



**CZECH TECHNICAL  
UNIVERSITY  
IN PRAGUE**

**F3**

**Faculty of Electrical Engineering  
Department of Cybernetics**

**Bachelor's Thesis**

# **Facial Time Lapse Video**

**A Generative Approach to Creating Facial Time Lapse  
Videos**

**Ondřej Vereš**

**May 2022**

**Supervisor: Ing. Jan Čech, Ph.D.**





# BACHELOR'S THESIS ASSIGNMENT

## I. Personal and study details

Student's name: **Vereš Ondřej** Personal ID number: **492095**  
Faculty / Institute: **Faculty of Electrical Engineering**  
Department / Institute: **Department of Cybernetics**  
Study program: **Open Informatics**  
Specialisation: **Artificial Intelligence and Computer Science**

## II. Bachelor's thesis details

Bachelor's thesis title in English:

**Facial Time Lapse Video**

Bachelor's thesis title in Czech:

**asobrné video obličeje**

Guidelines:

A time lapse video of a face documents an age progression of a person. The video might be attractive to visualize, e.g., child growing. The input would be an unorganized collection of images (a family photo album, automatically crawled images of politicians or other celebrities). A target person should be present in most of the images, but may appear on an image together with other persons or may not be present at all (on small number of images).

The algorithm would then process the input set: (1) Detects faces, (2) Finds the target person, estimates the acquisition time of the image (in case EXIF data is unavailable or is apparently wrong). (3) Represents face images in StyleGAN [1] latent space, (4) Morphs temporarily subsequent facial images by interpolation in the StyleGAN latent space to generate smooth transitions.

The output will be a video.

Optionally, prepare a web-service, when a user uploads his/her image collection, the algorithm is executed on the server to output the video available for download.

Bibliography / sources:

- [1] T. Karras, S. Laine and T. Aila. A style-based generator architecture for generative adversarial networks. In CVPR, 2019.
- [2] R. Abdal, Y. Qin and P. Wonka. Image2stylegan: How to embed images into the stylegan latent space? In ICCV, 2019.
- [3] A. Subrtova, J. Cech, V. Franc. Hairstyle Transfer between Face Images, In IEEE Face and Gesture Recognition, 2021.

Name and workplace of bachelor's thesis supervisor:

**Ing. Jan Čech, Ph.D. Visual Recognition Group FEE**

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **25.01.2022** Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

Ing. Jan Čech, Ph.D.  
Supervisor's signature

prof. Ing. Tomáš Svoboda, Ph.D.  
Head of department's signature

prof. Mgr. Petr Páta, Ph.D.  
Dean's signature

### III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

\_\_\_\_\_  
Date of assignment receipt

\_\_\_\_\_  
Student's signature

## Acknowledgement / Declaration

I would like to sincerely thank my supervisor Ing. Jan Čech, Ph.D., for giving me the opportunity to work on this very interesting project. Moreover, his guidance and expertise were essential to overcoming problems throughout the development of this project.

Also, I also would like to thank Ing. Daniel Večerka for helping me solve technical issues regarding making the web application available online.

Lastly, I would like to thank my family and friends for their continuous support.

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 20. May 2022

.....

## Abstrakt / Abstract

Tato práce popisuje generativní metodu pro tvorbu časosběrných videí obličejů a webovou aplikaci dostupnou na adrese <http://cmp.felk.cvut.cz/facialtimelapse>, do které uživatel může nahrát sbírku klíčových snímků dané osoby nebo zadat jméno známe osobnosti a sbírka se na základě internetového vyhledávání vytvoří automaticky. Aplikace na základě této sbírky vygeneruje plynulé časosběrné video obličejů dané osoby.

Použití této metody je umožněno díky významnému pokroku v oblasti generování foto-realistických obrázků obličejů pomocí generativní adversální sítě [1] StyleGAN[2].

Naše metoda funguje v následujících krocích: (0) Pokud je vstup jméno známe osobnosti, najde pomocí internetového vyhledávače fotky dané osoby. (1) Seřadí fotky podle automaticky odhadnutého věku. (2) Doporučí odstranění fotek, na kterých se pravděpodobně daná osoba nenachází. (3) Invertuje vstupní obrázky do latentního prostoru StyleGANu. (4) Vygeneruje plynulou sekvenci obrázků dané osoby použitím interpolace v latentním prostoru StyleGANu. (5) Prolne klíčové snímky s vygenerovanou sekvencí pro lepší zachování identity. (6) Spojí výslednou sekvenci obrázků do videa.

Ve srovnání s naivní metodou spočívající v interpolaci intenzit pixelů klíčových snímků, naše metoda produkuje videa, která působí přirozeněji a více realisticky. Pokud ovšem naši metodu porovnáme s naivní metodou v zachování identity, naivní metoda si vede lépe, protože neuronová síť, která ji vyhodnocuje, není citlivá na artefakty „duchů“ z předochozího či následujícího klíčového snímku.

**Klíčová slova:** časosběrné video, generativní adversální sítě, StyleGAN

This thesis describes a generative method for creating facial time lapse videos and a web application available at <http://cmp.felk.cvut.cz/facialtimelapse> where a user can upload his/her image collection of sparse key-frames or enter a well-known person's name, and the application will crawl the collection from the Internet automatically. Afterward, the application will generate a smooth facial time lapse based on the collection.

The proposed method is enabled by the significant progress in the generation of photo-realistic facial images based on an improved generative adversarial network [1], the StyleGAN [2].

The proposed algorithm works in the following steps: (0) If the input is a well-known person's name, crawl images of the person from the Internet. (1) Order input images according to an automatically estimated age. (2) Recommend removing images without the target person. (3) Invert the input images into the StyleGAN latent space. (4) Generate a smooth sequence of images of the target person using linear interpolation in StyleGAN latent space. (5) Partially blend the key-frames into the generated sequence to increase identity fidelity. (6) Combine the resulting sequence of images into a video.

Compared to the naive method, which produces a video by interpolating the pixel intensities of the key-frames, our method produces more natural and realistic facial time lapse videos. However, the naive method performed better in identity loss because the neural network computing the identity loss has minimal sensitivity to “ghosts” artifacts from the previous or next key-frame in the time lapse.

**Keywords:** time lapse, generative adversarial networks, StyleGAN


# Contents /

<b>1 Introduction</b>	<b>1</b>		
1.1 Definition	1		
1.2 Motivation	1		
1.3 Overview of the possible methods	1		
1.3.1 Classical approach	1		
1.3.2 Morphing approach	2		
1.4 Generative approach	2		
<b>2 Technical background</b>	<b>4</b>		
2.1 GANs	4		
2.2 StyleGAN	5		
2.2.1 StyleGAN2	6		
2.2.2 StyleGAN3	6		
2.3 Latent space interpolation	7		
2.4 Representing facial images in the latent space of StyleGAN	7		
2.4.1 Losses	7		
2.4.2 Input space $W^+$	8		
2.4.3 Latent space optimization	8		
2.4.4 Latent space encoding	9		
2.4.5 Combination of encoding and optimization	9		
2.4.6 Latent image manipulation	10		
<b>3 Approach</b>	<b>12</b>		
3.1 Definition of the problem	12		
3.1.1 Input	12		
3.1.2 Output	12		
3.2 General description of the proposed solution	12		
3.3 Crawling the images	12		
3.4 Detecting, aligning faces, and estimating age	13		
3.5 Representing facial images in the latent space of StyleGAN	13		
3.6 Morphing facial images in StyleGAN latent space	13		
3.7 Image blending	14		
3.8 Suggesting removal of images of people who are not the target person	15		
<b>4 Implementation</b>	<b>16</b>		
4.1 Frontend	16		
4.1.1 Components	16		
4.1.2 UI Design	20		
4.2 Backend	20		
4.2.1 API	20		
4.2.2 Crawling the images	21		
4.2.3 Image processing	22		
4.2.4 Inverting and generating images	22		
<b>5 Experimental evaluation</b>	<b>23</b>		
5.1 Methods	23		
5.2 Experiment 1: Generating a facial time lapse	23		
5.2.1 Input images	23		
5.2.2 Visual results	24		
5.2.3 Method evaluation	24		
5.3 Experiment 2: Recreating a facial time lapse	26		
5.3.1 Visual results	26		
5.3.2 Method evaluation	26		
5.4 Discussion	28		
<b>6 Conclusion</b>	<b>29</b>		
6.1 Possible improvements	29		
<b>References</b>	<b>30</b>		

## / Figures

<b>1.1</b>	Time lapse of a flower blooming ..	1
<b>1.2</b>	Example of a classical approach to creating facial time lapse video.....	2
<b>1.3</b>	Example of a morphing approach to creating facial time lapse video.....	2
<b>1.4</b>	Example of a generative approach to creating a facial time lapse video of Bill Gates ...	3
<b>2.1</b>	Visualisation of discriminator and generator .....	4
<b>2.2</b>	Example set of facial images generated by GAN .....	5
<b>2.3</b>	Example set of facial images generated by StyleGAN.....	5
<b>2.4</b>	Example set of facial images generated by StyleGAN2 .....	6
<b>2.5</b>	Example set of facial images generated by StyleGAN3 .....	7
<b>2.6</b>	Encoder Comparison.....	9
<b>2.7</b>	Demonstration of improvement in inversion quality when using a combination of encoding and optimization..	10
<b>2.8</b>	Example of latent space manipulation of age .....	11
<b>3.1</b>	Example of linear interpolation .....	13
<b>3.2</b>	Visualisation of image blending .....	14
<b>4.1</b>	Facial time lapse generator application.....	17
<b>4.2</b>	Enter name component .....	18
<b>4.3</b>	Upload images component .....	18
<b>4.4</b>	Item component .....	19
<b>4.5</b>	Result component .....	19
<b>5.1</b>	Example of an input set .....	23
<b>5.2</b>	Example of a processed list of images .....	24
<b>5.3</b>	Visual comparison of the five mentioned methods .....	24
<b>5.4</b>	Visual comparison of the 5 mentioned methods .....	25
<b>5.5</b>	Visual comparison of the 5 mentioned methods .....	25





<b>5.6</b>	Comparison of the five methods in identity loss .....	26
<b>5.7</b>	Visual comparison of the five mentioned methods .....	27
<b>5.8</b>	Comparison of the five methods in identity loss .....	27
<b>5.9</b>	Comparison of the five methods in identity loss .....	28



# Chapter 1

## Introduction

This chapter will explore the possible methods for creating facial time lapse videos.

### 1.1 Definition

A time lapse video is a series of photographs documenting a process, which generally takes a long time. These images are displayed at a much higher frame rate than initially captured. This visualization technique can take a process that has a duration of several months or years and condense it to several seconds. Typical topics include growing plants or mushrooms, blooming flowers, ants digging holes in ant farms, and other natural processes. An example of a time lapse is shown in Fig. 1.1.



**Figure 1.1.** Time lapse capturing blooming flower.

### 1.2 Motivation

The main motivation behind this work is to create an interesting time lapse video reflecting on how a person looked while growing up. The video should be suitable for sharing with friends or family.

### 1.3 Overview of the possible methods

#### 1.3.1 Classical approach

A person takes a photograph of herself/himself every day. Ideally, these photographs should be taken regularly, with the same lighting at the same angle. Then these photos can be put one after another into a video, as in Fig. 1.2.

The pros:

- Authentic representation of how the person looked.
- If done professionally, this method produces convincing results.

The cons:

- It requires many images.
- Unless done professionally, lighting and head position change in every photo, making the video not smooth.
- Produces visible transition artifacts.



**Figure 1.2.** Example of a classical approach to creating facial time lapse video. A reproduction from [3].

### 1.3.2 Morphing approach

The user selects a few images that she/he wants to include in the time lapse. The user selects facial parts, for instance, nose, eyes, eyebrows, chin, and forehead, on all photographs. The program geometrically warps and morphs the images together to create a smooth transition based on these selections. An example of such a program is WinMorph [4].

The pros:

- Generates a smooth facial time lapse video.
- Only a few key pictures are needed.

The cons:

- It requires a precise manual selection of facial parts on every image.
- Geometric warping and morphing can produce unwanted artifacts.



**Figure 1.3.** Example of a morphing approach to creating facial time lapse video. A reproduction from [5].

### 1.4 Generative approach

We are going to explore this approach in this work. The user selects a handful of images, and a generative network automatically produces the images “in-between”. See Fig. 1.4. as an example.

The pros:

- Generates a smooth facial time lapse video.
- Only a few key pictures are needed.
- The user does not need to select areas in the facial images.

The cons:

- Identity is not guaranteed to be preserved unless we combine this method with others.



**Figure 1.4.** Example of a generative approach to creating a facial time lapse video of Bill Gates.

# Chapter 2

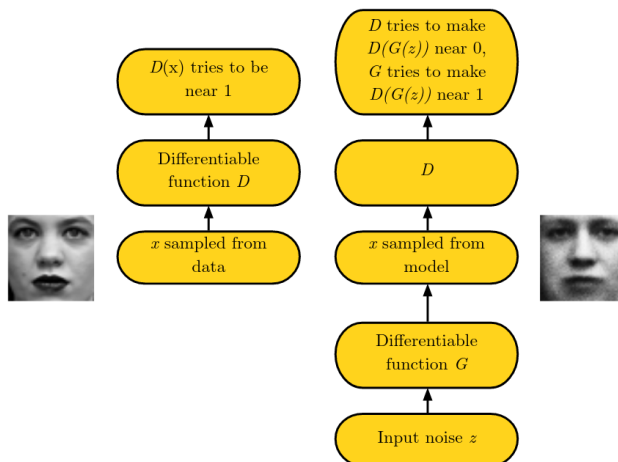
## Technical background

This chapter will explain the basics of GANs, StyleGAN, and GAN inversion.

### 2.1 GANs

Ian Goodfellow and his colleagues invented GANs [1] in 2014 at the University of Montreal.

The generative adversarial networks consist of two neural networks: Generator  $G(\mathbf{z})$  and discriminator  $D(\mathbf{x})$ . The generator produces an image from a random noise sample  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_n, \mathbf{I}_n)$ , where  $n = \dim(\mathbf{z})$ . The discriminator produces a “credibility score” between zero and one for an image. The “credibility score” corresponds to how “sure” the discriminator is that the image is real and was not generated artificially. These two networks are adversarial because they play a mini-max game together. The generator creates new samples, while the discriminator evaluates them and decides whether they are actual samples from the distribution or artificial ones from the generator. The generator aims to generate samples that look like authentic images so that the discriminator cannot tell them apart. However, the goal of the discriminator is to label generated samples as generated while labeling real ones as real with the highest accuracy possible. In other words, the generator tries to maximize “the credibility score” of the generated images while the discriminator tries to minimize the same value and keep “the credibility score” of real images high. GAN is visualized in Fig. 2.1. Early results of facial image synthesis can be seen in Fig. 2.2.



**Figure 2.1.** Visualisation of discriminator and generator. A reproduction from [6].



**Figure 2.2.** Example set of facial images generated by an early GAN. A reproduction from [1].

## 2.2 StyleGAN

StyleGAN [2] was published in 2019 by Nvidia researchers.

The authors of StyleGAN focused on changing the architecture of the generator  $G$  to increase the quality of the images produced. The vector  $\mathbf{z}$  is mapped to  $\mathbf{w} \in W$  by a nonlinear transformation, where  $\dim(Z) = \dim(W) = 512$ . The  $\mathbf{w}$  controls the generator because  $A(\mathbf{w})$  is injected into each of the 18 convolution layers, where  $A$  is learned affine transformation specific for that layer.

The authors trained StyleGAN on the FFHQ dataset [7] consisting of 70,000 images crawled from Flickr [8]. The data set was cropped and aligned using DLIB [9].

With these and several other changes, the authors were able to generate high-quality images with a resolution of  $1024 \times 1024$  pixels and reduce the FID (Fréchet inception distance) [10] score, which is a metric for evaluating the quality of generated images, significantly. More details about StyleGAN can be found in the StyleGAN paper [2]. An example set of images generated by StyleGAN is shown in Fig. 2.3.



**Figure 2.3.** Example set of facial images randomly generated by StyleGAN. The images contain some imperfections and artifacts. Images were downloaded from [11].

### 2.2.1 StyleGAN2

StyleGAN2 was released a year after StyleGAN by an extended team of researchers at Nvidia.

StyleGAN2 improves the generator  $G$  architecture to address issues in StyleGAN, such as various defects and blurred parts of images. Moreover, it brings more changes to increase the quality of the generated images and further reduce the FID score. For more information on these changes, visit the StyleGAN2 paper [12]. The matrix of example facial images generated by StyleGAN2 is shown in Fig. 2.4.



**Figure 2.4.** Example set of facial images generated randomly by StyleGAN2. Many types of artifacts from StyleGAN were removed, and the overall image quality improved. Images were downloaded from [13].

### 2.2.2 StyleGAN3

StyleGAN3 was published in October 2021 by researchers at Nvidia.

The main motivation behind StyleGAN3 is to remove the “Texture sticking problem”. This issue occurs when an image morphs into another in latent space. It appears as if the texture of hair or facial hair was “sticking to the screen”. It is caused because the convolution layers have a reference of pixel coordinates. So, to eliminate the issue, the authors had to get rid of any source positional reference. For instance, they removed per-pixel noise inputs or replaced the learned constant input with a Fourier feature to simplify working in the continuous domain. For more details, please read the StyleGAN3 paper [14]. An example set of images generated by StyleGAN3 is in Fig. 2.5.





**Figure 2.5.** Example set of facial images generated by StyleGAN3. The image quality stayed competitive with StyleGAN2 while removing the “Texture Sticking” issue. Images were downloaded from [15].

## 2.3 Latent space interpolation

Minor changes to the latent code  $\mathbf{w}$  produce minor changes in the image  $G(\mathbf{w})$ . Generating an image on each step from one point to another point in latent space produces a correct facial image. The smaller the steps in the latent space, the smaller the differences between the generated images. In other words, interpolating in latent space produces visually subsequent changes that can be put together into a smooth video. This principle enables our method.

## 2.4 Representing facial images in the latent space of StyleGAN

Image inversion is a problem where we have an image, and we are looking for a latent vector that the generator maps into an image as similar as possible to the input image. There are three most common ways to invert an image into latent space: latent space optimization, latent space encoding, and their combination.

### 2.4.1 Losses

Firstly, we have to be able to tell how good the inversion is compared to the target image. For that, we can use a loss function. This subsection will mention three types of loss metrics: pixel-wise loss, perceptual loss, and identity loss. We can use them independently or use their weighted combination.

- L2 pixel-wise loss is a mean of squared differences between the original image reshaped as vector  $\mathbf{a}$  and the generated image as vector  $\mathbf{b}$ . Formally:

$$L_{Pixel-wise}(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^n (\mathbf{a}_i - \mathbf{b}_i)^2}{n} \quad (1)$$

Where  $n$  is a number of pixels in the image times a number of channels, optimizing with this loss leads to blurry images with almost no detail.

- LPIPS (Learned Perceptual Image Patch Similarity) [16] loss is a perceptual similarity metric. It correlates with the human perception of similarity between images. As a result, the inversion does not lose detail, stays sharp, and matches the original image more closely.

Interestingly, this metric is calculated using neural networks  $\mathcal{F}$  trained for ImageNet [17] classification, most commonly VGG [18] or AlexNet [19]. ImageNet classification is a task where an algorithm is asked to assign a correct class to the image  $\mathbf{x}$ . The number of classes is one thousand, and they include, for instance, “coffee mug” or “limousine”.

LPIPS loss is computed by extracting the feature stack from  $L$  layers of the network  $\mathcal{F}$  and unit normalizing them in the channel dimension. These feature extractions are defined as  $F_l(x), F_l(x_0) \in \mathcal{R}^{H_l \times W_l \times C_l}$  for layer  $l$ , where  $H_l, W_l, C_l$  mean height, width and number of channels corresponding to layer  $l$ . These extractions are scaled by a trained weight vector  $w_l \in \mathcal{R}^{C_l}$  channelwise. Subsequently, the  $l_2$  distance is computed for these extractions. Lastly, the  $l_2$  distances are averaged for each layer  $l$ , and these averages are summed to produce the LPIPS loss.

$$L_{LPIPS}(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|w_l \odot (F_l(x)_{hw} - F_l(x_0)_{hw})\|_2^2 \quad (2)$$

- Identity loss is the cosine similarity of the identity descriptors generated by VGG [18] or ArcFace [20].

$$L_{Id}(\mathbf{d}_1, \mathbf{d}_2) = \frac{\mathbf{d}_1 \cdot \mathbf{d}_2}{\|\mathbf{d}_1\| \|\mathbf{d}_2\|} \quad (3)$$

## ■ 2.4.2 Input space $W^+$

The authors of the paper “How to Embed Images Into the StyleGAN Latent Space?” [21] found that embedding into the extended latent space  $W^+$  produces much better results than embedding into the input latent space  $Z$  or intermediate latent space  $W$ .

Any vector  $\mathbf{w}$  from  $W^+$  is a concatenation of eighteen vectors from  $W$ . Therefore,  $\dim(W^+) = 18 \times \dim(W) = 18 \times 512$ . Number 18 corresponds to a number of convolution layers in the StyleGAN architecture. Then each layer works with a different part of the latent vector  $\mathbf{w}$ , making it easier to find a better inversion.

## ■ 2.4.3 Latent space optimization

It is possible to use gradient descent to find an inversion with minimal loss. The process starts with zero or random latent vector  $\mathbf{w}_0$  and target image  $\mathbf{o}$ . The loss value  $l$  is calculated with loss function

$$l = L(\mathbf{o}, G(\mathbf{w})).$$

The  $\mathbf{w}_{i+1}$  is updated as:

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \mu \nabla L(\mathbf{o}, G(\mathbf{w}_i))$$

where  $\mu > 0$  is the learning rate,  $\nabla L$  is the gradient of the loss function. PyTorch [22] calculates this gradient automatically, using its autograd feature.

The drawbacks of this approach are that we might find a local minimum instead of a global one and that it is a generally slow process, which can take up to several minutes. Therefore, we will use latent space encoding as an alternative in this work.

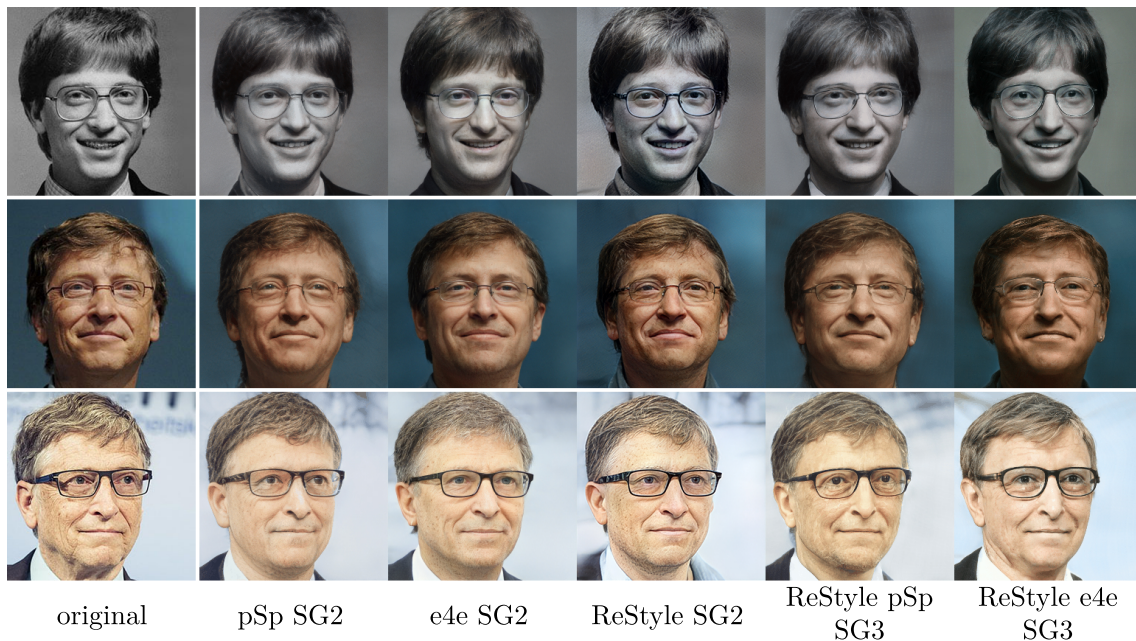
### 2.4.4 Latent space encoding

Latent space encoding is an alternative to latent space optimization. Instead of iteratively finding the minimum, a neural network is trained to find an inversion in a single forward pass with the smallest loss possible. This method takes only a fraction of a second per image. Therefore, it is more suitable for this work.

These are current state-of-the-art StyleGAN encoders:

- **Pixel2style2pixel StyleGAN2 encoder** [23] for short “pSp”. It is an encoder which transforms an image into style vectors, and (after possible modifications) the generator can create an image based on the style vectors.
- **Encoder4editing StyleGAN2 encoder** [24], for short “e4e”, is designed for image manipulation in latent space.
- **ReStyle StyleGAN2 encoder** [25] is a residual-based encoder. Instead of directly predicting the inversion, it predicts the residual between the estimated inversion and the original image. Based on the residual, it corrects the estimated inversion. This iterative process can run any number of times, but the best results are achieved when the number of iterations is relatively small, for instance, two or three.
- **ReStyle pSp StyleGAN3 encoder** [26] is an updated version of the ReStyle encoder based on pSp created for StyleGAN3.
- **ReStyle e4e StyleGAN3 encoder** [26] is similar to the previous encoder, but it is based on the e4e encoder.

A visual comparison of the encoders mentioned above is in Fig. 2.6.



**Figure 2.6.** Encoder Comparison. Restyle SG2 performed best in this comparison. However, the quality of inversions varies a lot based on the input image.

### 2.4.5 Combination of encoding and optimization

It is possible to combine the previous two methods to achieve better results. This process first encodes the image and then uses the inversion as the starting point for the optimization. As a result, it speeds up the optimization. The results produced by this method are shown in Fig. 2.7.



**Figure 2.7.** Demonstration of improvement in inversion quality when using a combination of encoding and optimization.

#### 2.4.6 Latent image manipulation

This subsection describes latent image manipulation. This technique was not used in this work. However, it could be used to improve it further, as will be discussed in section 6.1.

After inverting the image  $\mathbf{x}$ , we have a latent  $\mathbf{w}$ . We can make semantically meaningful changes to  $\mathbf{w}$  by adding or subtracting a latent directional vector  $\mathbf{d}$  scaled by scalar  $p$ .

The direction vector  $\mathbf{d}$  corresponds to a semantic direction of the generated image, such as age, gender, facial expression, or facial pose.

As stated in thesis “Face Image Editing in Latent Space of Generative Adversarial Networks” [27], for non-binary features such as age, the directional vector  $\mathbf{d}$  can be found using linear regression as

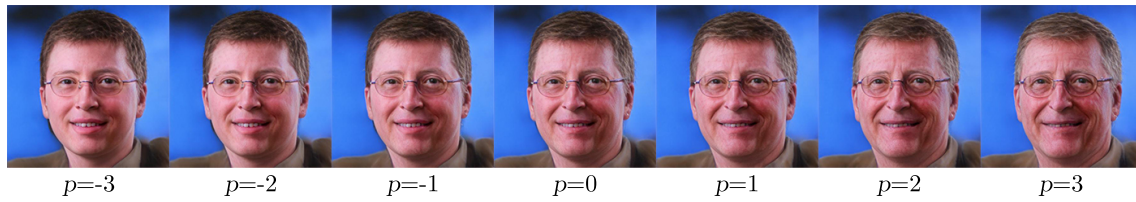
$$\min_{\mathbf{d}} \|\mathbf{W}\mathbf{d} - \mathbf{S}\|_2^2$$

where  $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_n]^T$ ,  $\mathbf{S} = [s_1, \dots, s_n]^T$  and  $s_i$  is a semantic score (e.g. normalized age) corresponding to the target feature of the image  $G(\mathbf{w}_i)$

For binary features, such as “gender”, or “having glasses”, the directional vector  $\mathbf{d}$  can be found using a linear classifier, such as SVM [28]. The linear classifier finds a hyperplane that separates latent vectors into two groups, depending on whether  $G(\mathbf{w}_i)$  has the binary feature. A normal vector for this hyperplane is the directional vector  $\mathbf{d}$ .

By adding or subtracting the directional vector  $\mathbf{d}$  to  $\mathbf{w}$ , we could see changes corresponding to the feature of the directional vector  $\mathbf{d}$  in the output image  $G(\mathbf{w} + p \cdot \mathbf{d})$ . This way, we could change the target person’s gender or make the person younger or older.

An example of latent space manipulation of age can be shown in Fig 2.8. More details about latent image manipulation can be found in the following sources [27, 29].



**Figure 2.8.** Example of latent space manipulation of the age of Bill Gates.

# Chapter 3

## Approach

### 3.1 Definition of the problem

#### 3.1.1 Input

Input is a collection of images of a particular person. This collection should include images from the person's entire lifespan, for example, Facebook profile photos, family albums, or a custom-made collection created for this project. Alternatively, the input can be just a name of a well-known person. In this case, the system automatically obtains the images by querying the Programmable search engine [30] made by Google.

#### 3.1.2 Output

The output is a facial time lapse video demonstrating the age progression of the target person.

### 3.2 General description of the proposed solution

This section presents an algorithm for generating facial time lapse videos. The algorithm operates as described in the following steps:

1. Optional only if the input is a name - crawl images from the Internet
2. Detects all faces on the input images.
3. Align, crop, and resize all detected faces.
4. Estimate the age for every facial image.
5. Order facial images by age and present them to the user.
6. Recommend removal of images that probably do not feature the target person.
7. Let user remove, add, or reorder facial images.
8. Encode facial images into StyleGAN latent space.
9. Linearly interpolate latent vectors and generate a sequence of latent vectors.
10. Generate an image from every latent vector in the sequence.
11. Blend the generated images with authentic images to increase identity fidelity.
12. Create a video from the generated images and show it to the user.

### 3.3 Crawling the images

The algorithm uses a search engine to download images of the target person. It tries to download photos from her/his entire lifespan. This goal is achieved by searching repeatedly and altering the search phrase to get age-diversified results. For instance, the algorithm uses phrases such as "photo of target person as a young adult".

### 3.4 Detecting, aligning faces, and estimating age

The algorithm uses third-party libraries and algorithms for these tasks. They will be described in more detail in the implementation Chapter 4.

### 3.5 Representing facial images in the latent space of StyleGAN

The algorithm should be interactive. Consequently, we cannot do latent space optimization because the user would have to wait several minutes for each input image to invert. Therefore, we decided to use latent space encoding.

To get the latent space representation (latent vector  $\mathbf{w}$ ) in  $W^+ \subseteq \mathcal{R}^{18 \times 512}$  the algorithm passes the input image  $\mathbf{x} \in \mathcal{R}^{256 \times 256 \times 3}$  to the encoder  $E: \mathcal{R}^{256 \times 256 \times 3} \rightarrow W^+$

$$\mathbf{w} = E(\mathbf{x}). \tag{1}$$

The algorithm supports the encoders for StyleGAN2 and StyleGAN3 to demonstrate the changes between them.

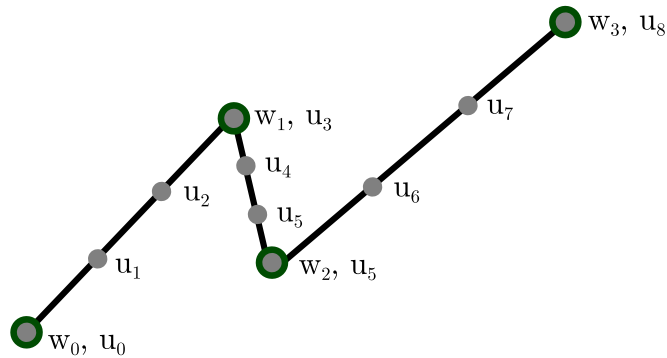
### 3.6 Morphing facial images in StyleGAN latent space

After inverting the input images  $\mathcal{J} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ , we have a sequence of latent vectors  $\mathcal{S} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_n)$  where  $n$  is the number of inverted images. The density constant  $d$  sets a number of vectors, between  $\mathbf{w}_i$  and  $\mathbf{w}_{i+1}$  including the point  $\mathbf{w}_i$  and excluding the point  $\mathbf{w}_{i+1}$ , to compute. We are looking for a sequence  $\mathcal{F} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_m)$  where  $m = (n - 1)d + 1$

The algorithm finds  $\mathcal{F}$  with a linear interpolation as

$$\begin{aligned} \alpha &= \frac{i}{d}, \\ j &= \lfloor \frac{i}{d} \rfloor, \\ \lambda &= \alpha - j. \end{aligned} \quad \mathbf{u}_i = (1 - \lambda)\mathbf{w}_j + \lambda\mathbf{w}_{j+1}, \tag{2}$$

A visual example of the linear interpolation equation (2) is shown in Fig. 3.1.



**Figure 3.1.** Example of linear interpolation with  $n = 3$  and  $d = 3$  in 2D space.

Algorithm passes the vectors from  $\mathcal{F}$  to StyleGAN generator  $G: W^+ \rightarrow \mathcal{R}^{1024 \times 1024 \times 3}$  and provides a sequence  $\mathcal{G} = (\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_m)$  of generated images.

$$\mathbf{g}_i = G(\mathbf{u}_i) \tag{3}$$

### 3.7 Image blending

The inversion does not usually preserve the identity of the target person perfectly. Some details such as wrinkles, scars, or freckles are often lost. This issue is fundamental because the user expects the algorithm to preserve the identity. A facial time lapse video of a person without a real identity is not interesting to watch.

The algorithm mixes inverted and real images to preserve the identity. For that, the algorithm uses the function

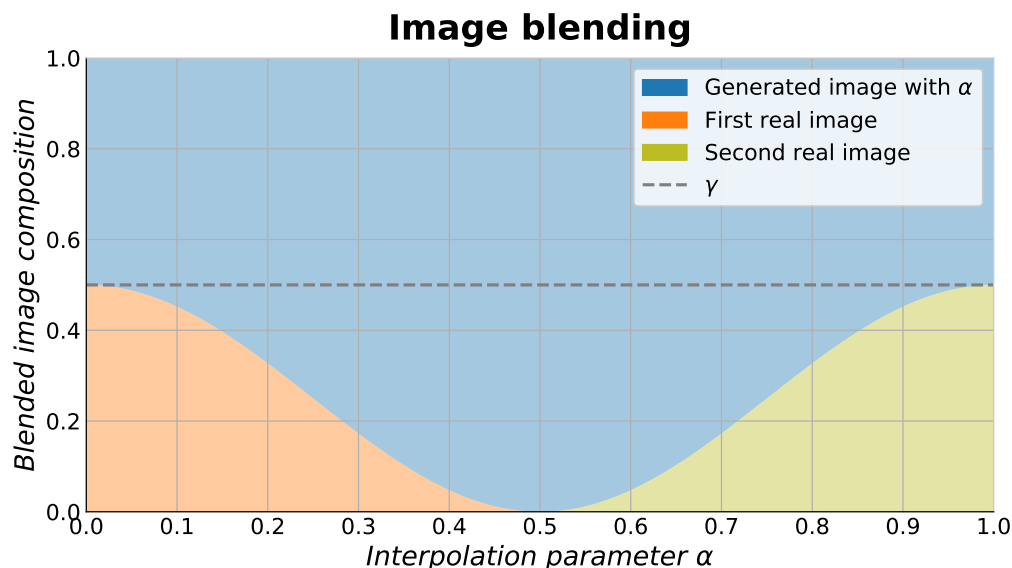
$$f(\alpha, \gamma) = \frac{\cos(2\pi\alpha)\gamma + \gamma}{2} \quad (4)$$

with parameter  $\alpha$  from Eq. (2) corresponding to the current position in the video and a new parameter  $\gamma$  which defines the maximal portion of the real image in the output image.

The output image is a blend of real image (crawled from the Internet or uploaded by user) and generated image. To calculate the pixel values of the *BlendedImage* the algorithm uses the following formula with function  $f$  from Eq. (4):

$$\text{BlendedImage} = f(\alpha, \gamma) \cdot \text{RealImage} + (1 - f(\alpha, \gamma)) \cdot \text{GeneratedImage}. \quad (5)$$

Visualization of the image blending functionality is in Fig. 3.2.



**Figure 3.2.** Visualisation of image blending based on the formula (5).

The image blending function  $f$  from Eq. (4) uses cosine (instead of, for instance, a piecewise linear function) to make the blending as smooth as possible. The algorithm lets the user adjust  $\gamma$ . From testing, the value  $\gamma = 0.5$  works generally well. If  $\gamma$  is too high, time lapse appears as if it stopped moving and only showed a static picture. If  $\gamma$  is too low, identity is not preserved.

The image blending function is “U-shaped” because the generated and authentic images are similar only when interpolation parameter  $\alpha$  from Eq. (2) is close to an integer, i.e., the current position in the video matches a period of time with a real input image. Otherwise, they would not match, and blending would result in strange artifacts.



### 3.8 Suggesting removal of images of people who are not the target person

The algorithm does not remove any images, but it suggests to the user that an image might show somebody else than the target person. This way, we avoid the accidental removal of images with the target person.

Due to the technical limitations of the web application, this feature does not take into account already uploaded images. It runs when the user asks the server to crawl images or when she/he uploads at least four images at once.

For this task, we decided to use a simple baseline method described in the paper “Learning CNNs from weakly annotated facial images” [31] in section 5.2. This method assumes most images feature the target person. Thanks to the median’s robustness, the element-wise median of identity descriptors extracted from the input key-frames corresponds to the representation of the target person well.

The algorithm takes a batch of images  $\mathcal{B} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ . It uses ArcFace [20] to get an identity descriptor  $\mathbf{id}_i \in \mathcal{R}^{512}$  for image  $\mathbf{x}_i$ . Afterwards, an element-wise median  $\mathbf{m}_{id} \in \mathcal{R}^{512}$  is calculated from the embeddings. Lastly, if

$$\|\mathbf{id}_i - \mathbf{m}_{id}\|_2^2 > \epsilon \quad (6)$$

the image is labeled as “recommended to be removed”. Value of  $\epsilon = 0.002$  has been set empirically based on testing.

# Chapter 4

## Implementation

This chapter will focus on implementing the web application for generating facial time lapse videos. The application is currently hosted at <http://cmp.felk.cvut.cz/facialtimelapse> (note: https is not supported - the application will load, but it will not be able to communicate with the backend server). If the application is down, please contact me at [veresond@fel.cvut.cz](mailto:veresond@fel.cvut.cz).

We decided to create a web application because it is a user-friendly way of showing our method. In addition, it does not require installation and runs on all platforms that can run an Internet browser.

The frontend runs in the user's browser, and the backend runs on a GPU server. We had to split the application into backend and frontend because the browser cannot run our program, mainly because it requires a GPU with CUDA installed. These two programs communicate using HTTP requests in the following way: Frontend (browser) sends an HTTP request (for example, to crawl images or to generate a GIF) to the backend, and the backend sends a response to these requests containing necessary data (aligned images with estimated ages, or the output GIF.)

The frontend is written in JavaScript, HTML, and CSS. The browser downloads these files when it visits the website. Then, the browser renders the website and displays it to the user.

Backend is written in Python because it offers many machine learning libraries. In addition, the key component for our project StyleGAN [2, 12, 14] is written in Python as well.

Fig. 4.1. shows the design of our application.

### 4.1 Frontend

The client-side of the application is built mainly in JavaScript [32] and uses react.js [33]. The react library is helpful because of two points:

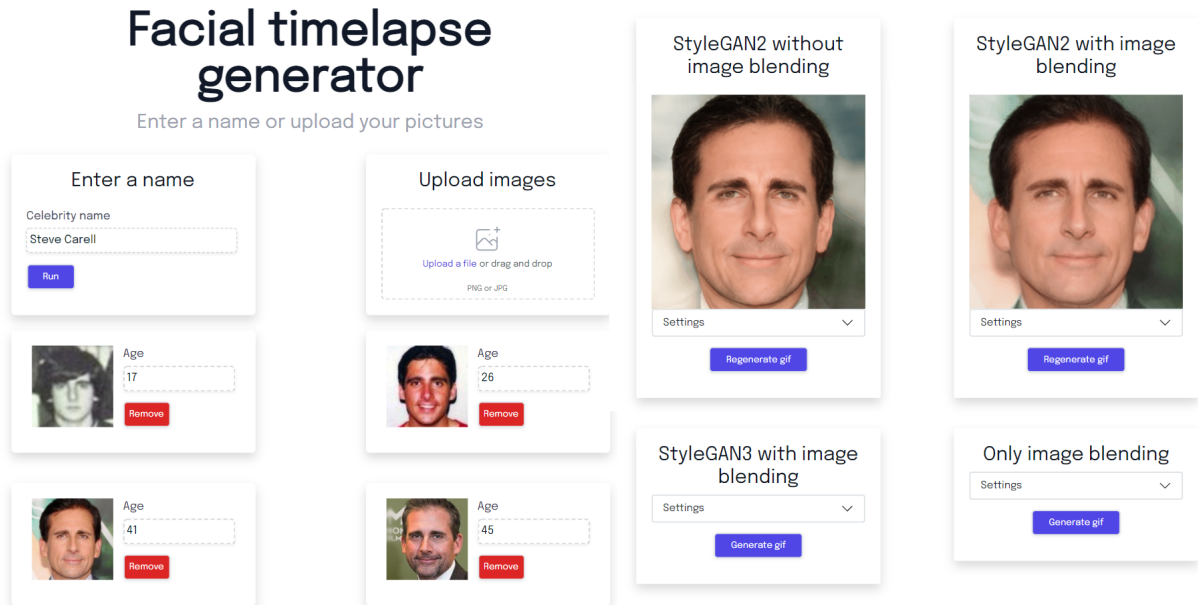
- We can create reusable components, such as containers for image and age, containers for displaying the facial time lapse, or containers for uploading the images. These components can then be used anywhere throughout the app, eliminating the need for code duplicity.
- React.js keeps track of the state of every individual component. Whenever the state of a component changes, the component is re-rendered with the values of the new state. Therefore, the displayed values are always in sync with the internal values.

#### 4.1.1 Components

The application uses the following react.js components:

- **appComponent** This component contains all other components and the web page as a whole. It keeps track of all the current images and their age in its state.

This component contains the following functions:



**Figure 4.1.** Facial time lapse generator application.

- **Removing items** This function takes `itemId` as input and removes the item with that `itemId` from the items array in the state.
  - **Changing age in an item** This function takes `inputId` and `newAge` and updates the corresponding item in the items array in the state
  - **Synchronizing the facial time lapse GIFs** When one GIF is loaded, the other ones reset to synchronize.
- 
- **enterNameComponent** This component handles entering the name of a well-known person. Sending the name to the backend. Waiting for the backend to reply with photos and ages and then putting it to the parent app component state.

This component has three different states:

- **default** Component is waiting for the input from the user. In this state, it displays an input field and a run button.
- **waiting** Component sent data to the backend and is waiting for the response. In this state, it displays loading animation.
- **failed** Component either received no response or a response with error. This state displays the status code of the error and the error message.

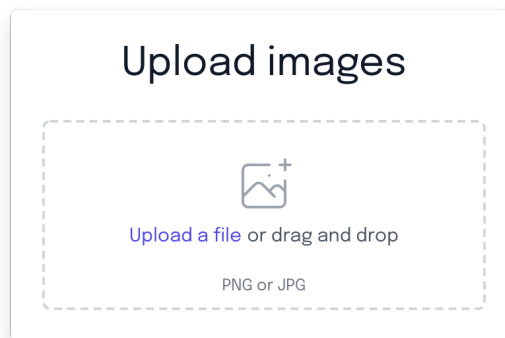
Screenshots of `enterNameComponent` are in Fig. 4.2.



**Figure 4.2.** Enter Name Component.

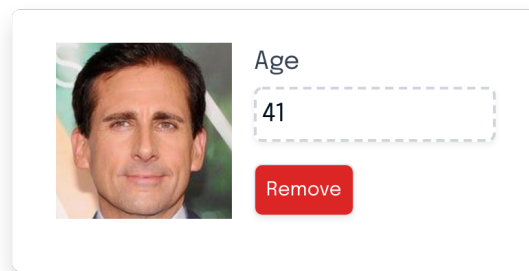
- **fileUploadComponent** This component handles manual image upload. Users can upload multiple images or a single image by dragging and dropping the images or clicking the “Upload a file” text and selecting the images from their file explorer. Images must be in JPEG or PNG formats.

Similarly, as the `enterNameComponent`, it has three states that behave similarly. This component is shown in Fig. 4.3.



**Figure 4.3.** Upload Images Component.

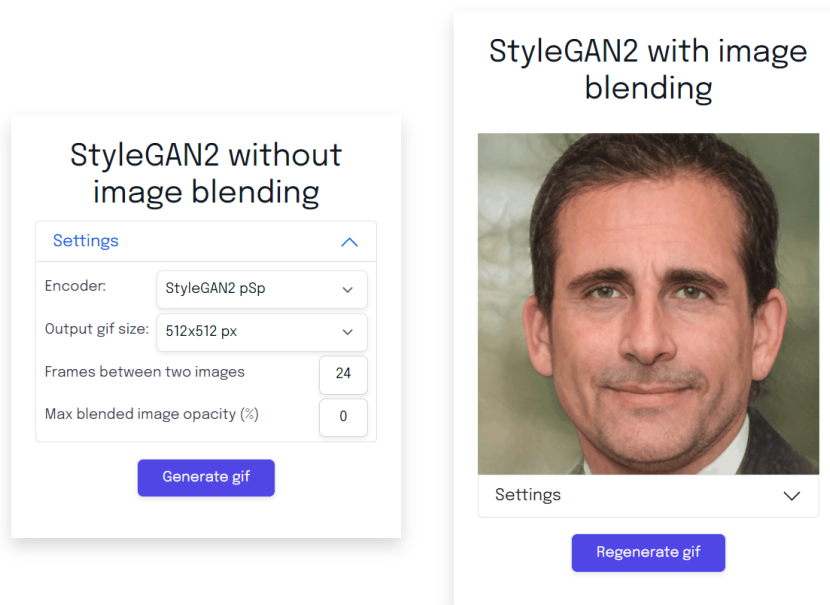
- **itemComponent** This component displays, as shown in Fig. 4.4., the aligned image along with the predicted age and button to remove the image. Also, it allows the user to edit the automatically predicted age. If the corresponding image has the “recommended to remove” label, the background is changed to light red, and the remove button also includes the text “recommended”.
- **resultComponent** This component handles entering the settings. It lets the user to select:
  - **Encoder:**
    - StyleGAN2 pSp
    - StyleGAN3 ReStyle pSp



**Figure 4.4.** Item Component.

- None - When a user enters this value, max blended opacity is ignored, and images are blended pixel-wise.
- **Output GIF size:**
  - 1024x1024 px
  - 512x512 px
  - 256x256 px
- **Frames between images** — this parameter corresponds to parameter  $d$  in Eq. (2). It sets the number of generated images between two original images.
- **Max blended image opacity** — this parameter corresponds to parameter  $\gamma$  in Eq. (4), but it is in percentage %.

Afterward, it sends images along with corresponding ages to the backend. When it receives the response containing the video in GIF format. It displays it to the user. The result component in its two states can be seen in Fig. 4.5.



**Figure 4.5.** Result Component.

### 4.1.2 UI Design

The application uses Tailwind CSS [34] which is a CSS framework. Tailwind CSS makes designing UI faster and easier by generating CSS code for predefined HTML classes.

Tailwind UI [35] is a UI component library offering both free components and paid components. The design of the components of this web application is based on Tailwind UI free tier components.

It uses Epilogue [36] as a font.

## 4.2 Backend

Backend is responsible for crawling the images, aligning them, inverting them into StyleGAN latent space, and generating facial time lapse video. All of the services are accessible to the frontend with the API. The backend uses PyTorch [22] for image inference, and Flask [37] for communicating over the web using the API.

### 4.2.1 API

The application provides the following API endpoints:

- **/uploadFile** Accepts a POST HTTP request containing a list of images. Send a response containing JSON with imageUrls and ages.
- **/uploadName** Accepts a POST HTTP request with celebrityName. Send a response containing JSON with imageUrls and ages of crawled results. Example request:

```
POST http://halmos.felk.cvut.cz:5000/uploadNameAPI
Content-Type: application/json;

{
  "celebrityName": "Steve Carell"
}
```

Example response with an array of items, each item contains the predicted age along with a path to the aligned and cropped image and a “recommended to remove” label:

```
[
  [
    "ee014acd-1358-4189-b4b7-08a20ae14216.jpg", //path to image
    12, //estimated age
    false // 'recommended to remove' label
  ],
  [
    "ae38552f-2503-4284-a09b-b72c6f61e826.jpg",
    17,
    true
  ], ...
]
```

- **/generateGif** Accepts a POST HTTP request containing image URLs, ages and settings. Example request:

```

POST http://halmos.felk.cvut.cz:5000/generateGifAPI
Content-Type: application/json;
{
  "encoder": "psp",
  "frames": "24",
  "opacity": "50",
  "size": "512",
  "items": [
    {
      "path": "d193da85-fbce-4c72-a074-27ef54e3c64a.jpg",
      "age": 25
    },
    {
      "path": "d73542fe-ddb7-4011-addd-3ccaba045812.jpg",
      "age": 26
    },...
  ]
}

```

The response contains a facial time lapse video in a GIF format.

■ **/storage/[filename]** Example request:

```
GET http://halmos.felk.cvut.cz:5000/storage/image.jpg
```

The response contains an image stored at that URL.

## 4.2.2 Crawling the images

The application uses Custom Search JSON API alongside a Programmable search engine [30] from Google to get images of well-known people.

Programmable search engine is a search engine from Google that can be set to search only a specific website. However, it is set to search the entire web in our case. Unfortunately, we cannot use Google.com because it does not have any public API. A Programmable search engine set to search the entire web is inferior to Google.com because it lacks several search features, and other features do not work well. Mainly, the duplicate filter does not seem to be working.

Custom Search JSON API is also a service from Google. The application can make a request with a search query, Programmable search engine id, and other search parameters and receive JSON with results as a response.

All search parameters are stored in the request URL. For example, one request may look like this:

```

GET https://customsearch.googleapis.com/customsearch/v1?
cx=b0aeb7e24cc9a435d&           //Programmable search engine id
q={search_term}&                 //Search query
imgType=face&                   //Image type [clipart, face, lineart,
                                //stock, photo, animated]
imgSize=medium&                 //Image size
num=4&                           //Number of results to return [1-10]
safe=active&                    //Enables SafeSearch filtering
searchType=image&               //Specifies image search
filter=1&                        //Enables duplicate filter

```

```
key=AIzaSy...& //Google cloud key, used for billing
start=1 //The index of a first result to return
```

To get images throughout the person’s life. The application makes four requests with the following queries:

- photo of [name] as a kid
- photo of [name] as a teenager
- photo of young [name]
- photo of [name]

Each request returns links to two top results. Afterward, the application downloads the images. To speed up the process. Search requests and download requests are sent in parallel using a modified Request boost Python package [38].

### 4.2.3 Image processing

DLIB [9] detects all faces on the images, then aligns, crops and scales to resolution  $1024 \times 1024$  pixels each detected face. Afterward, the application removes duplicate images by calculating L2 pixel-wise loss (1) between all images and removing those with a pixel-wise loss smaller than an empirically set threshold. Pixel losses are calculated only from a small portion of the image cropped from the center to speed up the process.

If the application is processing more than four images uploaded by the user or the input was crawled from the Internet, it selects images that are probably not the target person by following a procedure described in section 3.8.

Lastly, ages are estimated for each facial image using a neural network “DEX” proposed in the paper “Deep EXpectation of Apparent Age from a Single Image”[39]. DEX uses the VGG-16 [18] architecture and was trained on a crawled data set of 0.5 million images with available age from IMDB and Wikipedia.

### 4.2.4 Inverting and generating images

The backend server runs with the encoder models, age estimation model, and ArcFace identity model loaded to speed up this process. These models consume about 4.5 GB of GPU memory. The backend server is currently running on server Halmos at CMP. It uses a single NVIDIA GTX TITAN X card with 12GB of memory. A single inversion using SG2 pSp takes about 0.25 seconds, and using SG3 ReStyle pSp (with three iterations) takes about a second. Generating a single frame takes about 0.25 seconds with SG2 and 0.30 seconds with SG3. Because the output video usually contains about a hundred frames, the generation task takes the most time. The batch size for inverting and generating the images is kept at one to be able to generate multiple facial time lapse videos at once. When the user simultaneously generates multiple facial time lapse videos using different encoders, the GPU memory consumption goes up to 9GB.



# Chapter 5

## Experimental evaluation

### 5.1 Methods

- **Pixel interpolation** This is a naive baseline method. It linearly blends the images, i.e., interpolates pixel intensities, to create a time lapse. It does not use StyleGAN or any other generative network.
- **StyleGAN2 without image blending** This method uses pSp to invert the image into  $W^+$  latent space. Then it interpolates the images in the  $W^+$  latent space.
- **StyleGAN2 with image blending** Similar to the previous one, it blends authentic images into the video according to section 3.7 with  $\gamma = 0.5$ .
- **StyleGAN3 without image blending** This method uses ReStyle pSp to encode images into  $W^+$  latent space. The number of iterations for iterative refinement is set to the default value of 3.
- **StyleGAN3 with image blending** Same as the above method, but it also uses image blending from section 3.7 with  $\gamma = 0.5$ .

### 5.2 Experiment 1: Generating a facial time lapse

#### 5.2.1 Input images

Input is an unorganized set of images. Images in this experiment were not automatically crawled but manually collected to get the best result possible. The input set is shown in Fig. 5.1.



**Figure 5.1.** Example of an input set.

The algorithm detected aligned and cropped faces, as shown in Fig. 5.2. Then, the age was estimated for each image, and the images were ordered.



**Figure 5.2.** Example of a processed list of images.

### 5.2.2 Visual results

The algorithm generated facial time lapse video using the five methods mentioned in section 5.1. The results are visible in Fig. 5.3., 5.4. and 5.5. The Pixel Interpolation method produces “ghosts” of the previous/next person in the video. Generative methods without blending tend to miss some important details for identity preservation. In contrast, generative methods with blending bring those details back when  $\alpha$  is close to an integer.



**Figure 5.3.** Visual comparison of the five mentioned methods with  $\alpha \in [0, 1.5]$ .

### 5.2.3 Method evaluation

This section will evaluate all methods based on identity loss (3). Pixel-wise (1) and perceptual loss (2) are misleading because they are very sensitive to changes in the background. The identity loss graph in Fig. 5.6. was plotted from the input set in Fig. 5.2. Density from Eq. (2) was set to  $d = 24$ . The loss function is calculated against the closest neighbor among the authentic images. SG2 pSp and SG3 Restyle encoders perform in terms of identity loss similarly. One is sometimes better than the other one and vice versa. Image blending decreases identity loss significantly. Pixel interpolation has the smallest identity loss due to minimal sensitivity to “ghost” artifacts. They will be mentioned in more detail in section 5.4.



**Figure 5.4.** Visual comparison of the five mentioned methods with  $\alpha \in [1.75, 3.25]$ .



**Figure 5.5.** Visual comparison of the five mentioned methods.  $\alpha \in [3.5, 4.75]$ .

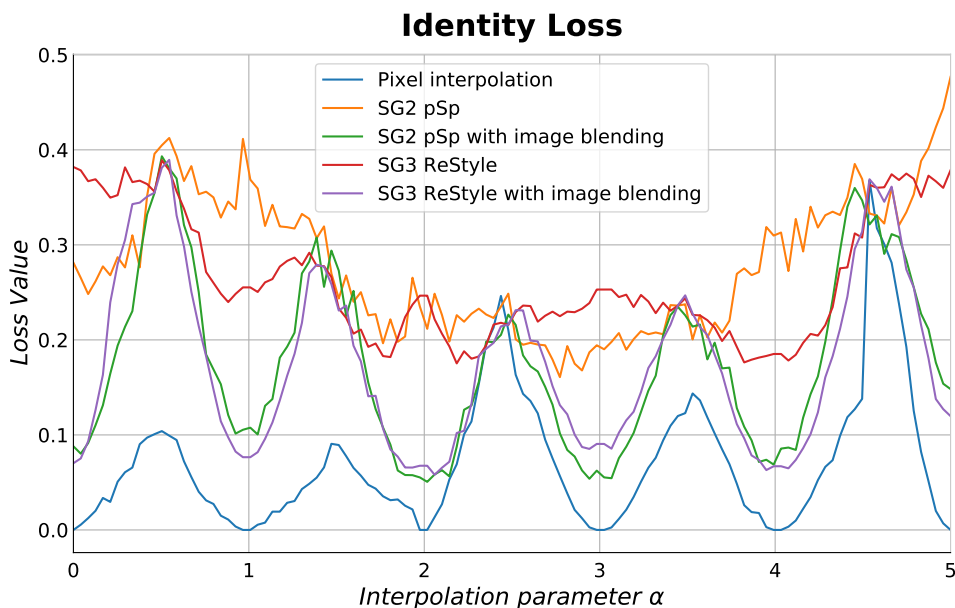


Figure 5.6. Comparison of the five methods in identity loss.

### 5.3 Experiment 2: Recreating a facial time lapse

This experiment recreates a professional imitation of a facial time lapse video “Danielle” [40]. Anthony Cerniello and his team created this imitation of facial time lapse in 2013 using numerous images and videos of family members. Then, the animators combined the images and videos into a single video using Adobe After Effects [41], Autodesk 3D Studio Max [42], and Nuke [43].

We can try to replicate only the first part of this imitation because further in the imitations, a part of the head is out of the frame.

This experiment takes the first and last image of the “usable” part of the imitation and compares the result with the imitation itself.

#### 5.3.1 Visual results

Facial time lapse videos generated in Experiment 2 are shown in Fig. 5.7.

#### 5.3.2 Method evaluation

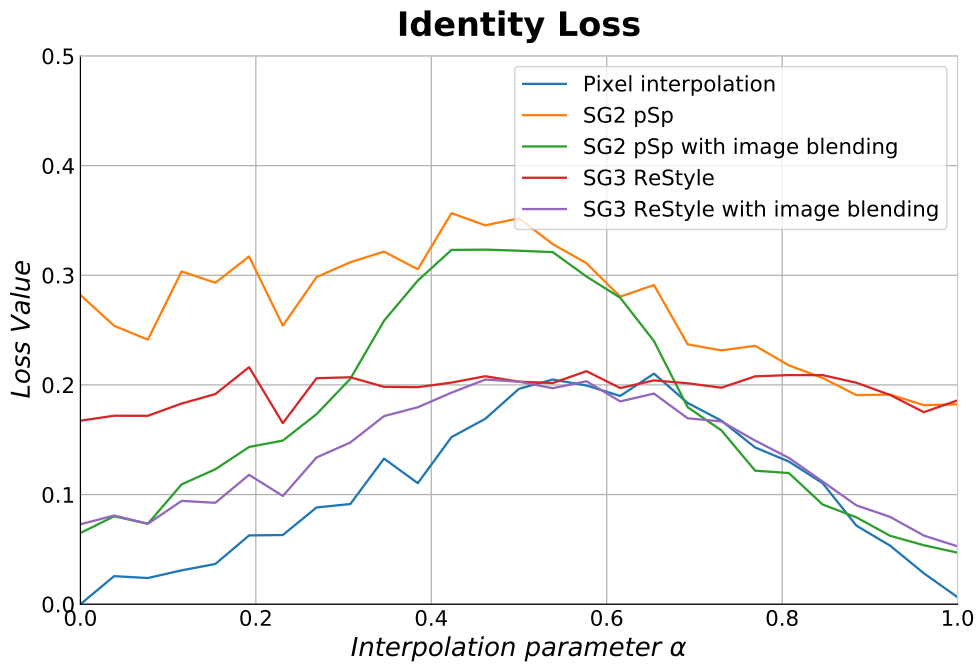
The identity loss is calculated against the corresponding image in the original time lapse imitation in Fig. 5.8.

SG3 Restyle encoder compared to SG2 pSp created an inversion with smaller identity loss for the first image. However, they performed similarly in encoding the last image of the time lapse. Image blending decreases identity loss when  $\alpha$  is close to an integer, as expected. Pixel interpolation has the lowest overall identity loss due to the minimal sensitivity to “ghost” artifacts of the identity loss.

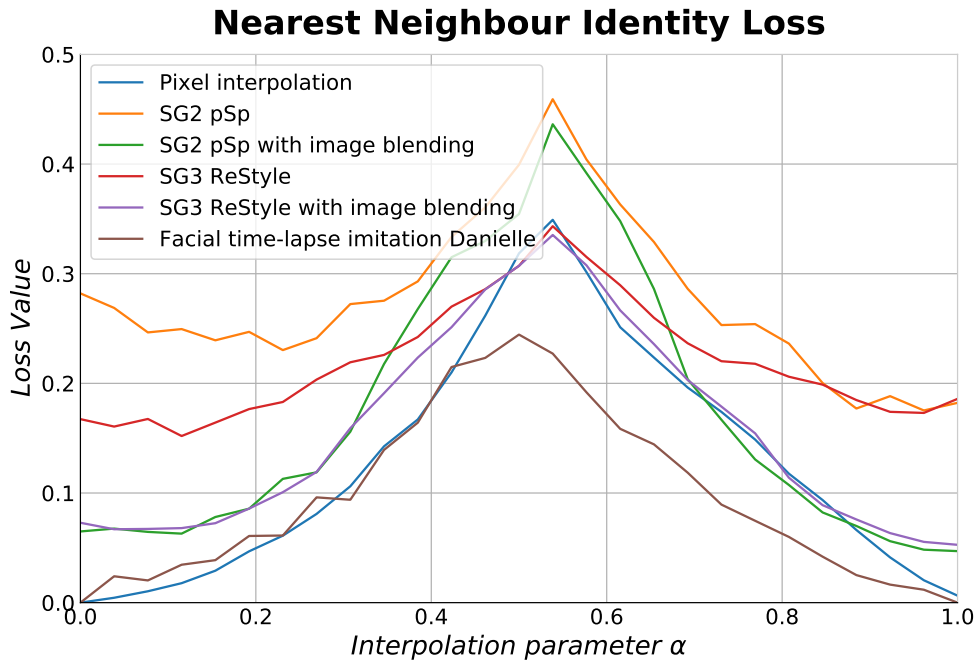
In contrast, in Fig. 5.9., the identity loss is calculated against the nearest neighbor among the first and last frames of the time lapse imitation. The results approximately correspond to the results in Fig. 5.8. Interestingly, we can see that the time lapse imitation has a significant identity loss, which theoretically should be 0. We believe it is because the identity descriptors generated by ArcFace [20] are not perfectly robust to aging. However, another reason could be that ArcFace is picking up that



**Figure 5.7.** Visual comparison of the five mentioned methods.  $\alpha \in [0, 1]$



**Figure 5.8.** Comparison of the five methods in identity loss calculated against the corresponding image in the time lapse imitation “Danielle”.



**Figure 5.9.** Comparison of the five methods in identity loss calculated against the nearest neighbor among first and last frame in the time lapse imitation “Danielle”.

“Danielle” is not a real time lapse video but a time lapse imitation created from images of multiple family members.

## 5.4 Discussion

We can see that the graphs look “upside-down V”-shaped because we have the most information about how the person looked when  $\alpha$  is an integer. As a consequence, the losses are smaller. On the contrary, we have the least information when  $\alpha$  corresponds to a position in the middle between two images. Therefore, the loss is higher.

The main takeaway from identity loss is that image blending improves identity preservation. Furthermore, as expected, the identity is not guaranteed to be preserved without image blending. However, image blending only increases identity preservation when  $\alpha$  is close to a whole number.

We were unable to demonstrate that our approach is better than straightforward pixel interpolation using identity loss because this loss is not sensitive to “ghosts” of the previous or next person in the time lapse, even though the “ghost” artifacts are particularly disturbing for humans. The artifacts are especially apparent when pose or hairstyle differ between images.

# Chapter 6

## Conclusion

To conclude, we introduced a generative method to create facial time lapse videos from a small number of key images. The method uses StyleGAN [2, 12, 14] to generate pictures in between the keyframes and combines them into a smooth video. To preserve the identity of the target person, the method uses blending with original images at keyframes. Humans (generally) agree that our method is superior to pixel interpolation. In addition, a web application was made available at <http://cmp.felk.cvut.cz/facialtimelapse> for anyone to try the method using their images and parameters. Additionally, the application can search for images on the Internet of a well-known person, use them as input, sort input images agewise, and recommend removing people who are probably not the target person.

### 6.1 Possible improvements

Given more time, we could use latent image manipulation to further improve our method in the two following ways:

- We could generate a facial time lapse from a single picture by slowly altering age using latent space manipulation. We would start with a lower age than the original image and smoothly increase the age until a certain threshold.
- We could extend the time lapse by manipulating age in the first and last images of the time lapse to cover the target person's entire lifespan.

## References

- [1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014.  
<https://arxiv.org/abs/1406.2661>.
- [2] Tero Karras, Samuli Laine, and Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2018.  
<https://arxiv.org/abs/1812.04948>.
- [3] Andrew Wyton. *Face Time-lapse (3 years)*.  
<https://www.youtube.com/watch?v=1TnTZOr5RMw>.
- [4] Satish Kumar. *WinMorph*.  
<https://www.debugmode.com/winmorph/>.
- [5] Mark Stead. *Child growth face morph time-lapse (from birth to almost 4)*.  
<https://www.youtube.com/watch?v=ZTjHLF3xKWo>.
- [6] Ian J. Goodfellow. *Generative Adversarial Networks - NIPS tutorial*. 2016.  
<https://www.iangoodfellow.com/slides/>.
- [7] Janne Hellsten Tero Karras. *Flickr-Faces-HQ Dataset (FFHQ)*.  
<https://github.com/NVLabs/ffhq-dataset>.
- [8] Stewart Butterfield, and Caterina Fake. *Flickr*.  
<https://www.flickr.com/>.
- [9] Davis E. King, and others. *DLIB C++ Library*.  
<http://dlib.net/>.
- [10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. 2017, DOI 10.48550/ARXIV.1706.08500.
- [11] Timo Aila Tero Karras, Samuli Laine. *Example images produced using StyleGAN*.  
<https://drive.google.com/drive/folders/100DJ0QXyG89HZzB4w2Cbyf4xjNK54cQ1>.
- [12] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. *Analyzing and Improving the Image Quality of StyleGAN*. 2019.  
<https://arxiv.org/abs/1912.04958>.
- [13] Phil Wang. *StyleGAN2 random people generator*.  
<https://thispersondoesnotexist.com>.
- [14] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. *Alias-Free Generative Adversarial Networks*. 2021.  
<https://arxiv.org/abs/2106.12423>.
- [15] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. *Example images produced using StyleGAN3*.



- <https://nvlabs-fi-cdn.nvidia.com/stylegan3/images/stylegan3-r-fhq-1024x1024/>.
- [16] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*. 2018. <https://arxiv.org/abs/1801.03924>.
  - [17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. *ImageNet: A large-scale hierarchical image database*. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009. 248-255.
  - [18] Karen Simonyan, and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. <https://arxiv.org/abs/1409.1556>.
  - [19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. In: F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, eds. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012. <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.
  - [20] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. *ArcFace: Additive Angular Margin Loss for Deep Face Recognition*. 2018. <https://arxiv.org/abs/1801.07698>.
  - [21] Rameen Abdal, Yipeng Qin, and Peter Wonka. *Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?* 2019. <https://arxiv.org/abs/1904.03189>.
  - [22] Facebook’s AI Research lab. *PyTorch*. <https://pytorch.org/>.
  - [23] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. *Encoding in Style: a StyleGAN Encoder for Image-to-Image Translation*. 2020. <https://arxiv.org/abs/2008.00951>.
  - [24] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. *Designing an Encoder for StyleGAN Image Manipulation*. 2021. <https://arxiv.org/abs/2102.02766>.
  - [25] Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. *ReStyle: A Residual-Based StyleGAN Encoder via Iterative Refinement*. 2021. <https://arxiv.org/abs/2104.02699>.
  - [26] Yuval Alaluf, Or Patashnik, Zongze Wu, Asif Zamir, Eli Shechtman, Dani Lischinski, and Daniel Cohen-Or. *Third Time’s the Charm? Image and Video Editing with StyleGAN3*. 2022. <https://arxiv.org/abs/2201.13433>.
  - [27] Čech Jan Nela Petrželková. Face Image Editing in Latent Space of Generative Adversarial Networks.
  - [28] Corinna Cortes, and Vladimir Vapnik. Support-vector networks. *Machine learning*. 1995, 20 (3), 273–297.
  - [29] Peiye Zhuang, Oluwasanmi Koyejo, and Alexander G. Schwing. *Enjoy Your Editing: Controllable GANs for Image Editing via Latent Space Navigation*. 2021. <https://arxiv.org/abs/2102.01187>.

- [30] Google LLC. *Programmable Search Engine*.  
<https://programmablesearchengine.google.com/about/>.
- [31] Vojtěch Franc, and Jan Čech. Learning CNNs from weakly annotated facial images. *Image and Vision Computing*. 2018, 77 10-20. DOI <https://doi.org/10.1016/j.imavis.2018.06.011>.
- [32] Brendan Eich of Netscape, and others. *JavaScript*.  
<http://www.ecma-international.org/publications-and-standards/standards/ecma-262/>.
- [33] Facebook Inc. *React.js, a javascript library for building user interfaces*.  
<https://reactjs.org/>.
- [34] Tailwind Labs. *Tailwind CSS, a utility-first CSS framework*.  
<https://tailwindcss.com/>.
- [35] Tailwind Labs. *Beautiful UI components, crafted with Tailwind CSS*.  
<https://tailwindui.com/>.
- [36] ETC Tyler Finck. *Epilogue, a sans serif variable font*.  
<https://etceteratype.co/epilogue>.
- [37] Armin Ronacher. *Flask*.  
<https://flask.palletsprojects.com/>.
- [38] Kuldeep Singh Sidhu. *A simple package for hitting multiple URLs and performing GET/POST requests in parallel*.  
<https://github.com/singhsidhukuldeep/request-boost>.
- [39] Rasmus Rothe, Radu Timofte, and Luc Van Gool. *DEX: Deep EXpectation of Apparent Age from a Single Image*. In: *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*. 2015. 252-257.
- [40] Edmund Earle Anthony Cerniello, Nathan Meier, and George Cuddy. *Danielle*.  
<https://www.youtube.com/watch?v=JRqPJdgdIM>.
- [41] Adobe Inc. *Adobe After Effects, digital visual effects, motion graphics, and compositing application*.  
<https://www.adobe.com/cz/products/aftereffects>.
- [42] Inc. Autodesk. *3ds Max, a professional 3D computer graphics program*.  
<https://www.autodesk.com/products/3ds-max/overview>.
- [43] Foundry. *Nuke, a node-based digital compositing and visual effects application*.  
<https://www.foundry.com/products/nuke/>.