Computer Vision 2 Assignment 3

Emma Quist 11706724, Haohui Zhang 14124262, Faye Raaijmakers 11848669

June 2, 2022

1 Introduction

2 Assignment

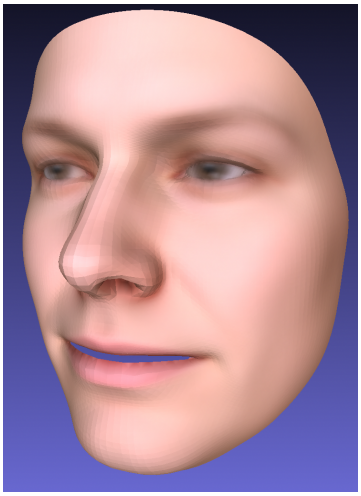
2.1 Deep Learning Based Face Swap

2.1.1 Morphable model

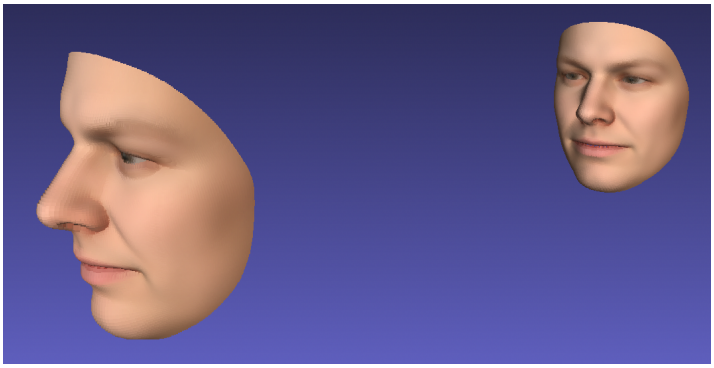
First of all we generated 3D point-clouds based on 30 components of PCA, making use of the Morphable model Basel Face Model 2017. The point-cloud was generated by following equation 1.

$$G(\alpha, \delta) = \boldsymbol{\mu}_{id} + \boldsymbol{E}_{id}[\boldsymbol{\alpha} \cdot \boldsymbol{\sigma}_{id}]\boldsymbol{\mu}_{exp} + \boldsymbol{E}_{exp}[\boldsymbol{\delta} \cdot \boldsymbol{\sigma}_{exp}] \tag{1}$$

The model allowed us to extract the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ for both the facial identity (id) and facial expression (ex). $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ were both uniformly sampled: $\boldsymbol{\alpha} \in \mathbb{R}^{20} \sim U(-1, 1)$ and $\boldsymbol{\delta} \in \mathbb{R}^{30} \sim U(-1, 1)$. In addition to this, the model allowed for extraction of the principal components (PC) of both the facial identity and expression. We reconstructed the point-cloud using 30 facial identity PC's and 20 expression PC's. The reconstruction is visualized in figure 1a. By transforming the reconstruction using a rotation- and translation matrix, we were able to move the projection around in space. An example is shown in figure 1b.



(a) Morphable model



(b) Projected morphable model

2.1.2 Latent parameters estimation

Following, we project the obtained point-cloud to a 2D-plane using the pinhole camera model. By doing so, we obtained facial landmarks that depend on the assignments of α, δ, ω and τ. The estimation of the parameter values could be optimized by usage of a deep-learning model. We initialized a neural network using the following loss functions:

$$\mathcal{L}_{fit} = \mathcal{L}_{lan} + \mathcal{L}_{reg} \tag{2}$$

$$\mathcal{L}_{lan} = \frac{1}{68}\sum_{j=1}^{68} ||\boldsymbol{p}_{k_j} - \boldsymbol{l}_{\boldsymbol{k}}||_2^2 \tag{3}$$

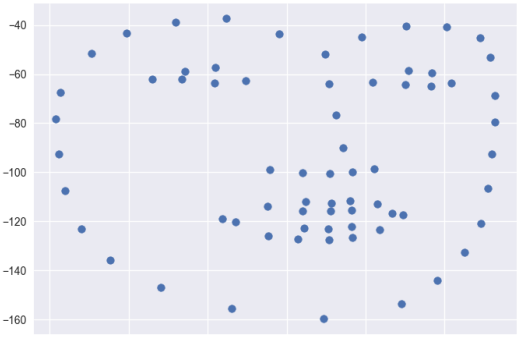
$$\mathcal{L}_{reg} = \lambda_{alpha}\sum_{i=1}^{30}\alpha_i^2 + \lambda_{delta}\sum_{i=1}^{20}\delta_i^2 \tag{4}$$

The parameters were initialized in the following way:

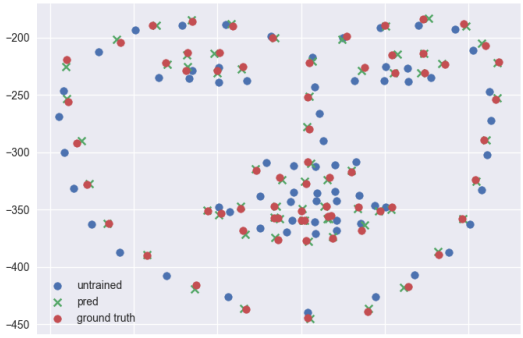
1. $\alpha \sim Uniform(-1, 1)$
2. $\delta \sim Uniform(-1, 1)$
3. ω = [0, 0, 0]
4. τ = [[0, 0, -500].

The model was trained for 2000 epochs and found to converge after about 140 epochs (see figure **??** for the training loss curve). The model was trained using the ADAM optimizer and a learning rate of $\epsilon = 1$ for the ω and τ parameters, and $\epsilon = 0.1$ for δ and α. Both λ_{alpha} and λ_{delta} are initialized with 0 and are fixed throughout the entire process. The tuning of these hyperparameters affect the degree of constraint of the facial features of the final results. However, as we observe in figure 3a, the inital prediction (untrained; blue) and the ground truth (red) are almost identical already. This is the result of using a very straight aligned image of someone with very mainstream features (figure 2. It was therefore unneccesary to put any additional constraints on the facial features, resulting in the choice to initialize λ_{alpha} and λ_{delta} with 0. In addition to this theoretical foundation, experiments using higher values for the hyperparameters resulted in final predictions with higher errors. The final results of the model are shown in figure 3b.

Figure 2: Input image



(a) Projected landmarks initial parameters



(b) Projected landmarks learned by model

2.1.3 Texturing

The texturing of the model was done by bilinear interpolation. Using the latent parameters derived using the model from the previous section, we computed the 2D projection for each point in the image, as shown in figure 5. The final parameters estimated by our model were ω = [2.8778, -8.7076, 4.4029] and τ = [-11.9338, -35.2426, -528.2776]. Using bilinear interpolation, we were able to extract the corresponding RGB values and render the image using the provided rendering function. The results are shown in figure 4.

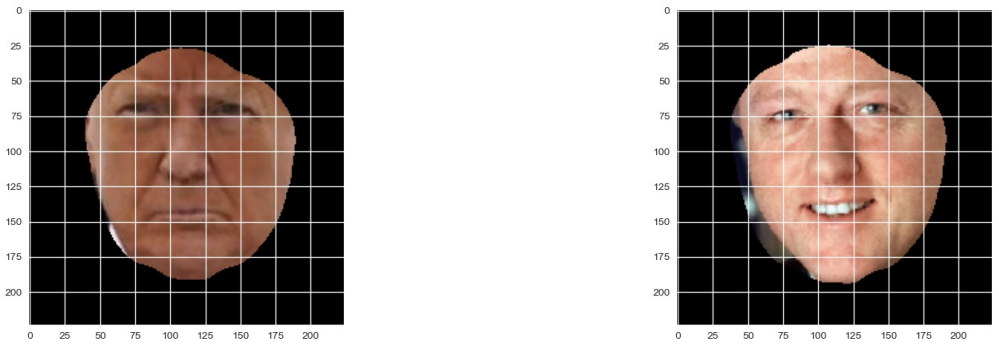
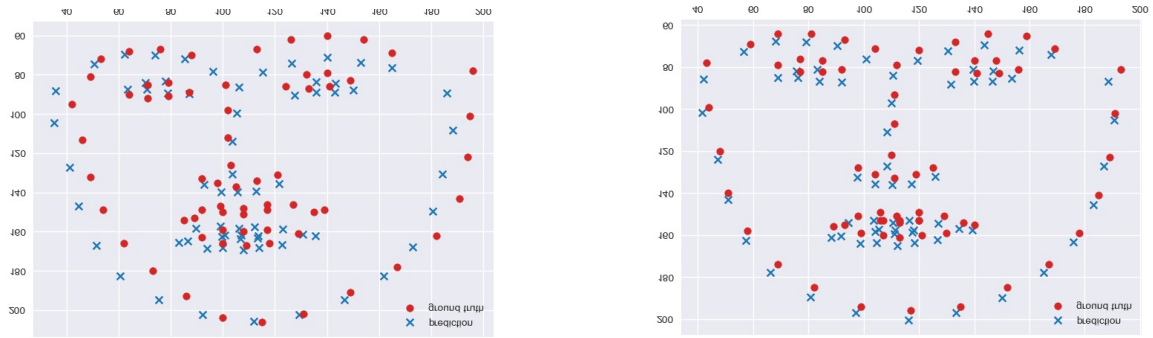


Figure 4: Results of bilinear interpolation



(a) Clinton (b) Trump

Figure 5: Projected landmarks learned by model

2.1.4 Energy optimization using multiple frames

Subsequently, we adjusted the pipeline so that we could apply it on multiple frames. The results are shown in figure 6. We obtained the results by seperately training to obtain the τ and ω parameters.

2.1.5 Face Swapping

We were able to combine the previously explained techniques in order to face swap two images; use the texture of one image and render it using the predicted geometry of another image. The results of this technique are shown in figure 7. As is clearly visible, this technique morphs one face into the another's anatomy. This is visible in the end result, where Clinton's face appears compressed.

Figure 8a shows the results of a deep-learning based face swap. As is clearly visible, the deep learning based method produces way more realistic results. The faces are more aligned with each other. This causes very unnatural borders between skins as apparent in the optimization based results, to disappear. In addition to this, we see that the deep learning method succeeded in actually combining both facial impressions. Figure 8b is another example of optimization-based face swapping. Here again it is clearly visible that this technique does nothing else than aligning anatomy of faces.

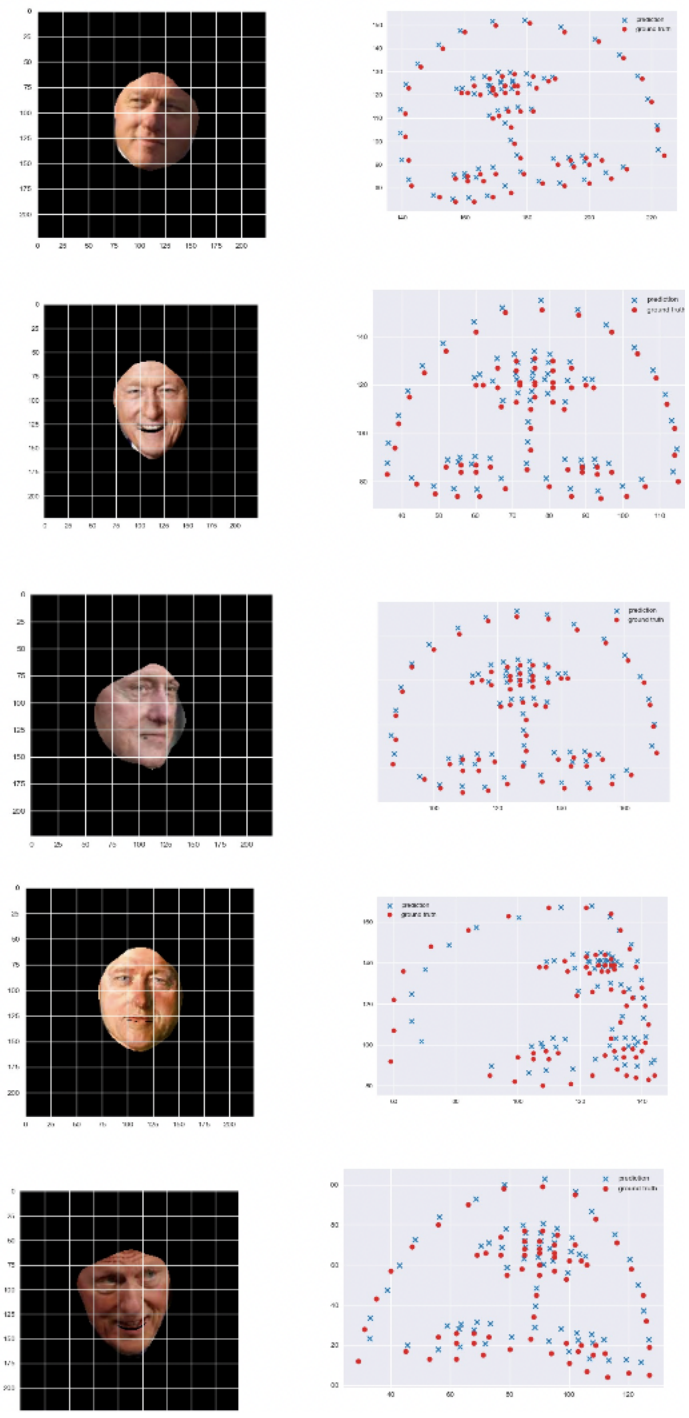
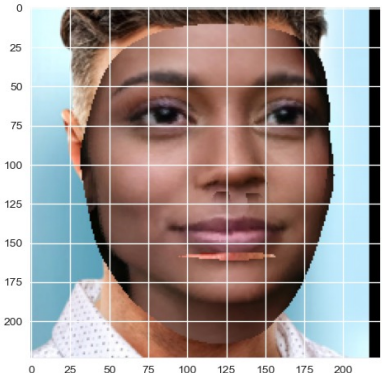


Figure 6: Caption

Figure 7: Optimization-based face swap

(a) Deep learning based face swap



(b) Optimization-based face swap

Figure 8

2.2 Face-swapping on Video

2.3 Blending

2.3.1 GAN Training Pipeline

As the results of the previous methods are theoretically correct, the results look rather unrealistic. To reach for more realistic results, we implemented a GAN. The input of the network is the following:

$$I_{fake} = net(x) \tag{5}$$

$$x = cat(\hat{t}, t, m), \tag{6}$$

where $\hat{t}$ is the 3D reconstructed face, t is the target image and m is the face mask. The ground truth is generated using the Poisson blending function in the OpenCV library. The results of the implementation after one parameter-update are shown in figure 9. Here, from left to right we see the following: the source image, the texture of the source image, the target image, the image predicted by the GAN and the ground truth retrieved by Poisson blending. As can be observed, the generated results look a lot more natural than the results obtained in section 2.1.3. The results after training for 13 epochs are shown in figure 10. In contrast to the results after 1 epoch, we see that the entire face looks more natural now. The skin tones and overall features match throughout the face, making it appear more natural.

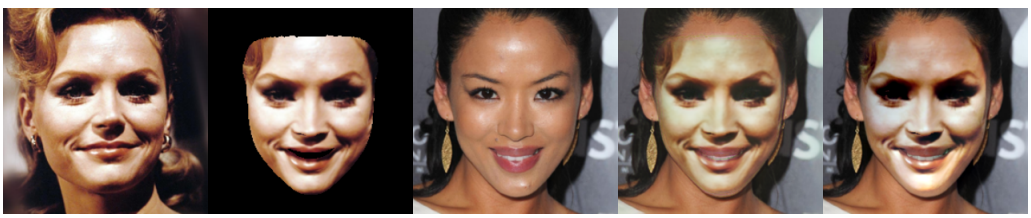


Figure 9: Poisson blending GAN results after 1 epoch

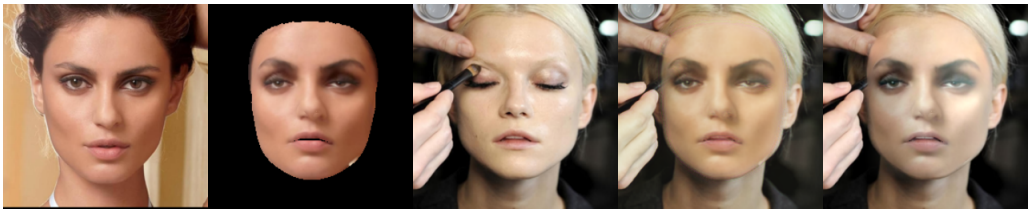


Figure 10: Poisson blending GAN results after 13 epoch

2.3.2 Losses

Later, we included generator losses in our pipeline. The generator's loss is defined in equation 7.

$$\mathcal{L}_G = -\mathcal{L}_D = \mathbb{E}_x[-log(1 - \mathbf{D}(\mathbf{G}(x)))] \tag{7}$$

To see the impact of the inclusion of the loss, we also trained the generator without the loss.

As mentioned before, the ground truth is generated using Poisson blending. However, we also investigated different blending techniques to create the ground truth and their impact on the results quality. First of all we used the OpenCV implementation of Alpha blending, the results are shown in figure 11. As Alpha blending in principal is a naive face swap, we see that the created ground truth (image on the right) looks very unnatural and is basically aligning two different faces in the correct autonomy. However, we see that the result of the GAN still looks very plausible, meaning that the GAN outperforms the ground truth.

Figure 11: Alpha blending GAN results after 1 epoch

Finally, we used the OpenCV implementation of Laplacian Pyramid Blending. The results are shown figure 12. Here, we again see that the ground truth looks fairly unnatural. However, in contrast to the Alpha blending, we see that Laplacian blending reaches the goal of combining facial expressions. Nevertheless, we see that the output of the GAN using Laplacian blending results in less natural results than using Alpha blending. This is the result of the skintone and hair colour not matching in the image.

Figure 12: Laplacian Pyramid blending GAN results after 1 epoch

2.3.3 Analysis

We implemented three methods of face swapping; optimization-based (figure 7), deep-learning based (figures 8b and 10) and naive (figure 13). What we see here, is that the naive face swap produces the worst results. Besides the correct placement of the images on top of each other, the naive face swap does not achieve any other details. For example, we see that the view-angle of both images do not align, the skin tone is incorrect with a hard edge, and the illumination of the images has not been adjusted. The optimization-based method improved the alignment of the faces, where the different

facial features actually line up. For example, when looking at figure 7, we see that the face of Clinton has been warped to match the feature placement of Trump. However, the other problems that were also present in the naive method are still present. The optimization based method was not able to overcome the problem of illumination differences which is clearly visible in the hard edge between the skin tones. The deep-learning based methods yielded the best results and overcame the problems present in the other methods. We see that the GANs were able to deal with illumination differences, as the final output images appear smoothly illuminated. In addition to this, the GANs succeeded in actually combining features of two images instead of just laying them on top of each other. Even though the models produce qualitatively impressive results, for human eyes it is very easy to distinguish them from actual images. This is due to the fact that even though the GANs produce 'correct' images, it lacks to remake faces that are plausible.

Figure 13: Naive face swap

The different blending techniques used to create the ground truth result in deviating results of the GAN. First of all, we see that Poisson blending increases its results after a longer training duration. In figure 9 we see that the GAN had trouble with aligning the different skin tones which was already a visible problem in the created ground truth. However, after 13 epochs (figure 10), we see that the GAN is able to create a more natural overall skin tone. In addition to this, we see that the GAN is able to remove unwanted artifacts, such as the make up brush.

Whereas the naive Alpha blending technique created a very unnatural ground truth, the results were fairly natural. As is clearly visible in figure 11, we see that the GAN has learned to create more natural outputs than the given ground truth. The output of the network after 1 epoch is quantitatively comparable to the output of the Poisson network after 13 epochs, which suggests that using the Alpha loss results in better output images.

Finally, the Laplacian blending technique seemed to result in the most combined facial expressions. The model had clearly learned to use features of both expressions and create a natural new facial expression from here. However, we see that the model has a hard time dealing with the different illuminations of the images. As a result of this, the output image contains a face with a fairly unnatural and weird skin-tone switch on the line where the faces border.

We also trained the GAN using only the generator, and thus removing the discriminator from the network. The results are shown in figure 14. One thing that is immediately visible, is that the GAN was not able to remove the watermark from the final result. A discriminator would most likely distinguish the produced image from being real due to the artefact caused by the watermark. However, due to the exclusion of the discriminator, the image was not filtered out.

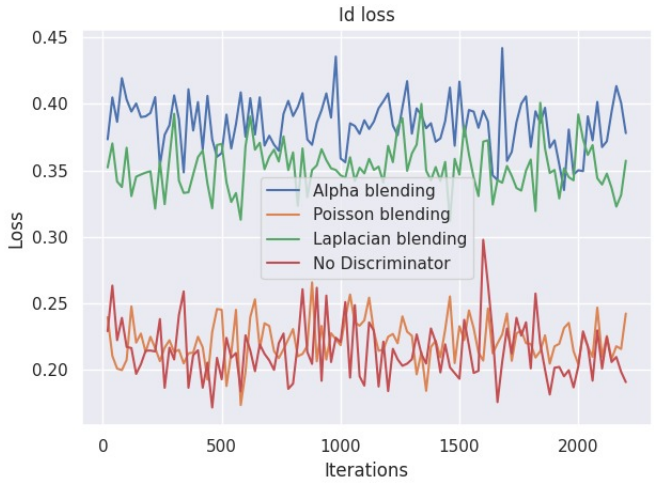


Figure 15: Losses GANs

Figure 14: GAN results without discriminator

The losses of all implemented GANS are reported in figure 15.

Besides the qualitative comparison, we also calculated the Frechet Inception Distance (FID) of all of the models, and found the following results:

1. Poisson 1 epoch: 32.96657022951814
2. Alpha 1 epoch: 33.130686535570135
3. Laplacian 1 epoch: 32.813608178223944
4. No discriminator 1 epoch: 31.856169314981827

Besides the FID, there are plenty of other quantitative metrics that allow for evaluation of the realness of images. A few of them are:

1. Coverage Metric
2. Mode Score
3. AM Score
4. Maximum Mean Discrepancy (MMD)

Another metric that we thought was interesting is the one used in the research of Isola et al. (2017) called the FCN-score. The FCN-score measures how well off-the-shelf recognition systems are able to classify the image correctly. They say the following about this: "The intuition is that if the generated images are realistic, classifiers trained on real images will be able to classify he images correctly as well".

2.4 Segmentation Masks and Analysis

2.5 Bringing it all together

3 References

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125-1134).