which segments are being selected.

Having already the segment we can proceed to execute the code. As it uses a pre-trained model the execution time is quite short compared to the previous one, getting the result in less than 10 minutes.

Figure 3.3: Segments from the source image.

3.3.3 STIT

As in the previous one, we will also have to use a new version of python and download the project from its repository https://github.com/rotemtzaban/STIT.

In this method when working with images the first thing we have to do is to choose the video we want to input and get its frames. To do it, I reused the scripts of the DFL method 3.3.1, precisely the first two scripts that extract the frames.

Once we have it the first thing we have to do is to get our inverted images. To do this we will use the following command:

python train.py –input_folder ./datasets/disasterartis –output_folder ./training_results/james –run_name james –num_pti_steps 80

With this, we would be doing the first two phases of the pipeline. A good value for PTI will be around 80 - 100 steps.

Once we have them we can proceed to make the edits. There are two scripts for editing, the first one is for native editing, *edit_video.py*, and the second one implements stitching tuning, *edit_video_stitching_tuning.py*. For the results obtained we have always used the second option so that we can evaluate the entire pipeline.

There are different options for editing, age, eye distance, eyebrow distance, eye ratio, open eyes, change gender, lips, mouth open, mouth ratio, the distance between mouth and nose, nose ratio, smile or yaw.

The ones I have used are age and gender. The functions used were exactly the following:

python edit_video_stitching_tuning.py –input_folder ./datasets/tommy –output_folder ./edits/theroom –run_name tommy –edit_name age –edit_range 8 8 1 –outer_mask_dilation 25 –border_loss_threshold 0.005

python edit_video_stitching_tuning.py –input_folder ./datasets/zendaya –output_folder ./edits/zendaya –run_name zendaya –edit_name gender –edit_range 6 6 1 –outer_mask_dilation 30 –border_loss_threshold 0.005

Both require their inversions to be obtained first. The dilation mask parameter, in my case, always has to be less than 30, otherwise, the program starts to crash after making 3 images due to a lack of virtual memory.

Finally, the resulting video will be found in the assigned folder, however, this video must be trimmed as the result is combined with the original Fig 3.4.

Figure 3.4: Result video, in order we have: original, with mask applied and stitching tuning

To obtain a clip with only the result the following command has been applied.

ffmpeg -i out.mp4 -filter:v "crop=1920:1080:3840:264" -c:a copy age.mp4

3.4 Personal evaluation

Once the results are obtained, I can make my evaluation.

The first thing to note is that all 3 methods have been able to achieve the objective without obtaining a fatal result. If we are talking in terms of quality, we can see that DFL and STIT performed very well. CoMotion has the limitation that it is prepared to work with low

resolutions, although it is true that it could be achieved that would involve modifying the code and doing more in-depth work.

In my case, I am studying it with an approach of a person with an intermediate knowledge in computer science, just enough to be able to compile and execute.

Talking now about how easy or difficult the execution has been, I have to say that with Windows it has not been very difficult but with some eventual failures. With DFL for example I had to expand the virtual memory because it exceeded what I had at the moment. Now during the execution process, I have also encountered errors that are quite notorious. In DFL for example, using the clips with women when using the XSeg we can observe that there are segmentation errors Fig 3.5.

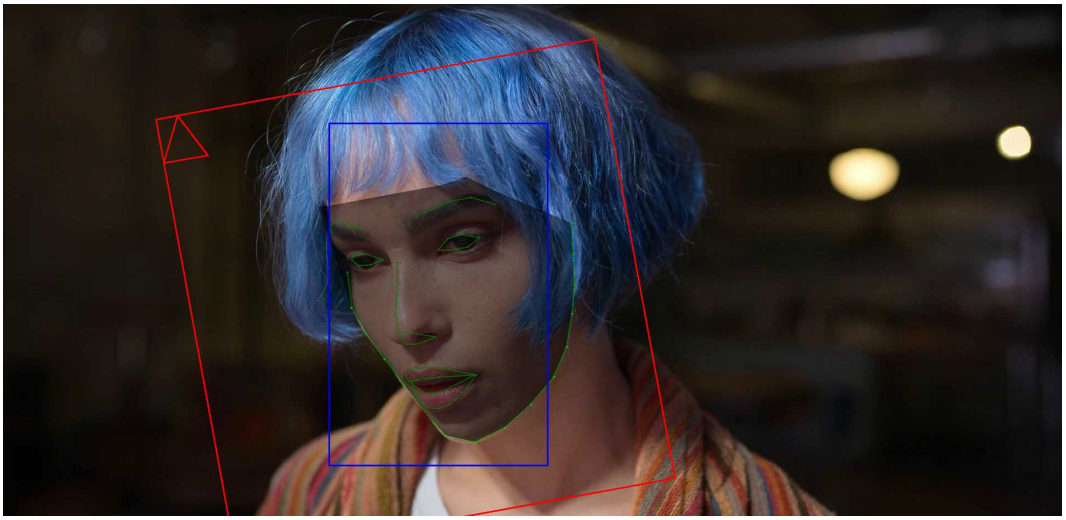

Figure 3.5: XSeg with Zoe Kravitz

The green line shows the segmentation performed by the algorithm. You can see how it captures the face well but there is a part of the hair above that it detects as if it were when it should be outside. This is nothing new as the developers themselves allow you to make changes to it and improve it manually.

With the STIT method I have also noticed failures when creating the inversions, there are random frames where it produces an error when loading the tensors, looking for the error in the source code and documentation I have not been able to find the cause, but, having frames to spare with the fact of eliminating that particular image was solved. Also in

the final result, we can observe some visual glitches. Investigating inside the code I can observe that it is because the generator detects glasses on the person when the person in question has his eyes closed.

I conclude with a comparative table 3.1 between the different methods.

	Quality	Face detection	Face Blending	User Friendly
DFL	7	8	9	10
CoMotion	4	7	7	3
STIT	9	6	9	4

Table 3.1: Table Comparitive

The score has been considered with 1 as the worst and 10 as perfection.

Chapter 4

Survey

With a personal evaluation, it would not be fair to make that the final comparison. That's why to get more feedback on the results I created a form and got a total of 59 results. This form can be found at the following link https://forms.gle/QbxUuJ77Yv14u5ct9.

4.1 Inputs and parameters

The results used for this purpose were: Face swap with DFL with the male clips, then two STIT clips using the age and gender edits, and finally a clip of CoMotion using me as the destination video.

These videos can be found at the following links:

- The Room - DeepFake with James Franco - https://youtu.be/9--PlkeBBbc
- Old Tommy Wiseau - https://youtu.be/bHgY1DyUZps
- Zendaya male - https://youtu.be/Qd1zKyCU5Q4
- Changing Mouth - https://youtu.be/gGlzzM-4AyI

4.2 Graph with results

In this section, we will discuss the results obtained from the survey with your question.

Viewing the Face Swap video between Tommy Wiseau and James Franco, it was asked what opinion there was about the quality Fig 4.1.

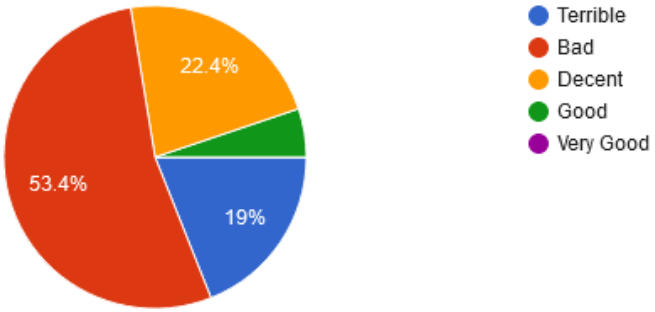

Figure 4.1: In terms of quality, how would you rate the video?

We can see that more than half of the people who answered the survey found the quality of the video to be bad. Then some people thought the video looked decent and some thought it was terrible.

The next question was about whether it was easy to detect that it was a deep fake Fig 4.2.

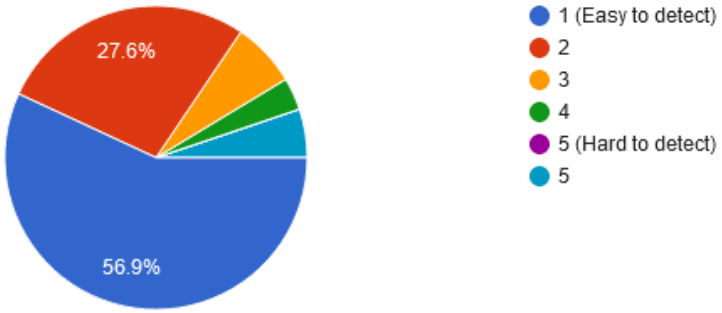

Figure 4.2: On a scale of 1 to 5. How easy was it to detect that it was a deep fake?

Most people say that it was easy to spot the edit.

Next, the question asked is more specific and asks which aspect of the face is the least matched to the target face Fig 4.3.

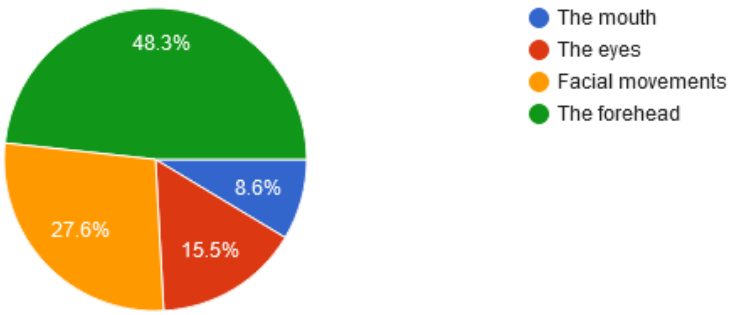

Figure 4.3: Focusing on the face, which is the worst match to the original face.

We can notice that most of the results note that the worst part is the forehead. And the next worst thing would be the facial movements.

Then, the participants were shown two photos with the original faces and asked if a great job was done in adapting the faces Fig 4.4.

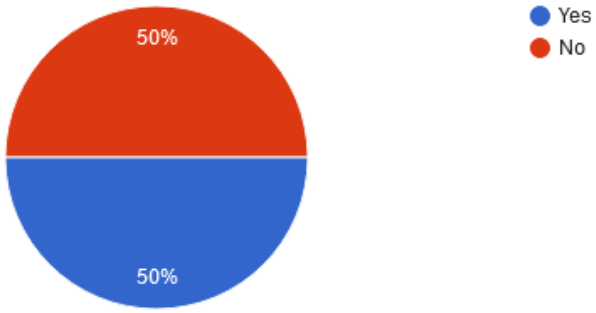

Figure 4.4: What was the worst thing about the face?.

We see that the opinion of the public is divided on this question, half of them think that the work that was done is correct and good. And the other half thinks just the opposite.

After this, we move on to the STIT method using the original clip with Tommy Wiseau as a reference. The audience was first asked to compare the quality of the video Fig 4.5.

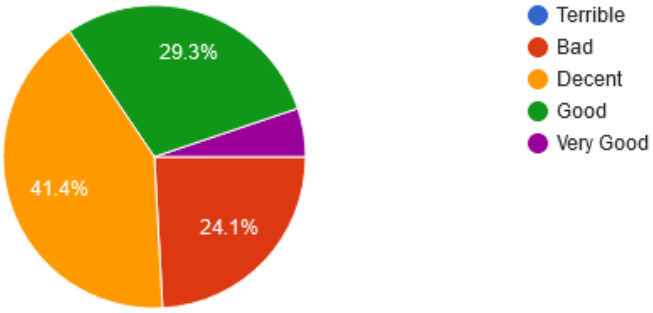

Figure 4.5: In terms of quality, how would you rate the video?

We can observe that a large number of people watch the video with decent or good quality. Some people consider it bad. Still, it is much more positive than the DFL video.

The video shown was of the actor Tommy Wiseau aging, so the public were asked to guess how old the person in the video appeared to be Fig 4.6.

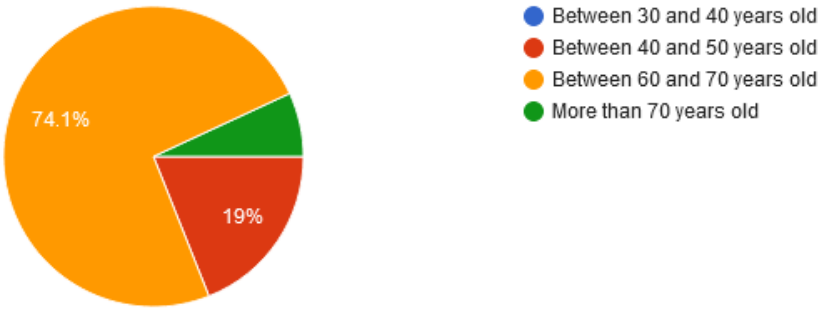

Figure 4.6: How old does the actor look now?

Most people noticed that the person was older, which was the objective.

Following the same method but changing the video with the Zendaya clip, this time changing the gender. The first thing that was asked was whether the actress in question was recognized Fig 4.7.

Most of the audience could recognize the actress.

In this clip, there are quite a few frames where visual glitches appear, so the question was

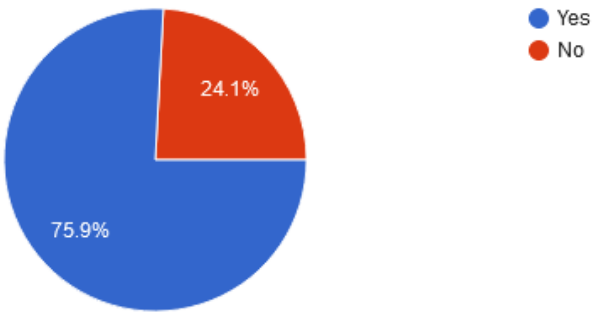

Figure 4.7: Can you tell which person is originally in the video?

asked if it had been noticed 4.8.

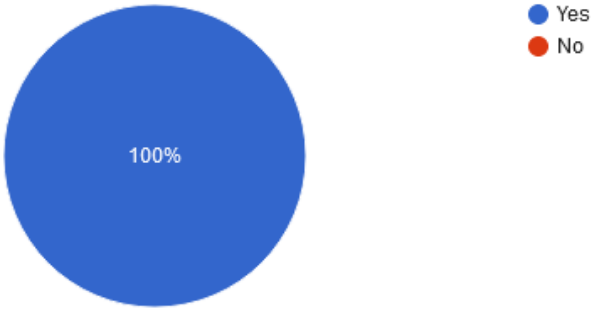

Figure 4.8: Do you notice any fault in the video?

The response was unanimous with a yes.

To finish with this method, it was asked whether a good job had been done in adapting the faces Fig 4.9.

Most people think the performance was good.

And finally, questions were asked based on the CoMotion method, the resulting video had as its destination a video of myself and an image of the actor Robert Downey Jr. as its source image.

The first, as with the previous methods, was to ask about the quality of the video Fig 4.10.

The vast majority of responses said it looked decent and a large number said it was of

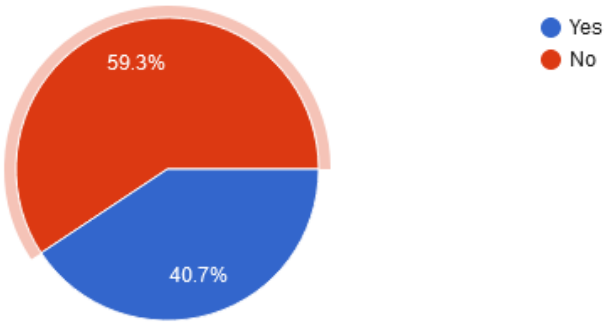

Figure 4.9: Do you think it has done the job of adapting the faces?

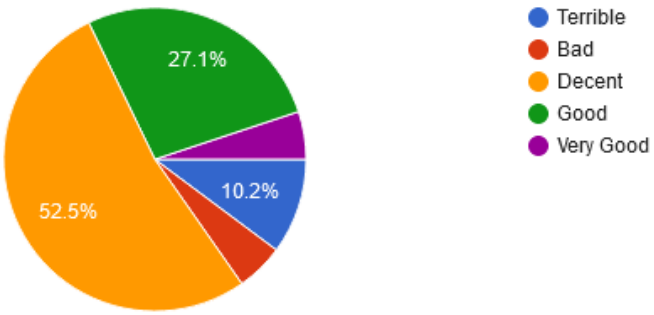

Figure 4.10: In terms of quality, how would you rate the video?

good quality. Even so, there are people who think that the result is terrible.

The most important thing about this clip was how the source image was combined with the destination video, especially with the chin and mouth. For this reason, a question was asked about the mouth movement in the final video Fig 4.11.

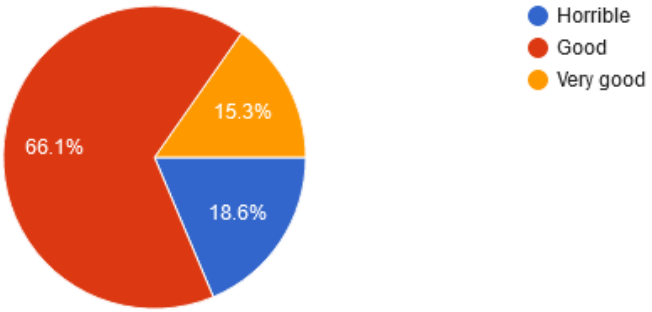

Figure 4.11: How would you rate the movement of the mouth?

In general, the public thinks that the result of the movement has been correct and only a small group of people think that it has been horrible.

Finally, the audience is asked which method was the best Fig 4.12.

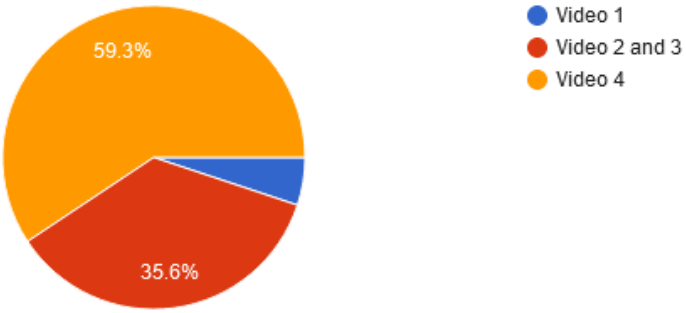

Figure 4.12: What do you think has been the best video?

As a result, the best method would be CoMotion, followed by the STIT method, and with very few results the DFL method.

Chapter 5

Ethics

Once we have compiled, executed, tested the methods, and obtained an evaluation of them, we can move on to talk about the implications of this technology.

So in this chapter, we will talk about the current state of facial editing, what is the social impact it causes and what the future looks like for it.

5.1 Current state of the technology

Having seen in depth these 3 methods and also looked over other methods, we can see that with the current technology we can do great things. However, we have a big quality problem to solve. An example of this is CoMotion which is intended to work with low-resolution inputs.

It is also nothing new for these projects, we can see that with the DFL they have been increasing the resolutions over time, so we can expect that in the future these methods will evolve and we will reach very good quality levels.

Also, to take some of the burdens off, we have to be selective with the kind of inputs we use, the inputs used in this state-of-art have been with the motivation to go looking for flaws. By this, I mean that you can make high-quality results using good inputs. Let's not forget also the fact of post-processing, the results of the methods can be very useful to have initial images and to be able to edit manually looking for a better result.

Another topic to talk about would be how user-friendly the software is. Of the three methods, the only one that gave great help to an end-user that barely knows about computer science is the DFL, this is important because by making it simply the use that can be given has a greater scope. However, this can also be seen in app implementations that use these methods but through a user interface.

5.2 Social Impact

As a result of this technology, we can find different uses and applications. We can find it in movies, making deep fakes with deceased actors, or making face editions to make actors younger. There is also the potential use in filters, that apply these techniques, for social networks, currently, there is a boom [3] with these filters and we can see it everywhere. It is incredible the great power that this technology has to generate entertainment content and entertain a large audience.

This is all well and good and that has been a big part of my motivation for this project, but, there is a huge aspect of the use that has not been discussed. This aspect is the pornography, As soon as you start researching these methods, especially those that have to do with deep fake, you may find that the major use is for deep fake, trying to recreate pornographic scenes with faces of celebrities or people they know, most of the content of deep fake is about women. And it is clear that this use causes many women to come upon pornographic photos or videos that are not real [9].

It is incredible to see the amount of data that there is on the deep fake aimed at this field, there are many datasets with gigabytes of information on the faces of celebrities and it is quite worrying that a major booster of these technologies is as a result of this.

And this is not the only misuse that is given to this technology, since the false speech of Obama in 2017 it has become popular to use these false speeches to defame politicians [19]. This causes the use of fake news and hoaxes that have grown a lot lately.

Chapter 6

Conclusions and further work

At the beginning of the project, the objective was to study methods and challenge them to see how they work deeply. All this to see how this technology was currently and what could be done with it. The first thing that was done was to choose 3 methods based on specific criteria. Study them in-depth to understand their pipeline and how they work and the next thing was to get inputs that can show where these methods fail.

The methods were then run to obtain results for comparison. From these results, the best ones were selected to make a comparison with them, a personal one and a general one by making a survey. And thanks to this comparison it has been possible to see the flaws of each method.

I believe that these objectives have been achieved as I am now able to understand the methodology of these methods and how to create effective facial edits. In addition, with the help of the comparative study, I can see and notice where they fail and what are the strengths of each method.

Apart from this goal, I have also been able to learn more about the use of this technology as well as to see the dark side of the community. I only knew the beginning of the social impact section 5.2 and now I am more aware of the misuse that can be made of it.

In further work, we have to detail more about these failures and overcome them to obtain perfection with the methods. One of the biggest flaws is the issue of quality, but work is already being done on this. Especially DFL has been increasing the resolutions and we

can see progress.

Another challenge that is also currently being worked on is the fact of applying these edits in real-time, which is necessary due to the use of filters and the use of current streaming content. That is why new methods are coming out focused on real-time editing [1].

Another issue to improve would be the implementation of user interfaces to make its use more general, as DFL is doing. Thanks to this, people with a basic level of knowledge could achieve good results without having to get a degree in the area.

Also, we are seeing great improvements with GANs [4] and these methods will be improved by using it.

Parameters for training in DFL

Iterations: 120.000

Resolution: 128

Face_type: f

Models_opt_on_gpu: True

Archi: liae-ud

Ae_dims: 256

E_dims: 64

D_dims: 64

D_mask_dims: 22

Masked_training: True

Eyes_mouth_prio: False

Uniform_yaw: False

Blur_out_mask: False

Adabelief: True

Lr_dropout: n

Random_warp: True

Random_hsv_power: 0.0

True_face_power: 0.0

Face_style_power: 0.0

Bg_style_power: 0.0

Ct_mode: none

Clipgrad: False

Pretrain: False

Autobackup_hour: 1

Write_preview_history: False

Target_iter: 120000

Random_src_flip: False

Random_dst_flip: True

Batch_size: 8

Gan_power: 0.0

Gan_patch_size: 16

Gan_dims: 16

For more information about the parameters go to the following link Section 6. Training https://mrdeepfakes.com/forums/threads/guide-deepfacelab-2-0-guide.3886/.

Bibliography

[1] Martin Anderson, *Real-time deepfake streaming with deepfacelive*, Aug 2021.

[2] Adrian Bulat and Georgios Tzimiropoulos, *How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks)*, 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, oct 2017.

[3] Elena Cavender, *Everyone is crying on tiktok. thanks, snapchat.*, May 2022.

[4] Boris Dayma, *Dall·e mini - generate images from any text prompt*, May 2022.

[5] Stav Dimitropoulos, *A new film with a dead star? thanks to cgi and deepfake technology, it's possible*, Nov 2020.

[6] Yao Feng, Fan Wu, Xiaohu Shao, Yanfeng Wang, and Xi Zhou, *Joint 3d face reconstruction and dense alignment with position map regression network*, 2018.

[7] Gereon Fox, Ayush Tewari, Mohamed Elgharib, and Christian Theobalt, *Stylevideogan: A temporal generative model using a pretrained stylegan*, 2021.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial networks*, 2014.

[9] Karen Hao, *Deepfake porn is ruining women's lives. now the law may finally ban it.*, Feb 2021.

[10] Vladimir Iglovikov and Alexey Shvets, *Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation*, 2018.

[11] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu, *Spatial transformer networks*, Advances in Neural Information Processing Systems (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[12] Justin Johnson, Alexandre Alahi, and Li Fei-Fei, *Perceptual losses for real-time style transfer and super-resolution*, 2016.

[13] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila, *Analyzing and improving the image quality of stylegan*, 2019.

[14] Rachel Metz, *The number of deepfake videos online is spiking. most are porn*, Oct 2019.

[15] Ivan Perov, Daiheng Gao, Nikolay Chervoniy, Kunlin Liu, Sugasa Marangonda, Chris Umé, Mr. Dpfks, Carl Shift Facenheim, Luis RP, Jian Jiang, Sheng Zhang, Pingyu Wu, Bo Zhou, and Weiming Zhang, *Deepfacelab: Integrated, flexible and extensible face-swapping framework*, 2020.

[16] François Pitié, Anil C. Kokaram, and Rozenn Dahyot, *Automated colour grading using colour distribution transfer*, Computer Vision and Image Understanding **107** (2007), 123–137.

[17] E Reinhard, M Ashikhmin, B Gooch, and P Shirley, *Color transfer between images*, IEEE Computer Graphics and Applications **21 (5)** (2001), 34 – 41 (English), Publisher: Institute of Electrical and Electronics Engineers Other:.

[18] Daniel Roich, Ron Mokady, Amit H. Bermano, and Daniel Cohen-Or, *Pivotal tuning for latent-based editing of real images*, 2021.

[19] Aja Romano, *Deepfakes are a real political threat. for now, though, they're mainly used to degrade women.*, Oct 2019.

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015.

[21] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe, *First order motion model for image animation*, Advances in Neural Information Processing Systems (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[22] Aliaksandr Siarohin, Subhankar Roy, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe, *Motion-supervised co-part segmentation*, (2020).

[23] Karen Simonyan and Andrew Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2014.

[24] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or, *Designing an encoder for stylegan image manipulation*, 2021.

[25] Rotem Tzaban, Ron Mokady, Rinon Gal, Amit H. Bermano, and Daniel Cohen-Or, *Stitch it in time: Gan-based facial editing of real videos*, 2022.

[26] S. Umeyama, *Least-squares estimation of transformation parameters between two point patterns*, IEEE Transactions on Pattern Analysis and Machine Intelligence **13** (1991), no. 4, 376–380.

[27] James Vincent, *Watch jordan peele use ai to make barack obama deliver a psa about fake news*, Apr 2018.

[28] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli, *Image quality assessment: From error visibility to structural similarity*, IEEE TRANSACTIONS ON IMAGE PROCESSING **13** (2004), no. 4, 600–612.

[29] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang, *Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation*, 2020.

[30] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang, *The unreasonable effectiveness of deep features as a perceptual metric*, 2018.

[31] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z. Li, *S^3fd: Single shot scale-invariant face detector*, 2017.