Face Editing in Texture Space

Bastien Husler

Bachelor Thesis
August 2023

Dr. Endri Dibra
Dr. Beren Kaul
Prof. Dr. Markus Gross

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ARBREA
LABS

cgl
computer graphics laboratory

Abstract

This thesis aims at developing a novel texture editing pipeline for human faces. The goal is to detect and remove facial wrinkles and other potential skin anomalies in a fully automated way. The new skin texture obtained by the pipeline will serve as a source texture for simulating aesthetic medicine and plastic surgery procedures. Therefore, supersampling is used at the end to enhance quality and address the high expectations imposed by medical uses.

The hereby proposed solution is structured into three major steps:

1. Detection: Wrinkles and skin imperfections are being labelled at this stage. The perfect detection outputs a binary mask covering accurately all unwanted facial areas while covering as little as possible wanted areas. The mask shall provide a reliable input for the next step.
2. Inpainting: Provided the input image and the generated detection mask, this step has to recreate skin texture to fill the labelled areas. The challenge is to create detailed skin texture and avoid filling the areas with new computer-generated wrinkles. Skin tone and lighting should also match the overall scene of the input picture, in order to provide a seamless inpainting.
3. Upscaling: Finally, the processed image is upscaled to gain details and provide a workable texture for future applications. The upscaler should not smooth nor modify the colours of the image.

This work provides a comparison of multiple implementations of these steps. Each containing naive and straight-forward solutions as well as more involved and state-of-the-art algorithms.

Finally, a complete and automated pipeline is proposed to evaluate the performance of the solutions on different hardwares.

Zusammenfassung

Diese Arbeit handelt sich um die Entwicklung einer neuartigen Pipeline zur Bearbeitung von Textur für menschliche Gesichter. Ziel ist es, Gesichtsfalten und andere potenzielle Hautanomalien vollautomatisch zu erkennen und zu entfernen. Die durch die Pipeline gewonnene neue Hauttextur wird als Quellentextur für die Simulation von Verfahren der ästhetischen Medizin und plastischen Chirurgie dienen. Daher wird am Ende ein Supersampling durchgeführt, um die Qualität zu verbessern und den hohen Erwartungen der medizinischen Anwendungen gerecht zu werden.

Die hier vorgeschlagene Lösung gliedert sich in drei große Schritte:

1. Erkennung: Falten und Hautunreinheiten werden in dieser Phase gekennzeichnet. Die perfekte Erkennung gibt eine binäre Maske aus, die alle unerwünschten Gesichtsbereiche genau abdeckt, während die gewünschten Bereiche so wenig wie möglich verdeckt werden. Die Maske soll eine zuverlässige Grundlage für den nächsten Schritt liefern.

2. Einfärben: Ausgehend vom Eingabebild und der generierten Erkennungsmaske muss in diesem Schritt die Hauttextur neu erstellt werden, um die markierten Bereiche zu füllen. Die Herausforderung besteht darin, eine detaillierte Hauttextur zu erzeugen und zu vermeiden, dass die Bereiche mit neuen, computergenerierten Falten gefüllt werden. Hautton und Beleuchtung sollten ebenfalls mit der Gesamtszene des Eingangsbildes übereinstimmen, um ein unbemerkbares Einfärben zu ermöglichen.

3. Hochskalierung: Schließlich wird das verarbeitete Bild hochskaliert, um Details zu gewinnen und eine brauchbare Textur für zukünftige Anwendungen zu erhalten. Der Upscaler sollte die Farben des Bildes weder glätten noch verändern.

Diese Arbeit bietet einen Vergleich mehrerer Implementierungen dieser Schritte. Sie enthalten jeweils naive und einfache Lösungen sowie komplexere und modernere Algorithmen.
Schließlich wird eine vollständige und automatisierte Pipeline vorgeschlagen, um die Leistung der Lösungen auf verschiedenen Hardwaresystemen zu bewerten.

Acronyms

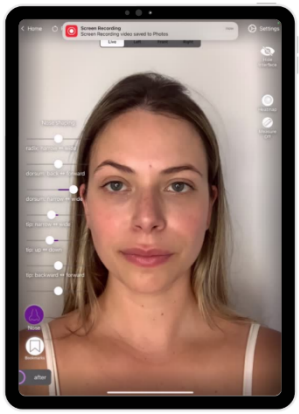
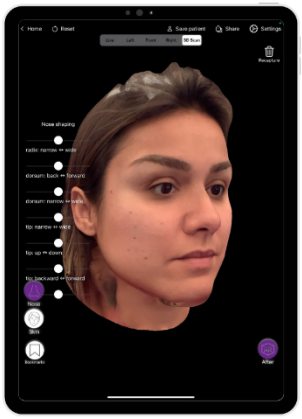
GMM	Gaussian Mixture Model
MRF	Markov Random Fields
GMM-MRF	Gaussian Mixture Model and Markov Random Fields
EM	Expectation-maximisation
JSI	Jaccard Similarity Index
HHF	Hybrid Hessian Filter
MAE	Masked AutoEncoder
FFHQ	Flickr-Faces-HQ Dataset [18]
GAN	Generative Adversarial Network

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

computer graphics laboratory

Bachelor Thesis

Face Editing in Texture Space

Introduction

Arbrea Labs builds **AR&3D** surgery simulators that have revolutionized patient-surgeon visual communication during consultations and are a game-changer in aesthetic medicine and plastic surgery. Integral technologies to such tools vary from 3D Reconstruction and AR/VR to Neural Rendering and Physic Simulations. In this thesis we look into the texture component of a facial capture.

Task Description

The bachelor thesis consists of the following steps:

- Get acquainted with the relevant literature
- Detect the wrinkles in the image of a face
- Detect other potential skin anomalies/lesions/imperfections
- Implement face inpainting to replace the image in the areas where wrinkles and other face features are detected
- Implement super resolution to improve the resolution of the inpainted area
- (Bonus) Implement the same for the eye region for double eyelid surgery
- (Bonus) Implement controllable face inpainting where local or global attributes (e.g. age or beard presence) can be chosen

Remarks

A written report and an oral presentation conclude the thesis. The thesis will be overseen by Prof. Markus Gross and supervised by Dr Endri Dibra and Dr Beren Kaul.

Contact

For further information, please contact (endri.dibra@arbrea-labs.com) or
the CGL thesis coordinator (cgl-thesis@inf.ethz.ch)

Arbrea Labs

Acknowledgments

I would like to express my sincere gratitude to my advisors, Dr. Endri Dibra and Dr. Beren Kaul, for their inestimable guidance and support throughout my thesis research. I am very grateful to them for providing me with the opportunity to conduct my research at Arbrea Labs. The whole Arbrea team as well as my fellow Bachelor colleagues provided helpful insights and strong support, which greatly helped me to complete this research and write this thesis.

I would also like to express many thanks to Prof. Dr. Markus Gross for overseeing the work I provided for this project. A special thanks also goes to the entire CGL team, for hosting the thesis at the Computer Graphics Lab.

I would like to acknowledge the lecturers at ETH Zürich, especially the Visual Computing Lecturers, Prof. Dr. Mark Gross and Prof. Dr. Marc Pollefeys. The captivating topics taught in these lectures gave me the motivation to conduct research in this field.

I am also thankful to my friends and family for their support and patience. Their enthusiasm for my research project forced me to surpass myself and achieve results I could be proud of, myself. Motivation and inspiration is often to be found during a pleasant chat or a great laugh.

Contents

List of Figures

List of Tables

1

Introduction

The rise of cellphone cameras and social media has popularised facial filters as a fun way to engage with others. Interestingly, this trend has also found its place in the aesthetic medicine and plastic surgery industry. These innovative facial filters have the ability to alter the shape and texture of human faces, which is crucial for creating accurate digital simulations of plastic surgery outcomes. Procedures like facelifts and eyelid lifts not only change the shape of a patient's face, but also address issues with skin texture. Through techniques like filler injections and skin stretching, cosmetic professionals aim to eliminate signs of aging, injuries, wrinkles, and other skin irregularities. Therefore, it is essential for professional aesthetic applications to consider these modifications in order to deliver realistic and precise previews.

This work will explore and evaluate methods in the image space to perform skin texture editing. The removal of facial wrinkles, signs of injuries and other skin irregularities and the overall enhancement of the skin texture resolution are going to be investigated. The newly generated skin once converted to a texture map should then be able to be applied to the geometrically modified face to generate a realistic simulation of an aesthetic procedure. Therefore, the quality of the removal and the resulting resolution are of crucial importance. The computational efficiency of the proposed methods will also be important, as these may eventually be used on a portable device such as a smartphone or a tablet.

The first challenge of this research is to identify the current literary context, in order to pin down the existing answers and voids. To do so, an extensive study of the current literature was necessary. The findings will be presented in the following section, dedicated to related works. These can be classified into two different main parts. Filter and rule-based approaches were first detailed in the literature. These kinds of method attempt to solve the goals by applying predefined filters to the image or by changing pixel values based on some algorithm or mathematical formula. Applying these general methods usually requires some tweaking by the user in order to find suitable hyperparameters for a given input image. Today's computing power enabled new emerging methods for image processing. Most recent research in the area build their

solutions based on deep learning models that are built and trained especially for the given task. These require strong computing power in order to be trained on lots of images and data. If correctly trained, they offer a robust and sometimes efficient solution not requiring input specific parameters from the user.

The first goal of the work is to detect the wrinkles and other potential skin anomalies. These unwanted regions should be labelled as such. The perfect detection should successfully identify all of these regions and cover them entirely such that it would provide a suitable inpainting mask. This step is more than just correctly labelling, it also addresses the removal of these detected areas. Removing wrinkles in a targeted manner requires accurate detection, but also high-quality inpainting. In order to achieve maximum quality, the detection should retain as much original skin as possible, while it removes the wrinkled areas just enough for the inpainting step to avoid recreating the removed wrinkle.

The second goal of this assignment is to use the detection work to implement face inpainting. Inpainting consists of bridging the gaps created by a mask on the image by deducing new pixel values based on the remaining context of the image. An inpainting algorithm attempts to recover the lost information. This method will allow the replacement of the image, in the areas where wrinkles and other face features had been detected. This inpainting step should avoid recreating the unwanted wrinkles and skin lesions that have been labelled out, but it should also provide a fresh skin texture of high quality. Ideally, the inpainted skin should completely merge with the subject's skin and reflect the same attributes as the surrounding skin.

The third goal of this project is the use of super resolution to improve the overall texture quality of the output image. As the obtained texture is going to be used as a texture map for a 3D reconstruction of the face, increasing its resolution will avoid loosing too much quality by stretching it to fit the 3D model. The upscaler should implement super resolution and keep the details of the input image in order to achieve comparable or even better performances regarding the overall realism. The upscaled image should thus avoid creating artefacts, changing the perceived texture and colours.

The main section of the work will be composed of three subsection each for the previously presented goals. Each subsection will present a variety of methods and assess their performance. All presented methods will finally be compared to each other regarding the quality of their output and also their computational efficiency at the end of each subsection.

2

Related Work

2.1. Filter/Rule-based Approaches

The 2014 published paper *Detection and inpainting of facial wrinkles using texture orientation fields and markov random field modeling* [1] presents a full detection and inpainting pipeline for facial wrinkles. The algorithm uses Gabor filters to extract the image features. Afterwards, the amplitude and orientation of these features are used to perform image segmentation using a GMM-MRF approach similar to the one described in this 2013 published paper [24]. As this method not only relies on the Gabor amplitude response, but also on the orientation of the extracted features, it takes advantage of the longitudinal shape of wrinkles. The paper uses the Gaussian Mixture Model (GMM) to model each pixel's class depending on the amplitude of the Gabor filters' responses. A Markov Random Fields (MRF) framework is also used to determine an appropriate prior distribution for each pixel depending on its neighbours. Doing so improves spatial smoothness and segmentation accuracy. Given these priors and the texture orientation field provided by the Gabor filters, the final segmentation mask is obtained by applying the Expectation-maximisation (EM) algorithm provided. Results can be seen on Figure 2.2.b.

The same paper published in 2014 [1] uses texture quilting [6] to inpaint the labelled regions with wrinkle-free skin texture. Image Quilting [6] is a famous inpainting technique that consists of using samples extracted from a source texture image and pasting them on the gaps to be inpainted. On the contrary to pixel-based inpainting techniques, this method offers to determine the value of a small patch of pixel at the same time. Pasting random blocks to fill the gaps leaves visible boundaries in the texture (see Figure 2.1.a). Therefore quilting [6] also consists of choosing the most appropriate block in the input texture to get an overlap as seamless as possible (see Figure 2.1.b). However this still leaves visible artefacts, thus stitching is implemented as well to create a minimum error boundary cut for each new block. This will avoid the apparition of straight lines (see Figure 2.1.c).

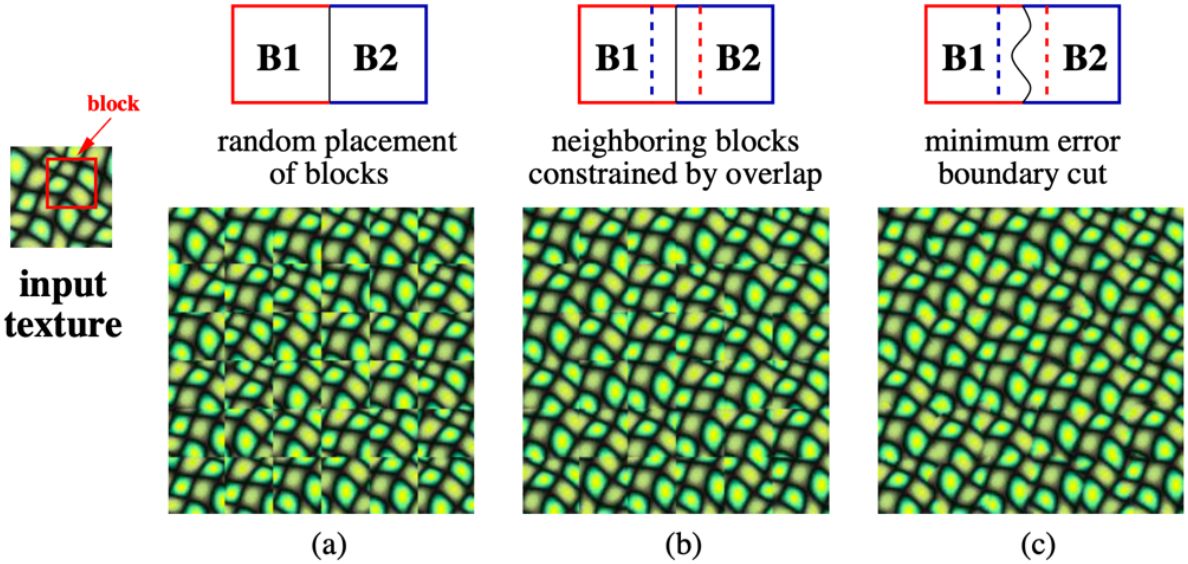


Figure 2.1.: *Quilting texture (figure originates from [6])*

The unlabelled regions of the face are used as the source texture for quilting. It is divided into samples. The gaps created by the detection part of the pipeline will be filled using texture quilting [6], thus selecting the most appropriate samples extracted (see Figure 2.2.c).

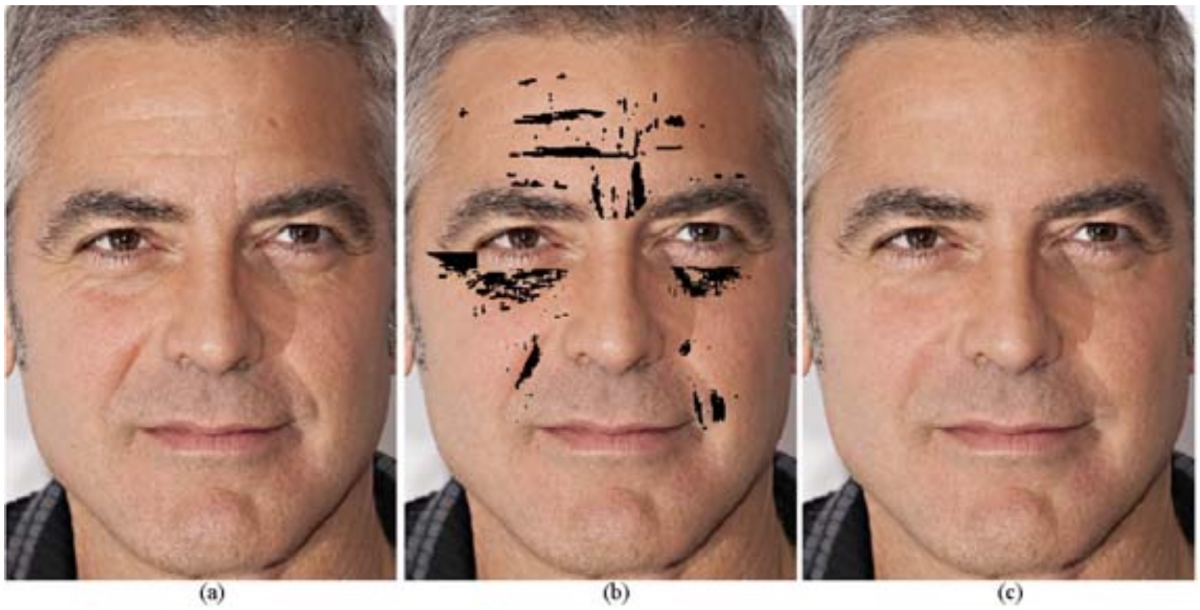


Figure 2.2.: *Presented GMM-MRF results (figure originates from [1])*

The 2020 published paper *Evaluation of automatic facial wrinkle detection algorithms* [7] presents a comparison of facial wrinkle detection algorithms developed in the past. Researchers manually labelled wrinkles on various face images to get a ground truth dataset. Afterwards, the results of the tested detection algorithms were compared to these ground truth masks with the Jaccard Similarity Index (JSI). The Hybrid Hessian Filter (HHF) [23] and the presented GMM-MRF [1] methods are evaluated in this work. Along with the evaluation, the paper pro-

poses to use Jerman et al.'s 2D enhancement method [15] to detect wrinkles. The enhancement method [15] has been initially developed for medical imaging purposes in order to enhance the visualisation of vascular structures. The proposed enhancement method computes the eigenvalues of the Hessian matrix of each pixel in the image. For 2D images, a vascular structure is characterised by a big gap between the magnitudes of the two eigenvalues $|\lambda_2| \gg |\lambda_1|$. The evaluation of these methods shows that the enhancement method [15] yields overall better and more robust results than the two other algorithms (see Figure 2.3). The proposed method also detects much smaller details on the skin (see Figure 2.4).

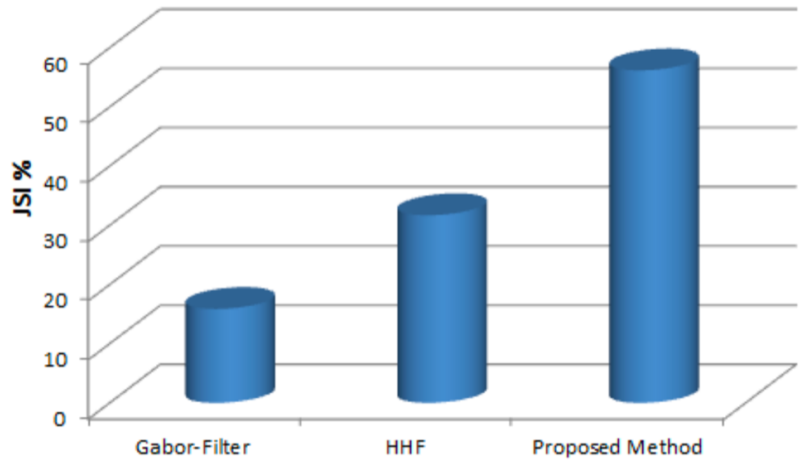


Figure 2.3.: *Accuracy of automatic wrinkles detection (figure originates from [7])*

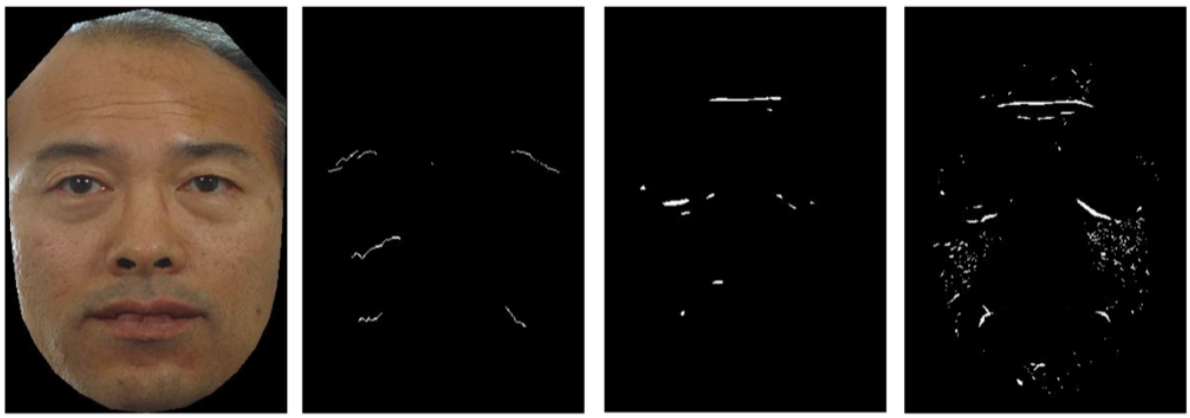


Figure 2.4.: *Comparison of automatic wrinkle detection techniques (figure originates from [7]). (a) Original image; (b) GMM-MRF; (c) HHF; (d) Enhancement method.*

A variety of image interpolation methods for upscaling already exist and the most popular ones have been evaluated in a 2013 published paper [10]. The nearest neighbour, the bilinear, the bicubic and the cubic B-Spline interpolations all consist of bridging the pixel gaps generated by the upscaling with functions over the already existing pixels. Choosing the most appropriate method among them usually depends on the computational efficiency to image quality trade-off that they offer.

2.2. Deep Learning Approaches

Deep learning is gaining importance in all fields of Computer Science, especially for Computer Vision and Computer Graphics. The most up-to-date publications in the field are using neural network architectures to tackle the addressed issues. The 2022 published paper entitled *Photorealistic facial wrinkles removal* [33] proposes a pipeline consisting of the same steps of detection and inpainting, this time by using deep learning methods. The wrinkle detection is performed by a fine-tuned version of UNet++ [42], a medical image segmentation architecture. A custom dataset based on the Flickr-Faces-HQ Dataset [18] (FFHQ) is used to train a pretrained implementation of the UNet++ [42] model. This new method is compared with the GMM-MRF [1] method and results show, that the proposed method offers a more precise and less invasive labelling of facial wrinkles.

Sanchez et al. [33] present an inpainting module using the 2021 proposed LaMa [35] inpainting architecture. LaMa [35] is a novel deep learning inpainting model that uses Fast Fourier Convolution [5] layers to provide a global context of the image and to make use of periodic structures found in the input image. Although LaMa [35] has been trained in low resolution (256×256), it scales to high-resolution images without loss of quality (see Figure 2.5). The DeepFillv2 [40] architecture presented in 2019 is used among other deep learning inpainting models to compare the performance of the LaMa [35] architecture. DeepFillv2 [40] is a generative image inpainting model specialised in free-form inpainting masks. As labelled wrinkles are very rarely polygons, this method is well suited for the shapes of the masks obtained by the wrinkle detection modules.

Figure 2.5.: *Inpainted images using LaMa (figure originates from [35])*

Facial inpainting is still an active research area and some recent findings in the area are algorithms built in a similar way to the LaMa [35] architecture in order to tackle the drawbacks of the original version. Enhancement of the LaMa [35] architecture by using a Masked AutoEncoder (MAE) as an attention prior has been presented in 2022 [4]. It claims to provide more long-distance information to the inpainting model and tests on the FFHQ [18] dataset showed a significant improvement for facial inpainting. The 2022 published paper entitled *DIFAI: Diverse Facial Inpainting using StyleGAN Inversion* [39] presents an innovative facial inpainting method, that uses the embedding space of the famous StyleGAN [19] model. In comparison to all previously mentioned algorithms, DIFAI [39] also provides a variety of possible inpaintings, in which the user can choose to select the most appropriate one.

Generative Adversarial Networks (GAN) are also widely used to address the upscaling problem. The 2018 presented ESRGAN [38] is already an improvement of existing GAN upscalers. This enhanced version of a super-resolution GAN is implemented using a new Residual-in-Residual Dense Block, which has higher capacity and is easier to train than the original blocks used. Results showed, that ESRGAN [38] outperformed the other existing super-resolution GANs. This model also serves as a benchmark for other experimentations and super-resolution models. BSRGAN [41] is an improvement of ESRGAN [38] proposed in 2021. By further degrading training input images, the model could achieve better results. Super-resolution GANs also gained in importance with the recent rise of text-to-image synthesis, like with the famous DALL-E 2 [31] model. GigaGAN [16] was presented in early 2023 as a novel GAN architecture for text-to-image synthesis. The model contains a GAN-based upsampler that asserts to surpass the newest improved version of ESRGAN [38] called RealESRGAN [37] (see Figure 2.7).

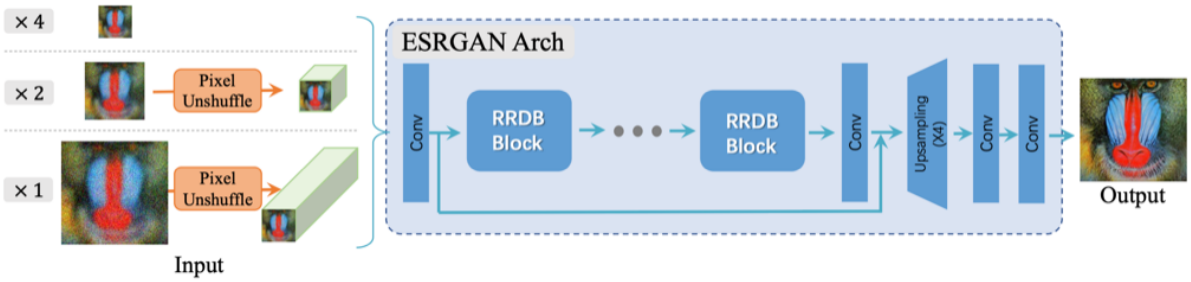


Figure 2.6.: *ESRGAN generator architecture (figure originates from [37])*

Figure 2.7.: *Comparison of different upscaling methods (figure originates from [37])*

3

Building and Assessing an Automatic Wrinkle Removal Pipeline

3.1. Detecting Wrinkles and Skin Lesions

The detection step consists of creating a suitable mask for the image to be used in the next step which is inpainting. In this section, different approaches to the wrinkle and skin lesion detection problem will be presented, going from the most naive one to the most modern and innovative one. As the detection is only targeting the face, the face mesh pipeline provided by Mediapipe [20] has been used to locate the face present in the pictures and to rule out some areas (such as eye, mouth and nose). Using this tool can open possibilities concerning the modularity of the pipeline. Indeed, one could imagine using it to rule out the beard area if the subject has one. Incorporating Mediapipe [20] in the pipeline also allows to extract the texture map in future works.

3.1.1. Canny Edge detection

Wrinkles are characterised by longitudinal folds on the skin. These mostly appear as darker lines on face images. A first approach to the wrinkle segmentation problem hints at using an edge detection technique to identify the wrinkles. Therefore, using the Canny Edge detection [3] algorithm seems to be a suitable solution to tackle the problem. This algorithm is know as one of the best rule-based edge detection methods. It produces a binary mask containing fine and connected lines corresponding to the edges in the input image.

The Canny algorithm [3] is built from 5 big steps:

1. Applying a Gaussian filter to the image.

2. Computing the magnitude and orientation of the gradient.
3. The non-maxima suppression of the magnitude image will then only keep the local maxima of the gradient in the image.
4. Using double thresholding to distinguish weak and strong edge pixels. The thresholds are entered by the user and will determine how sensitive the algorithm is to edges.
5. Finally, weak edge that are not connected to strong edges are discarded.

Figure 3.1 shows how the Canny edge detection algorithm [3] performs on a face extracted from the FFHQ dataset [18]. The most prominent wrinkles are detected and labelled correctly. However, due to the non-maxima suppression step of the algorithm, the labelled regions are usually only 1 pixel thick. The thinness of the detected wrinkles does not provide a suitable mask for inpainting, because the pixels around the labelled line still suggest that the missing pixels belong to a wrinkle. To solve this, an expansion of those regions with a dilation operation with a 3x3-kernel full of 1s is proposed. This will allow to thicken the regions and hide enough pixel information for the inpainter to create wrinkle-free skin in those spots. This operation can also be repeated to grow sufficiently large masks for wrinkles.

As wrinkles are irregular, dilation may not be enough to get sufficient results. Indeed, insufficient dilation will leave wrinkle artifacts and exaggerated dilation will remove flat skin and decrease the inpainting potential of the mask. Therefore, first an under-approximation of the wrinkle regions is performed with the Canny edge detection algorithm. The labelled pixels will be considered as the ground truth for wrinkles. Then, an over-approximation of the wrinkles are obtained by dilating the mask several times. Pixels outside of the over-approximation are now considered as the ground truth for skin without wrinkles. Finally, the pixels that are contained in the over-approximation and outside of the under-approximation are labelled by using the Mahalabonis distance [22]. Those pixels are getting the label from the set for which the Mahalabonis distance [22] is the shortest. The test performed in the algorithm is the following:

$$(x - \mu_W)^T \Sigma_W^{-1} (x - \mu_W) < (x - \mu_S)^T \Sigma_S^{-1} (x - \mu_S) + B$$

x denotes the value of the tested pixel. μ_W and μ_S both denote the mean values for the wrinkle areas and the skin areas respectively. Σ_W and Σ_S both denote the covariance matrices for the wrinkle areas and the skin areas respectively. B stands for a bias term which allows to penalise

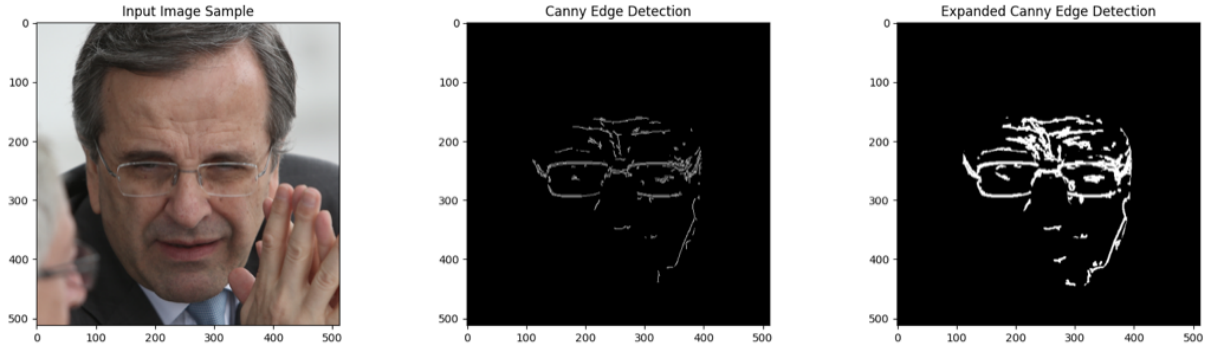


Figure 3.1.: *Comparison of Canny detection*

the skin areas as wrinkle and skin areas sometimes have similar distributions. The bias allows pixels on the boundary to be mostly labelled as wrinkles.

Figure 3.1 shows the results from the expansion algorithm presented earlier. Wrinkles are clearly more perceptible, the mask covers more area on the face and keeps gaps between individual wrinkles in order to maintain high quality skin in areas without wrinkle.

3.1.2. Gaussian Mixture Model and Markov Random Fields

This 2014 presented method [1] is one of the first automatic techniques developed especially for facial wrinkles. Therefore, it is also used as a benchmark for other wrinkle detection methods. It is incorporated in this work as a baseline of how well the presented detection method perform.

(a) Original *(b) Detection* *(c) Original* *(d) Detection*

Figure 3.2.: *Generated masks using the GMM-MRF[1] method.*
(b) and (d) are the masks generated for images (a) and (b) respectively.

Figure 3.2 shows the generated segmentation masks with the GMM-MRF method. These masks have a tendency to be very aggressive in the eye region which contains a lot of colour variations and small skin folds. Wrinkled areas are usually identified correctly, but small artefacts can be found in the masks. These artefacts are an indication of the sensitivity of the method. Wrinkles are detected on baby faces which obviously do not have aging features. Since Gabor filters' responses (see Figure 3.3) are used with the method, most detected areas are composed of thin and straight lines which do not really mimic the actual form of wrinkles. Highly wrinkled areas can thus get saturated and will then be segmented as an opaque patch (see eye regions on Figure 3.2). Saturated areas remove a lot of context around the areas to be inpainted, which can sometimes lead to weird inpainting artefacts.

3.1.3. Jerman Vesselness 2D

The enhancement technique presented in 2016 by Jerman et al. [15] has been initially developed to enhance the visualisation of vascular structures for medical imaging. Since facial wrinkles are characterised by similar forms on 2D images, Jerman's enhancement method has also been used to detect facial wrinkles in the 2020 paper evaluating automatic wrinkle detection methods [7].

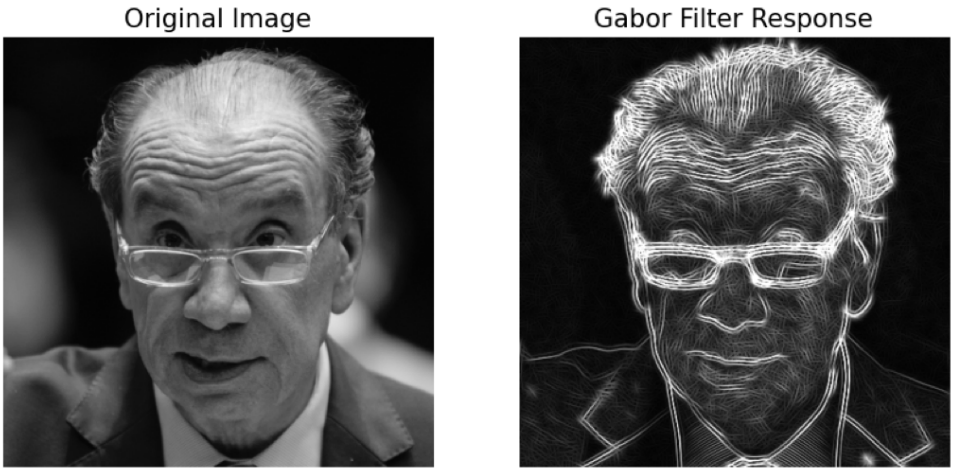


Figure 3.3.: *Gabor filter response*

Using the Jerman detection technique for faces requires to adapt the input parameters. Especially τ which is a cutoff threshold applied to determine which response is high enough to be considered or not. Other parameters to adapts are the σs and the spacing. Each of these parameters have the following impact on the detection process:

- σ: smaller σs capture smaller variations, while bigger σs will capture bigger variations. Similar to a Gaussian filter, this parameter determines which kind of variation between pixels is relevant to the detection.
- Spacing: determines the frequency of the output filter in the xy-coordinates. Higher spacing means higher frequency and thus finer grains in the detection mask. In other words, the spacing determines the thickness of the wrinkles that will be considered relevant by the detection method.
- τ: determines how sensitive the mask will be to the detected features. A high value corresponds to a sensitive mask. $\tau \geq 1$ will almost always result in a fully opaque mask, since all the responses will be considered by the mask. However, a smaller τ will allow finer detection.

Thanks to these parameters, the method can be adapted to almost all the wrinkles and skin lesions one might want to detect on a person's face. The method is useful to detect not only wrinkles but also pimples and smaller skin irregularities. The masks are usually grainy with tiny spots capturing the skin variations (see Figure 3.4). Some inpainting methods do not work well with such tiny masks and it is then necessary to adapt the parameters to the inpainting as well.

Jerman's detection method is one of the most accurate and computationally efficient (see Table 3.2) wrinkle detection methods. This makes it a very good candidate to be a secondary detection method to be used on top of a deep learning wrinkle detection model. Indeed, while a trained model will mainly focus on wrinkles, the Jerman detection method can be configured to detect unwanted skin lesions. Its use with the Mediapipe [20] face detector allows it to focus on certain face regions by adapting the facial landmarks. Therefore facial hair can also be ignored by the detection algorithm and remain untouched by the pipeline.

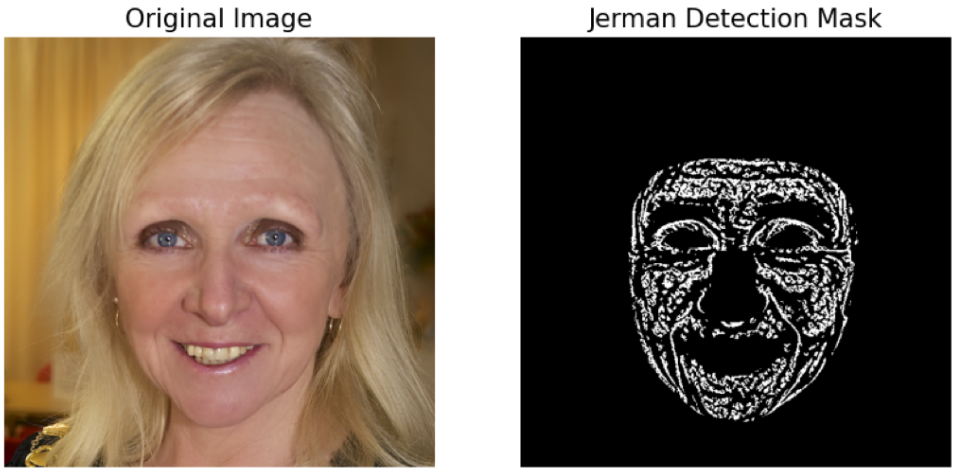


Figure 3.4.: *Masked generated by the Jerman detection technique*

3.1.4. Training the UNet++ Architecture with a Custom Dataset

The UNet++ [42] architecture is a widely known deep learning model for image segmentation. Given a dataset of images with their corresponding segmentation mask, the model can be trained to identify certain structures in similar images. Therefore by building an appropriate dataset, training the UNet++ [42] architecture to identify wrinkles on face images is possible. Such a dataset has already been built in the past and presented in 2022 [33]. For legal reasons, the presented dataset entitled FFHQ-Wrinkles could unfortunately not be retrieved for the current work. Therefore, a custom-made dataset for this problem is being proposed.

As building the dataset requires doing the segmentation by hand for several images, the size of the built dataset is limited. It is then necessary to carefully choose the images to label. To do so, use of statistical data for plastic surgery collected in the USA [26] has been made. Since the goal is to target wrinkles, great focus was drawn towards data concerning facial procedures, especially facelifting. The 55-69 age group represents 64% of the overall facelifting procedures and will thus be the target group for building the dataset. Using aging data for the FFHQ dataset [32], 160 random faces were drawn from the FFHQ dataset [18] according to the distribution presented in Table 3.1. Some young faces have been incorporated into the dataset in order to teach the model that some faces do not contain any wrinkles. This distinction is important for the model to actually locate wrinkles instead of face attribute such as the forehead or the cheeks.

Age Group	10-14	15-19	20-29	30-39	40-49	50-69	70+
# of Images	2	3	10	15	40	45	35

Table 3.1.: *Amount of labelled images depending on age group*

Images were then labelled by hand to create binary masks for the wrinkles using the free software Vectornator [8]. The masks represent the desired output and will serve as ground truth for the model training. A sample of the hand labelled images can be seen on Figure 3.5.

The UNet++ [42] model is composed of an encoder and a decoder, that are both connected

Figure 3.5.: *Samples of hand-labelled images from the dataset. (b) and (d) are the masks for input images (a) and (b) respectively.*

by a series of dense convolutional blocks (See Figure 3.6). The backbone also known as the encoder chosen here is a ResNeXt50_32x4d initialised with pretrained weights obtained with the ImageNet dataset.

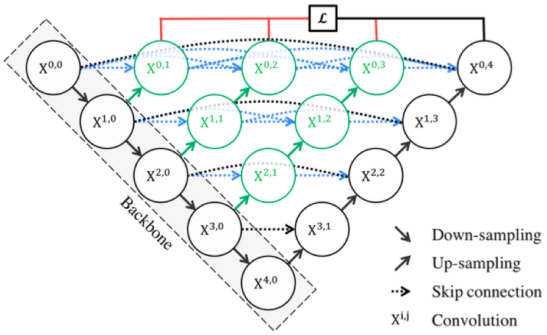


Figure 3.6.: *UNet++ Architecture, taken from [42]*

The training over the custom-made dataset was stopped after 100 epochs with a learning rate of 0.001 and a batch size of 4. Dice loss [34] is used as the loss function with the Adam optimiser. The Dice loss [34] works by comparing the overlapping regions of the predicted mask with the ground truth mask. This famous loss function is well known for its high performance for image segmentation tasks when a ground truth is available, which is considered to be the case with the created dataset. It is computed as follows:

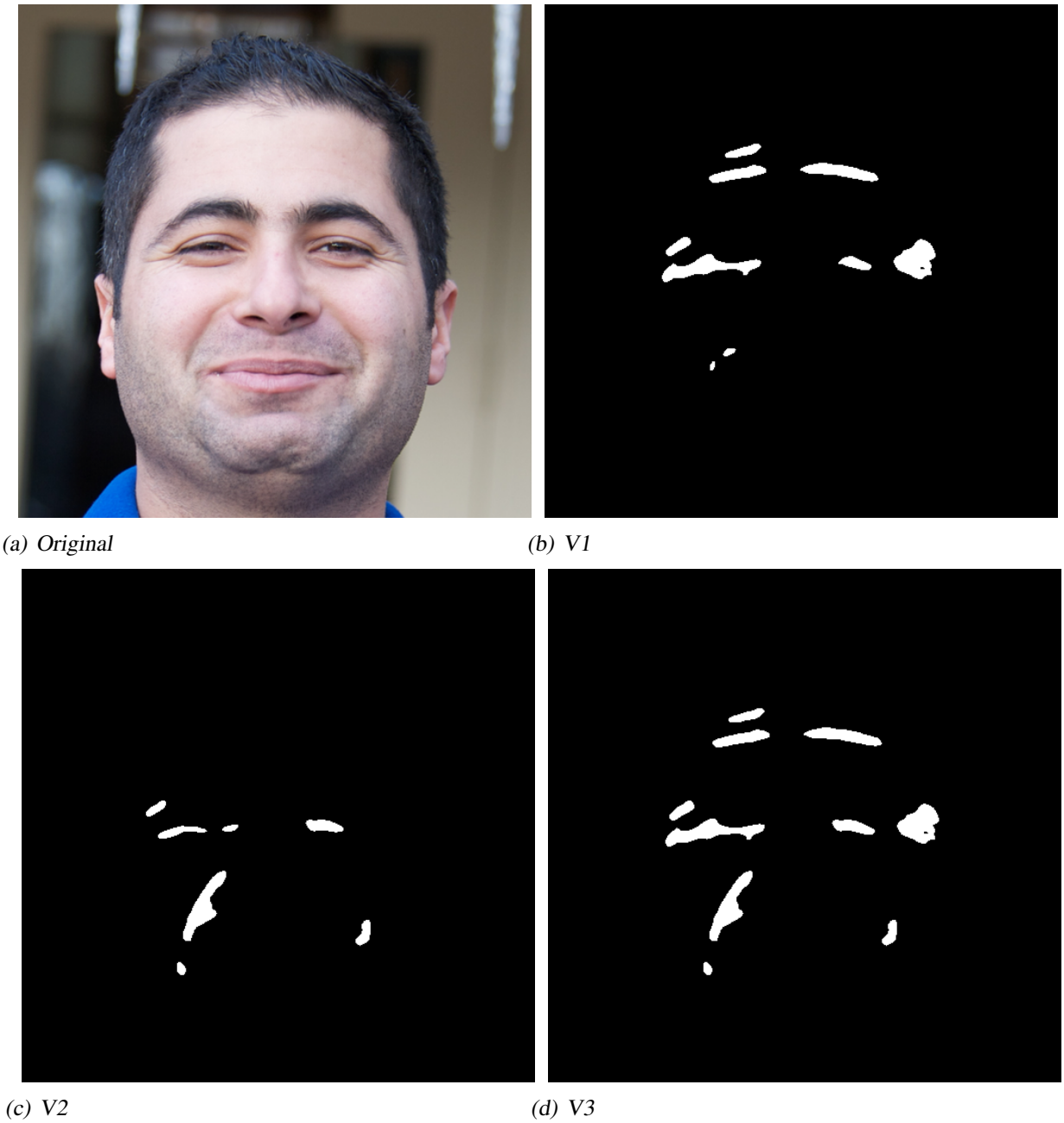
$$L_{Dice} = 1 - \frac{2 \cdot \sum_{i=1}^{N} a_i b_i}{\sum_{i=1}^{N} a_i + \sum_{i=1}^{N} b_i}$$

Let A be the predicted mask and B be the ground truth mask provided by the dataset, such that $N = |A| = |B|$ holds. a_i and b_i both denote elements from the sets A and B respectively with same index i. Considering both masks are binary, pixels of index i are in the overlap of A and B whenever $a_i b_i = 1$.

To leverage the limited amount of data in the dataset, image augmentation has been used. Horizontal and vertical flips, cropping, blurring, contrast changes, noise addition as well as affine transforms have been applied randomly to the data in order to augment it.

Training the model first on 100 images lead to impressive results, especially in the forehead

and eye regions (see Figure 3.7.b). Lacking precision in the lower regions of the face, the model has thus been trained over the whole dataset. The results were indeed more sensitive to lower face wrinkles like the nasolabial folds, but precision was lost in the forehead region (see Figure 3.7.c). Therefore, the best results could be achieved by combining the two masks generated from the two different trainings (see Figure 3.7.d).



(a) Original *(b) V1*

(c) V2 *(d) V3*

Figure 3.7.: *Generated masks using the trained UNet++ models*

3.1.5. Final Detection Comparison

Figure 3.8 shows masks generated with the different detection methods explained previously. Each of these have been chosen to illustrate different configurations and to show the diversity of uses of the presented methods.

The first image shows how the different methods perform on faces wearing make-up. The child is obviously wrinkle-free, so each method should return an almost completely black mask. Yet only the trained UNet++ deep learning model does not pickup the rainbow as being unhealthy skin. The reason might be the way the different methods deal with colours. Indeed, the Canny, GMM-MRF and Jerman methods all convert the input image into grayscale which may convert the rainbow into a grayscale gradient. These methods thus may interpret the abrupt changes in colour as being wrinkles on the skin. UNet++ uses a special preprocessing function to standardise the image, but the three colour channels remain, resulting in a better colour distinction from the model.

The second image shows how the different methods perform on faces with glasses. Here again, the Canny, GMM-MRF and Jerman methods detect the glasses. This also may be due to their altered colour perception. Glasses may also be coloured in a similar way to human skin, therefore some faces wearing glasses were incorporated in the dataset. By doing this, the UNet++ model also learned that glasses may be worn by the subject and that these should remain unchanged. The small periorbital region is prone to wrinkles, but is also a detailed area where wrinkle detection should be extremely precise such that most of the healthy skin can be retained.

The third image shows how the different methods perform on faces with beard. The small granularity and the colours of the beard produce a high frequency area. Therefore, methods using the image gradient or a similar method will be triggered by this face region. Similar to the glasses configuration, UNet++ is the only method which does not fall into this pitfall, since this configuration has been shown in the training set previously.

The fourth image shows the limitations of the UNet++ model. As the exposure is very uneven on the subjects face, the deep learning model is unable to accurately label the eye wrinkles. Although the subject is squinting, the UNet++ method does not locate any wrinkle in the periorbital region. The under eye region is also completely missed by the segmentation model, even though it has been shown to be very effective in this area. For such a specific case, complementing the deep learning model with a targeted version of the Jerman method can be beneficial.

To conclude, the Canny, GMM-MRF and Jerman methods all perform similarly with some clear limitations. These methods are sensitive to fluctuations of the input image and can only be reliable in specific setups from pictures taken in a controlled environment. The UNet++ model on the other hand presents itself with a more robust and more reliable wrinkle detection. Moreover, measurements of the computation time of the different methods (see Table 3.2) show that only the Jerman method is timely more efficient than the deep learning model. The UNet++ model usually performs better on slightly overexposed images, since exposure was not taken into consideration when building the dataset. Therefore, complementing it with the Jerman method can be shown helpful and still be efficient, as both methods can run concurrently.

Method	**Average Computation time** (s / image)
Canny	1.25331
GMM-MRF	2.02229
Jerman	0.112789
UNet++*	0.784308

Table 3.2.: *Average computation time of the detection methods (in seconds per image) over 100 images (on M1 Mac with 16GB RAM) (* executed on the GPU)*

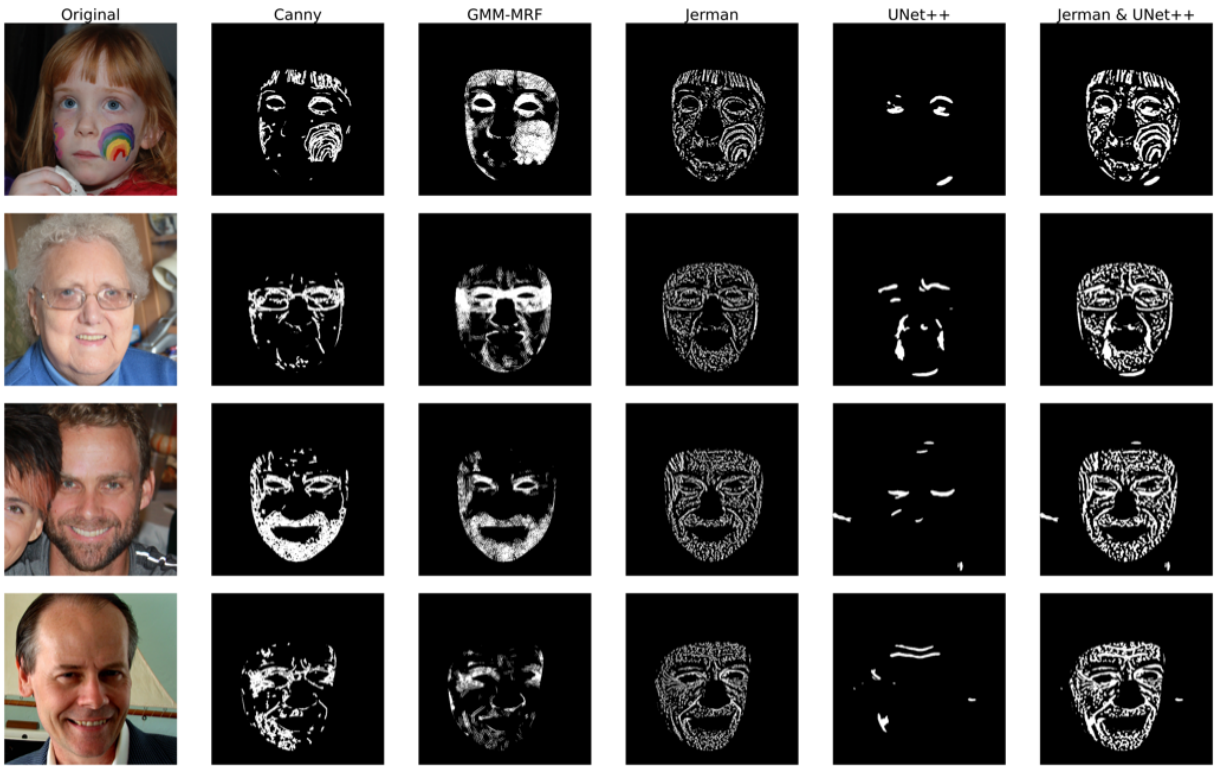


Figure 3.8.: *Comparison of the detection methods in different configurations*

3.2. Inpainting Skin Texture

The inpainting step is the ultimate step for the removal of the previously detected wrinkles and skin irregularities. In this section, the process of inpainting using different methods will be explained, going from a naive pixel-based algorithm to a state-of-the-art deep learning inpainting model.

3.2.1. Naive Pixel-based Approach

Inpainting an image, as fancy as it can possibly be, always boils down to predicting the values of missing pixels in the picture, based on the context provided by the rest of the image. Therefore, a naive way to perform inpainting is to draw the value of the pixel to be inpainted from a distribution built from the neighbourhood of that pixel.

Given the detection mask, an order in which the pixels are going to be inpainted is determined. To get as much context as possible, pixels are going to be inpainted from the outside to the inside, layer by layer (See Figure 3.9, the brighter the pixel is, the later it is going to be inpainted by the algorithm).

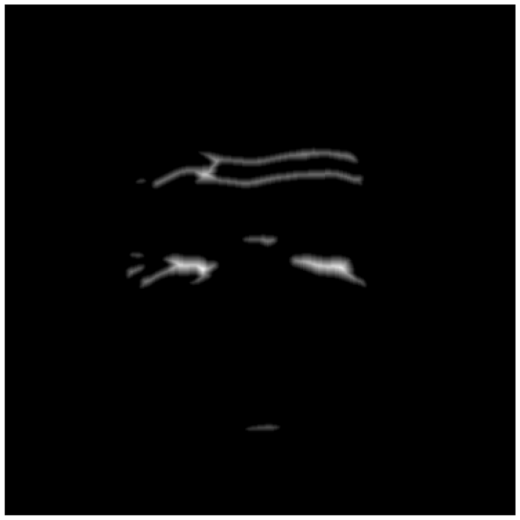


Figure 3.9.: *Order of pixel inpainting*

Each pixel is then drawn from a Gaussian distribution constructed from the "safe" pixels in its neighbourhood. A "safe" pixel is defined as a pixel, that is not labelled by the detection mask, or a pixel that has been already inpainted in a previous iteration of the algorithm. Drawing a pixel value from a Gaussian distribution denotes drawing 3 values each for the red, green and blue channel respectively. 3 Gaussian distributions are thus constructed for each of these channels from the selected neighbourhood.

The size of the neighbourhood can be adjusted to get more or less context. A small neighbourhood might not capture enough variations for the inpainted pixel to have a meaningful value. On the contrary, a too big neighbourhood might incorporate pixels outside of the face region and thus contaminate the distribution with colours not corresponding to the subject's skin tone.

Figure 3.10.: *Comparison of Naive Pixel Inpainting*

Figure 3.10 shows the result of this naive pixel inpainting technique. One can see, that the inpainted areas have colours matching the subject's skin tone. However, as each pixel is drawn independently from all others, the inpainted areas do not have a classical skin texture. The inpainted skin appears fuzzy, as if it had been blurred. Since the pixels are drawn from a Gaussian distribution, a result close to applying a Gaussian blur to the areas could have been expected.

3.2.2. Texture Quilting

Texture quilting for skin texture requires special precautions. As the texture source is the input image itself, only wrinkle-free skin regions need to be considered in order to build the missing texture. Also, as no face is perfectly flat, differences in lighting can really be a challenge for this inpainting technique.

Some facial regions also do not have the same tone as other regions, like the periorbital region which is more prone to shadows or the under-eye skin which can be subject to a darker and blueish colouration. The 2014 paper [1] tackles this issue by using Poisson image editing [30] on those facial areas.

Poisson image editing [30] is a way to seamlessly paste items from one source image onto a target image. Instead of simply changing the pixel values from the target image to be the pixel values from the source image, the Poisson equation is used to make the gradients match. Boundary pixel values of the target image are kept unchanged, while pixel values of the item to be pasted are interpolated such that they fit the target image as well as possible. This method provides a seamless transition between the background (provided by the target image) and the foreground (being the pasted item from the source image). In this research project, the target image is set to be the original input image, while the items to be pasted are the detected wrinkled areas that have been inpainted using texture quilting [6].

Figure 3.11 shows each step of the inpainting algorithm. Quilting [6] serves as a way to cover the gaps created by the detection method with new skin texture. Block boundaries may still appear and skin tone might not match the surroundings of the inpainted area. Therefore, Poisson image editing [30] is used to blend these quilted areas with the rest of the face. Although the improvement of the Poisson method is clearly perceivable, some artefacts are still visible. The result image is also not completely wrinkle-free. Wrinkles on the forehead are well detected by the detection model, but the inpainting algorithm only attenuates their presence instead of erasing them completely.

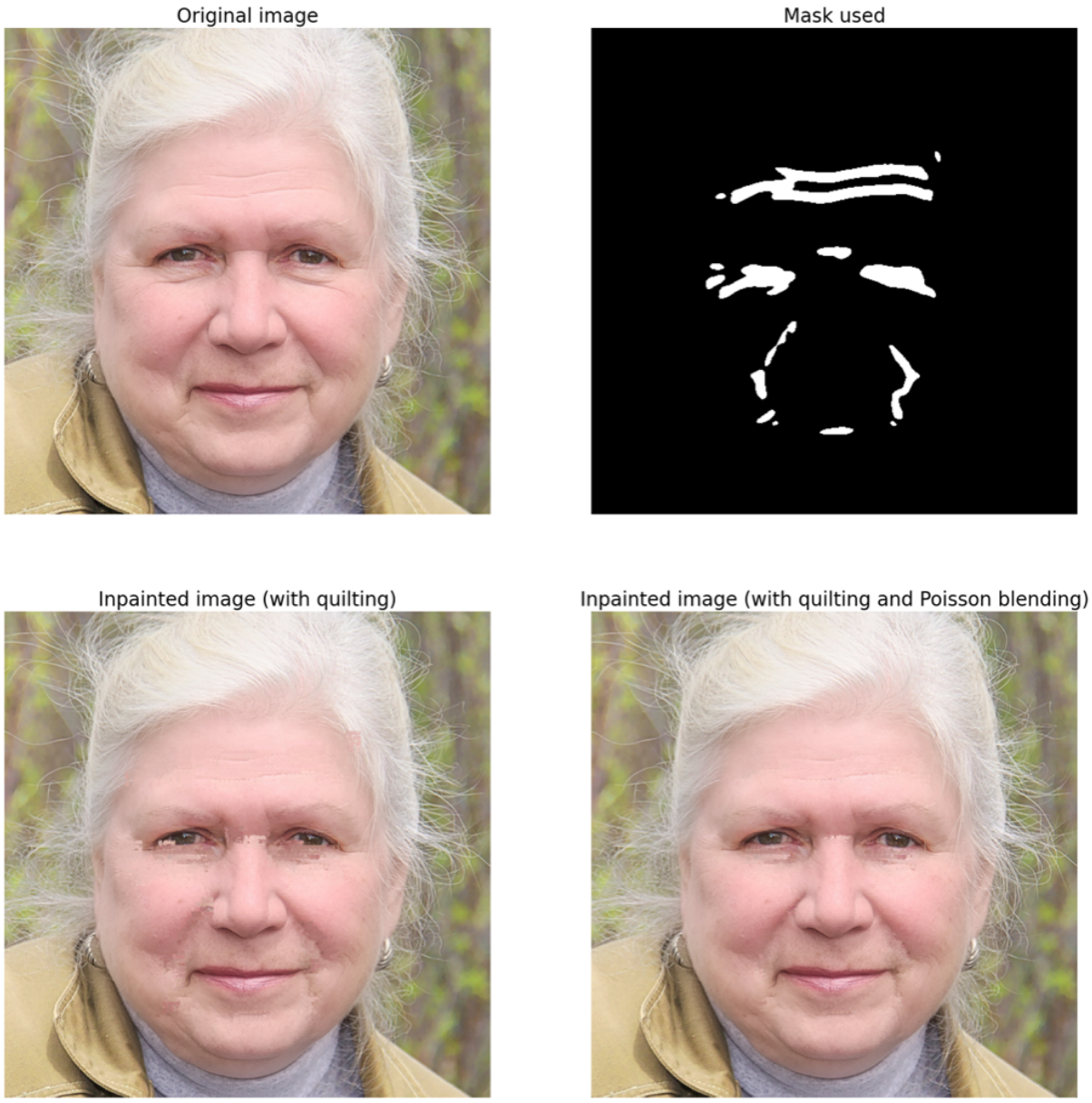


Figure 3.11.: *Texture quilting and Poisson blending*

3.2.3. Training the Deepfillv2 Architecture on a Custom Dataset

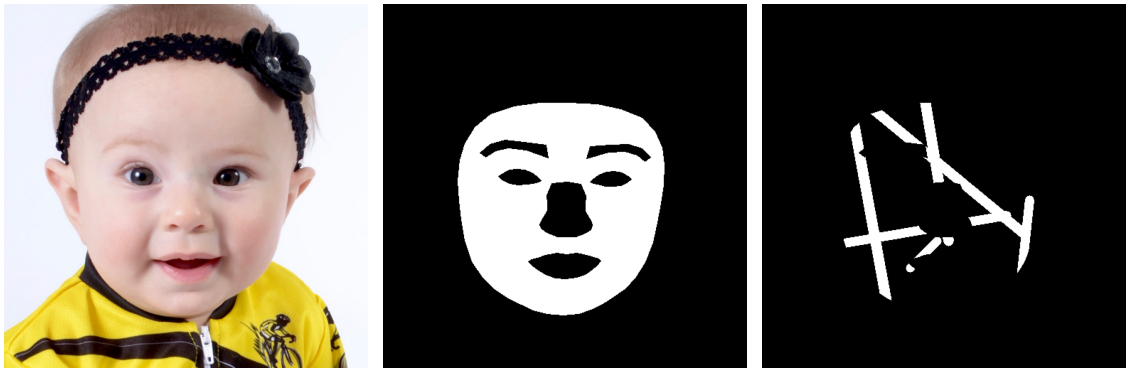
The Deepfillv2 [40] is a deep learning model which is specialised in free-form mask inpainting. Free-form masks can be generated by hand by the user and thus do not necessarily contain polygonal shapes. These correspond to detection masks that can be generated by the aforementioned detection methods, therefore the model seems to be a very suitable inpainting model for the problem defined in this study.

The model has already been trained for inpainting on facial images with the CelebA-HQ faces dataset [17]. As the results experimented with the model on skin texture were a bit blurry and lacking details, a dataset was constructed, in order to further train the model to inpaint fresh and wrinkle-free skin texture. To do so, 1100 young faces were extracted from the FFHQ dataset [18] using the aging data [32] with the age distribution shown on Table 3.3 . The goal is to label fake wrinkles on those wrinkle-free faces such that the model will learn how to inpaint skin texture, but not wrinkles. A binary mask locating the face is also generated for each image (see Figure 3.12.b). An algorithm is then used during the training to create random lines within the generated mask. Since these lines are randomly generated, the data will be diverse and data augmentation will remain simple (see Figure 3.12.c).

Age Group	0-2	3-6	7-9	10-14	15-19	20-29
# of Images	150	150	200	300	200	100

Table 3.3.: *Amount of images chosen depending on age group for the young faces dataset*

The training was performed on a maximum of 1000 batches of size 4. The learning rate for both the generator and the discriminator was kept at 0.0001. Before the training, the generator was initialised with the provided weights for facial inpainting, thus it was pretrained with the CelebA-HQ faces dataset [17]. The training stopped at 285 epochs due to convergence.



(a) Original *(b) Face mask* *(c) Random mask*

Figure 3.12.: *Sample of the young faces dataset*

Figure 3.13 shows results obtained for the two versions of the model. Both results are very similar and the trained version does not really improve the original weights' performance if not worsening it. Indeed, the original version seems to provide a slightly smoother inpainting and

slightly better colours that better match with the overall skin tone. This could be caused by the way wrinkles were simulated in the training. Deepfillv2 [40] being a free-form inpainting model may have lost in flexibility due to the straight lines used to create masks in the newly created dataset. Inpainting tones not matching the actual skin tone and texture may also be caused by the pictures chosen to build the dataset. In fact, differences between young and older faces does not fully rely on the presence of wrinkles, but is also determined by the overall texture and colour of the skin. The model thus may also have picked up the features of young skin texture as the texture to inpaint. In any case, both results can be a bit blurry and do not fully remove the form of the wrinkles.

3.2.4. Evaluation of LaMa Inpainting for faces

The LaMa [35] architecture is specialised in large mask inpainting. Large masks usually cover a big area of the picture and thus remove a lot of information about the image. This architecture also needs to be computationally efficient in order to cover the big areas it has to inpaint. Wrinkle masks generated by the first step of the pipeline can sometimes cover a lot of the subject's face, therefore using an inpainting model capable of recreating a large amount of skin texture might be necessary.

The architecture comes with various weights corresponding to different specialisations. These specialised models are obtained using a variety of techniques and dataset. The most popular and probably the best one regarding quality of the output is called *Big-LaMa*. Figure 3.14 shows how these models perform differently for facial inpainting. Only models trained on the CelebA face dataset [17] were selected, since these are the ones that will perform best for a facial inpainting task. The three compared models perform very similarly and provide a high quality result. The *Big-LaMa* still keep an almost unperceivable form of the removed wrinkle and has a slightly better colour fitting with the surrounding skin. Since the results obtained from the *Big-LaMa* model closely resemble those achieved through an actual facelifting procedure, this model will be referred as LaMa for the remaining of the paper.

Figure 3.15 shows results of the LaMa inpainting used on images labelled by hand. These images were extracted from the dataset, built to train the UNet++ segmentation model. Small wrinkles, like the crow's feet, have completely vanished, while stronger wrinkles, like the nasolabial folds are attenuated. As strong wrinkles are never completely gone after a cosmetic procedure, these results remain realistic even if not all the identified wrinkles are completely removed. Most importantly, the LaMa inpainting leaves almost no noticeable artefacts. This is especially important, because of the Uncanny Valley Effect [12]. These results also show, that a mask labelled by hand is sufficient for the LaMa inpainting model to remove facial wrinkles. The user can thus refine the output of the automatic wrinkle detection method by simply highlighting the undetected wrinkles and skin lesions. These inpainting results also prove the correctness of the custom dataset for wrinkle detection, as the ground truth examples actually perform as expected with a high quality inpainting model.

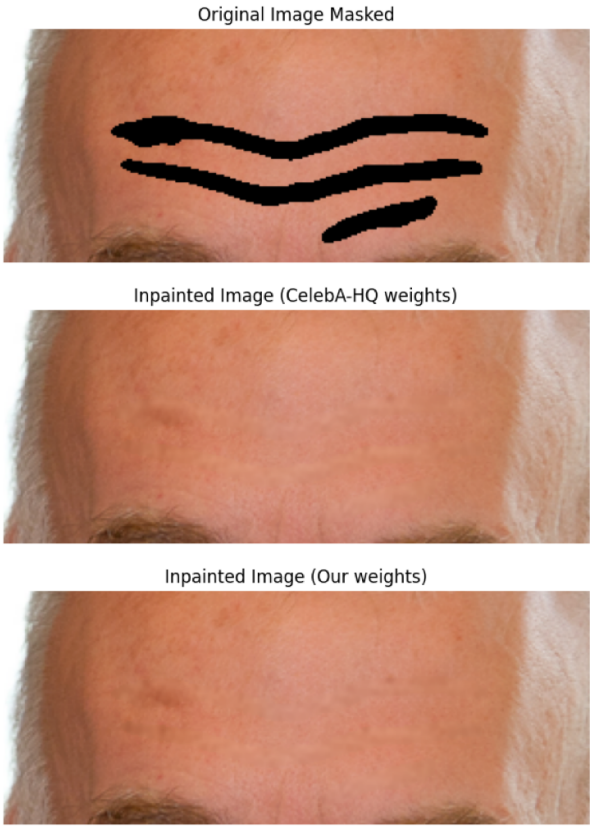


Figure 3.13.: *Comparison of Deepfillv2 inpainting*

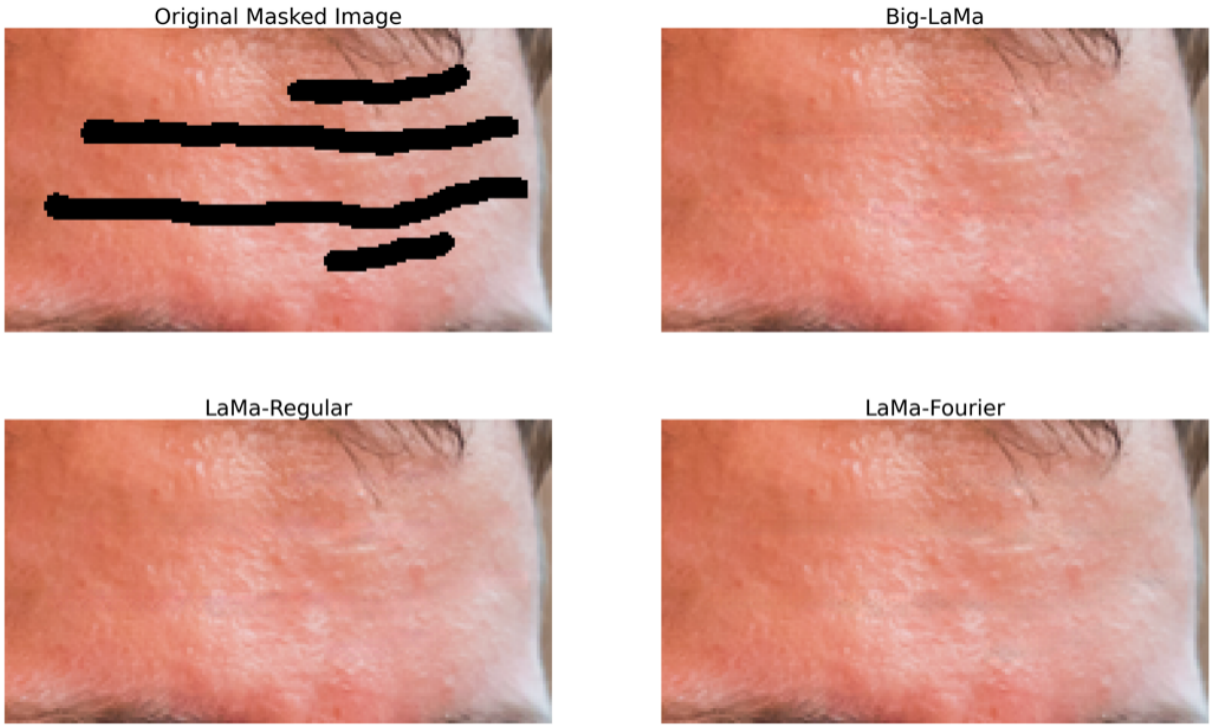


Figure 3.14.: *Comparison of the different LaMa models*

Figure 3.15.: *LaMa inpainted images with hand-labelled mask*

3.2.5. Final Inpainting Comparison

Figure 3.16 shows a sample of images inpainted by all the aforementioned inpainting methods. These images were selected for their strong detection response in various facial areas. Thus the quality of the different inpainting techniques will be compared, based on the different types of wrinkles.

Forehead wrinkles are most probably the most visible and large wrinkles on a face. Covering them requires a large mask. Luckily, since the forehead is a large and mostly flat area, the surrounding skin usually follows the same distribution and texture as the desired replacement skin. Therefore, the pixel-based inpainting and the texture quilting [6] can usually perform well on these areas. However, the deep learning models still show improvements in comparison to these methods. Indeed, as the pixel inpainting and the texture quilting mostly get rid of the shadowing effect of forehead wrinkles, they still leave some perceivable marks after inpainting. The Deepfillv2 [40] model can also produce blurry results and the amount of details inpainted is usually inferior to the ones inpainted by LaMa [35].

Under-eye wrinkles and crow's feet are usually the most targeted wrinkles by facial cosmetic procedures. Therefore an efficient and realistic inpainting of those areas is particularly important for the given use case. These areas are also tricky, because they are subject to high lighting differences and can have a different skin tone than the rest of the face. Since these areas are located close to the eyes and thus close to a variety of colours, the pixel-based inpainting method can produce colourful artefacts. Texture quilting [6] also has some issues in these areas, as

the different skin tones may not allow a seamless stitching of the patches. The deep learning methods are better suited for this task. However, LaMa [35] requires the whole wrinkle to be labelled in order to correctly remove it. If only a part of it is covered by the mask, it will recreate a similar wrinkle. This issue is not to be found with the results coming from Deepfillv2 [40] (see first image from Figure 3.16).

Nasolabial folds and other wrinkles on the cheeks are especially targeted by faceliftings. These represent a similar challenge than the periorbital region. The naive pixel-based inpainting and especially the texture quilting [6] create artefacts around the mouth, which clearly hinder the realism of the result. The deep learning models achieve very comparable results and mostly attenuate the strong nasolabial folds without completely removing them.

Method	**Average Computation time** (s / image)
Pixel	1.87372
Quilting	2.93051
Deepfillv2*	0.525852 (1.67993 on CPU)
LaMa	2.80301

Table 3.4.: *Average computation time of the inpainting methods (in seconds per image) over 100 images (on M1 Mac with 16GB RAM) (* executed on the GPU)*

To conclude, the deep learning models achieve much better results when it comes to inpainting on human faces. The LaMa [35] model provides the most realistic inpainting. Table 3.4 shows measurements of the computation time. The Deepfillv2 [40] model is way faster than the three other methods because it was possible to execute it on the GPU (the average CPU time is still lower than the other methods). For compatibility reasons, LaMa [35] could not be executed using the GPU and is thus one of the slowest inpainting methods. Therefore, one may have to choose between quality and computational efficiency.

Figure 3.16.: *Comparison of the inpainting methods*

3.3. Upscaling Face Images

The upscaling step consists of increasing the size of the image. By doing so, the amount of pixels in the image will increase, which opens the door to increasing the resolution of the image. The overall image quality of the output image should be greater than the one from the input image. In this section, an upscaling baseline will firstly be set up and then different versions of a super-resolution GAN will be presented.

3.3.1. Bicubic Upscaling

Upscaling using bicubic interpolation is applied to the pipeline as a quality baseline. Bicubic interpolation might be computationally more expensive than nearest-neighbour or bilinear interpolation, but it usually yield much better results when it comes to picture quality. Since the pictures are taken in a natural environment, the additional smoothness bicubic interpolation provides is most probably going to result in a much better upscaling.

To determine the value of a new pixel, the bicubic interpolation uses the value of the 16 pixels surrounding it (see Figure 3.17). It allows the algorithm to take account of the actual value of the 4 neighbouring pixels as well as their derivative. A cubic polynomial can then be computed and used to deduct the value of the newly created pixel.

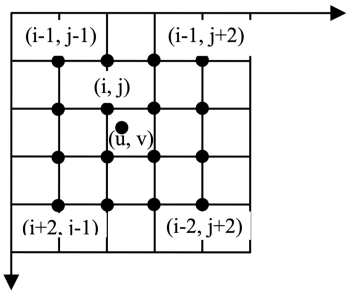


Figure 3.17.: *Diagram of bicubic interpolation algorithm (figure originates from [9])*

Bicubic interpolation is also computationally less expensive than the state-of-the-art super-resolutions GANs. It thus can be used as a way to speed up the pipeline by using it for unimportant parts of the image. The background can for example be upscaled using bicubic interpolation, while the face is being upscaled with a much fancier method.

Figure 3.18 shows results of times 4 bicubic upscaling. A small 50×50 patch of the input image is transformed into a much bigger 200×200 patch. Although pixels are not distinguishable anymore, the bicubic upscaling does not increase the details of the picture. Instead of having a pixelated zoomed image, the bicubic interpolation transforms it into a blurry zoomed image. At this level of zooming, having a sharper outcome as the bicubic upscaled image is desired.

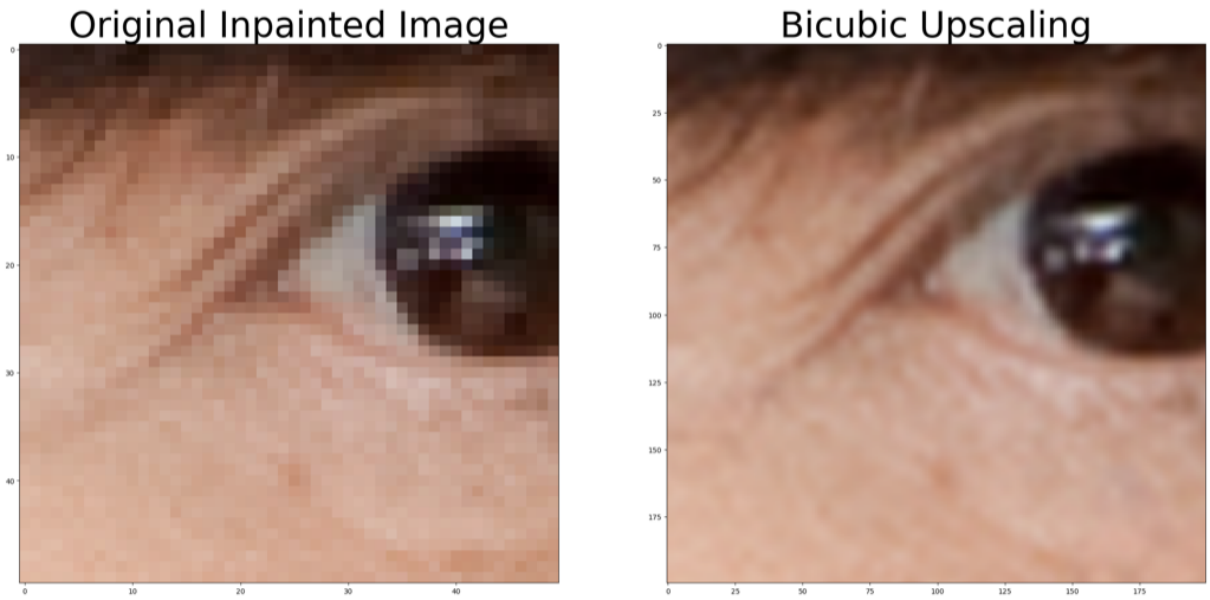


Figure 3.18.: *Example of 4× bicubic upscaling (Axes indicate the amount of pixels)*

3.3.2. Super-resolution GANs

GANs have gained a lot of importance when it comes to super-resolution and image upscaling. These deep learning models are trained to deduce the new pixel values. Unlike interpolation methods, super-resolution GANs are able to gather more context about the pixel distributions, textures and lighting conditions. As a comparison baseline, the upscaling factor will be kept at 4, meaning that the output image will have 4 times the dimensions of the input image.

These models are usually very computationally expensive (even when run on GPU). Therefore, background separation is applied to increase the speed of the pipeline without loss of quality. Using the face detector, a bounding box containing the subject's head is cut out. Only the content of this box is upscaled using a super-resolution model. The rest of the image, which is considered as the background, is upscaled using the bicubic interpolation method. The speedup can be seen on Table 3.5. As two different upscaling methods are used for a same image, the boundary of the bounding box can sometimes be perceived. It denotes the difference of the two upscaled zones.

Figure 3.19 shows a comparison of the Super Resolution GANs. The comparison shows us how these different GANs perform on a given image. The newest RealESRGAN [37] can sometimes produce colourful artefacts that completely change the output image (see the blue spot located on the subject's nose in Figure 3.19). Moreover, it seems to smooth out the skin texture, just like the BSRGAN [41]. The oldest and original ESRGAN [38] on the other hand seems to retain the skin details and does not smooth nor change the colours of the original image.

Figure 3.19.: *Comparison of the Super Resolution GANs*

3.3.3. Final Upscaling Comparison

Figure 3.20 shows a variety of image samples upscaled using the presented upscaling and super resolution methods. To evaluate their performance as efficiently as possible, very detailed face regions were selected. The eye and mouth regions usually present most of the attributes of a human face. Skin folds, hairs and colour boundaries can be found there. A zoom on those areas has been performed and 40px $\times$ 40px samples were taken from the original images and the 160px $\times$ 160px corresponding patches from the upscaled outputs.

As foreseen in the previous subsections, the bicubic interpolation method only blurs the pixelated original image. The three super resolution GANs have very different outputs. The RealESRGAN [37] and the BSRGAN [41] accentuate the edges and smooth out textured surfaces into more uniformly coloured areas. These GANs increase the resolution, but also modify the colours and textures. Their output is mimicking cartoon shapes and colours, which is absolutely not suited for the foreseen medical purpose. Only the ESRGAN [38], which has been developed before, seems to be implementing super resolution according to the desired definition. It enhances the overall texture and sharpness of the image, without falling into a simplification pitfall. Although the output skin texture can appear drier than the input texture, ESRGAN [38] is able to reconstruct thin and sharp shapes like hairs and small variations on the skin.

To conclude, only ESRGAN [38] is achieving the expected super resolution quality. Moreover, it is not the most computationally expensive method. Table 3.5 sums up the average computation time of the presented upscaling methods. The speedup enabled by the background separation is also to be observed. Although the best upscaling method only gets sped up by 1 second on average, this almost represents a 16% speedup.

Method	**Average Computation time with background separation** (s / image)	**Average Computation time without background separation** (s / image)
Bicubic	0.0110244	0.0132221
ESRGAN*	5.13704	6.11023
BSRGAN*	3.66467	6.24298
RealESRGAN*	9.76016	12.5718

Table 3.5.: *Average computation time of the upscaling methods (in seconds per image) over 100 images (on M1 Mac with 16GB RAM) (* executed on the GPU)*

Figure 3.20.: *Comparison of the upscaling methods (Axes indicate the amount of pixels)*

4

Conclusion and Outlook

The current research aimed at removing wrinkles and other skin lesions by editing faces in texture space. Since the results obtained by this work are destined to be used in a medical context, an emphasis has been made on their quality. The elaborated pipeline is built from three main steps: detection, inpainting and upscaling. For each of these steps, a variety of methods have been considered and evaluated. Overall, deep learning models have outperformed more classical approaches. Given they are trained in a thoughtful manner, they can achieve impressively realistic results. The combination of using UNet++ [42], LaMa [35] and ESRGAN [38] successively, for each of the three steps, yields a nearly photorealistic quality. Given the quality achieved, the computational overhead remains user friendly on portable devices. Along with this research, a pipeline is being provided to evaluate each method on different hardware systems.

This work testifies achievable results using state-of-the-art methods. It evaluated and depicted the best pipeline configuration. Obvious next steps would be to tackle the texture mapping process and apply the current pipeline to the generated facial texture maps. These texture maps could then be applied on a 3D model reconstructing the patient's face. The virtual face could then be reshaped in order to simulated cosmetic procedures outcomes such as facelifting, filler injections or eyelid lifting.

A

Additional Content

A.1. Documentation

A.1.1. Implementation Details

The code for this work has been written in Python 3.10.9.
OpenCV [2] was used for applying masks and image processing tasks.
Numpy [11] was also enabled to efficiently perform pixel-based operations on images directly.
Matplotlib [13] was used to export results and illustrate the work.
Scikit-learn [29] and Pytorch [28] were used for the machine learning tasks and also for compatibility with the specific implementations found.
The Pandas [27] library was used to select images based on the data found.
Mediapipe [20] has been used to identify and locate faces in the input images. It also provided a face mesh, that has been used in order to protect specific areas.
The Jerman enhancement filter repo [36] has been used to implement the Jerman method [15].
The UNet++ architecture provided by the segmentation models framework [14] was used. It uses Pytorch [28] to implement different segmentation models and a variety of loss functions, among those the Dice loss has been found.
A 512×512 version of the FFHQ dataset [21] has been used as a way to optimise storage and computation.
Deepfillv2 [40] was tested and trained using the deepfillv2-pytorch repository [25].
The linked repository for the LaMa [35] inpainting method was used to assess the different LaMa models, but the simple-lama python library [43] was used in the end for the pipeline implementation.
The implementations provided directly from the ESRGAN [38], BSRGAN [41] and RealESRGAN [37] papers were used and incorporated into the pipeline.

A.1.2. How to use the pipeline

The pipeline is implemented in the file `texturepipe.py`. Two other files are imported, each of those are implementing each of the two first pipeline steps:

- `detection_algorithms.py` defines functions that perform the detection step of the pipeline.
- `inpainting_algorithms.py` defines and implements the inpainting methods discussed in the current work.

The upscaling step of the pipeline is directly performed in `texturepipe.py` calling the associated libraries.

The pipeline takes 512px $\times$ 512px input images. It can save the outputs of each step directly to the disk. Upon termination, the pipeline prints three tables in the terminal, each containing the computational statistics of the methods executed at each step of the pipeline.

The main block of `texturepipe.py` enables the user to specify the number of images to process, the input directory and the output directory. The output directory should contain three subdirectories `detection`, `inpainting` and `upscaling`. These will contain the results of the pipeline after each of these steps depending on the boolean values set in `save_results`. In the same block, three lists are defined, each one defining which methods have to be executed for the corresponding pipeline step.
WARNING: Increasing the amount of methods used for each step will increase the computational time exponentially, as all possible combinations are going to be tested. Output directories can thus become exponentially big.

There are two ways to call the pipeline from the terminal:

- '`python texturepipe.py`': This command will sample `n_imgs` random images from the input directory and execute the pipeline on each of them.
- '`python texturepipe.py XXXXX`': With `XXXXX` being a specific image from the input folder. This command will execute the pipeline as defined in the main block on the provided image.

A.2. Before/After

Before-After examples generated with the best pipeline:

- Detection: UNet++ [42]
- Inpainting: LaMa [35]
- Upscaling: ESRGAN [38]

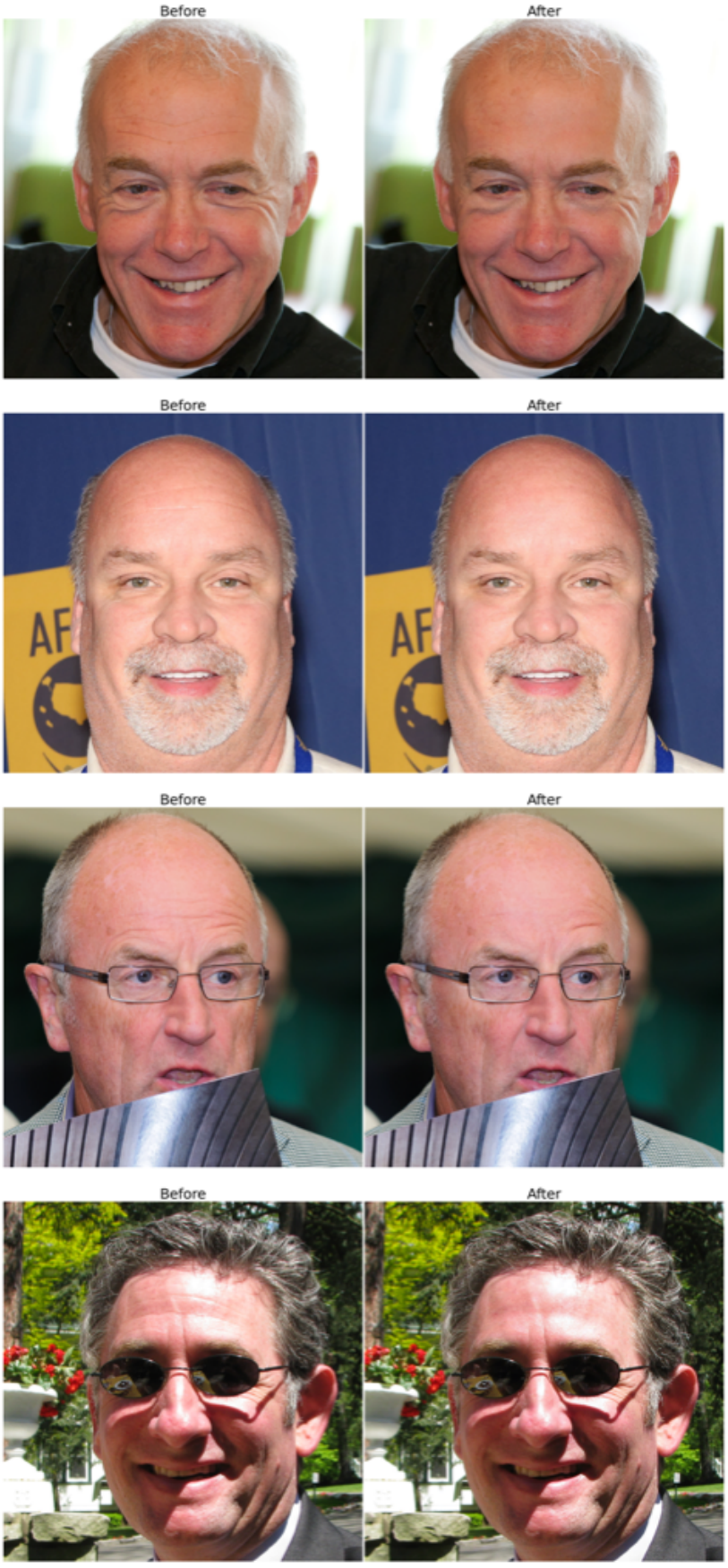


Figure A.1.: *Before-After generated with the best proposed pipeline (I)*

Figure A.2.: *Before-After generated with the best proposed pipeline (II)*

Figure A.3.: *Before-After generated with the best proposed pipeline (III)*

Figure A.4.: *Before-After generated with the best proposed pipeline (IV)*

Bibliography

[1] Nazre Batool and Rama Chellappa. Detection and inpainting of facial wrinkles using texture orientation fields and markov random field modeling. *IEEE Transactions on Image Processing*, 23(9):3773–3788, 2014.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8:679 – 698, 12 1986.

[4] Chenjie Cao, Qiaole Dong, and Yanwei Fu. Learning prior feature and attention enhanced image inpainting, 2022.

[5] Lu Chi, Borui Jiang, and Yadong Mu. Fast fourier convolution. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4479–4488. Curran Associates, Inc., 2020.

[6] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 341–346, New York, NY, USA, 2001. Association for Computing Machinery.

[7] Remah Elbashir and Moi Hoon Yap. Evaluation of automatic facial wrinkle detection algorithms. *Journal of Imaging*, 6:17, 04 2020.

[8] Linearity GmbH. Vectornator. `https://www.vectornator.io`, 2022.

[9] Dianyuan Han. Comparison of commonly used image interpolation methods. *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*, 03 2013.

[10] Dianyuan Han. Comparison of commonly used image interpolation methods. *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*,

03 2013.

[11] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[12] Heloir and Alexis Héloir. The uncanny valley, 2007.

[13] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[14] Pavel Iakubovskii. Segmentation models pytorch. `https://github.com/qubvel/segmentation_models.pytorch`, 2019.

[15] Tim Jerman, Franjo Pernuš, Boštjan Likar, and Žiga Špiclin. Enhancement of vascular structures in 3d and 2d angiographic images. *IEEE Transactions on Medical Imaging*, 35(9):2107–2118, 2016.

[16] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

[18] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019. `https://github.com/NVlabs/ffhq-dataset`.

[19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[20] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for perceiving and processing reality. In *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019*, 2019. `https://developers.google.com/mediapipe`.

[21] Hector Marinez. Flickr-faces-hq dataset (ffhq), 2020. `https://www.kaggle.com/datasets/arnaud58/flickrfaceshq-dataset-ffhq`.

[22] G. Mclachlan. Mahalanobis distance. *Resonance*, 4:20–26, 06 1999.

[23] Choon-Ching Ng, Moi Hoon Yap, Nicholas Costen, and Baihua Li. Automatic wrinkle detection using hybrid hessian filter. In *Asian Conference on Computer Vision*, pages 609–622, 11 2014.

[24] Thanh Minh Nguyen and Q. M. Jonathan Wu. Fast and robust spatially constrained gaus-

sian mixture model for image segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(4):621–635, 2013.

[25] nipponjo. deepfillv2-pytorch, 2022. https://github.com/nipponjo/deepfillv2-pytorch.

[26] American Society of Plastic Surgeons. Plastic surgery statistics report. *ASPS National Clearinghouse of Plastic Surgery Procedural Statistics*, pages 15–20, 2020. https://www.plasticsurgery.org/documents/News/Statistics/2020/plastic-surgery-statistics-full-report-2020.pdf.

[27] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[30] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *Proceedings of ACM SIGGRAPH*, 22(3):313–318, 2003.

[31] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[32] royorel. Ffhq aging dataset, 2021. https://github.com/royorel/FFHQ-Aging-Dataset.

[33] Marcelo Sanchez, Gil Triginer, Coloma Ballester, Lara Raad, and Eduard Ramon. Photorealistic facial wrinkles removal. In *Asian Conference on Computer Vision*, 2022.

[34] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248. Springer International Publishing, 2017.

[35] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. *arXiv preprint arXiv:2109.07161*, 2021. https://github.com/advimman/lama.

[36] vinhnguyen21. Jerman enhancement filter - python, 2020. https://github.com/vinhnguyen21/python_JermanEnhancementFilter.

[37] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data, 2021.

[38] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018. `https://github.com/xinntao/ESRGAN`.

[39] Dongsik Yoon, Jeong-Gi Kwak, Yuanming Li, David Han, and Hanseok Ko. Difai: Diverse facial inpainting using stylegan inversion. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1141–1145, 2022.

[40] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas Huang. Free-form image inpainting with gated convolution, 2019.

[41] Kai Zhang, Jingyun Liang, Luc Van Gool, and Radu Timofte. Designing a practical degradation model for deep blind image super-resolution, 2021. `https://github.com/cszn/BSRGAN`.

[42] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation, 2018.

[43] Enes Muvahhid Şahin. simple-lama-inpainting, 2022. `https://github.com/enesmsahin/simple-lama-inpainting`.