Assignment 3

Diego van der Mast
12222933

Vasiliki Vasileiou
13640496

Sameer Ambekar
13616064

Abstract

In this assignment, we implement a face-swapping pipeline with the help of 2D-to-3D reconstruction. Furthermore, we experiment with the construction of realistic results using GANs.

1 Introduction

The target of this assignment is the Deep Learning Face Swap algorithm implementation. However the full implementation of this algorithm has high demands so we firstly we use the given setup to “play" with the algorithm. Consequently, we implemented the optimization-based face swapping both in single images and video. Finally, apply a GAN to achieve a successful and realistic blending.

2 Deep Learning Based Face Swap

Given two images, foreground and background, the task of face swapping is to change identity of the background image into foreground while keeping all other attributes (pose, expression, environment etc.) to be the same. In this part of the assignment we evaluate the results of the given implemented algorithm, parts of which we implement in the following section of this assignment. Following the structure of the given notebook, we firstly set one image as a background and one as foreground on which we detect and visualize the facial landmarks in the following figure:

Figure 1: Apply landmarks

We chose some representative figures with different poses (e.g. 3/4 or en-face), expressions(e.g. smiling, raised eyebrow, serious) and lighting (e.g. light or dark), however the algorithm is very robust and works really well in all of the cases. Based on these landmarks the algorithm crops the faces and as we can observe, the crops are also very accurate, so it is one more way to confirm the robustness of the algorithm of landmarks detection.

( ), .

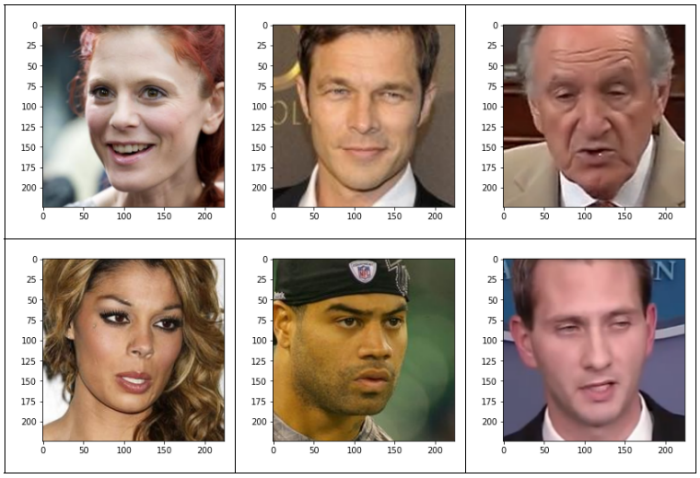

Figure 2: Crop faces

One way to naively swap two faces is to detect the face region of the foreground image, crop it and put it on top of the background image.The masks of the background images are correctly extracted, also they are placed as more correctly as possible, while as we can see there is a good position matching for the eyes and mouth. One the other hand, as it was expected, this method has a lot of problems while background and foreground images could have different poses (e.g. first and second image) or different skin colour. This method could work in an acceptable lever in cases that the faces have the same pose and similar skin colour.

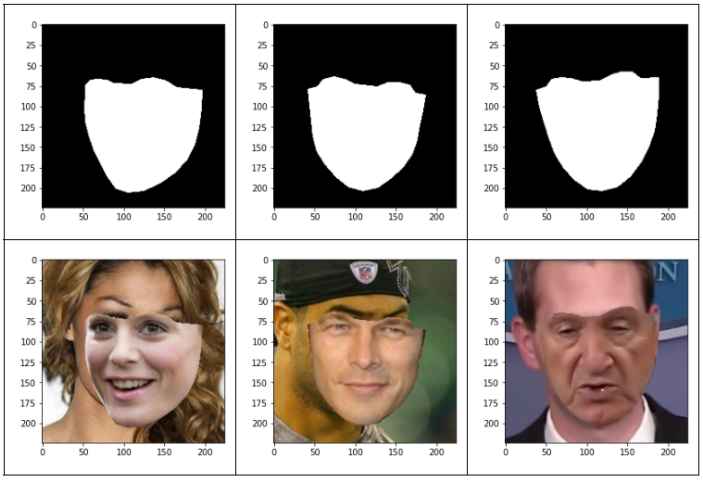

Figure 3: Crop faces

These problems can be faced by a 3D alignment before swapping. A deep learning method is applied to reconstruct a 3D model of two faces and determine their head pose and facial expression, expression and pose of the background face to be applied on the foreground face before merging them together. The results in this case are clearly better but it is still obvious that the final image is non-realistic while there is no perfect match in the shapes and colors of the faces 4.

Figure 4: Blending

Now the results are much more smoother than the previous cases, but as we can see in the second and third image the result is still non-realistic because of huge difference in the pose of the swapped images. On the other hand the first image quite realistic both in expression, pose and color. One example that is very realistic is the face swapping between the US former presidents Trump and Clinton:

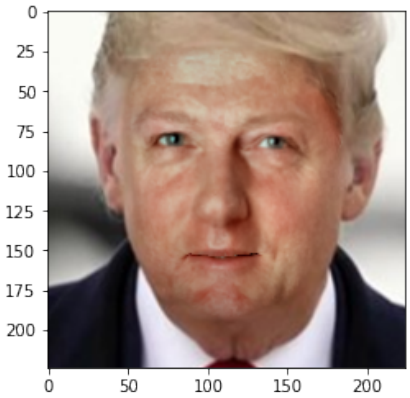

Figure 5: Blending of US presidents

3 Optimization-based Face Swapping

3.1 Morphable model

Human face geometry can be represented as a point cloud of N vertices $\mathbf{G}(\boldsymbol{\alpha}, \boldsymbol{\delta}) \in \mathbb{R}^{N\times 3}$ using a multilinear PCA model. From the provided morphable model, Basel Face Model 2017, we extract 30 PC for facial identity and 20 PC for expression and generate a point cloud using the following equation:

$$\mathbf{G}(\boldsymbol{\alpha}, \boldsymbol{\delta}) = \boldsymbol{\mu}_{id} + \mathbf{E}_{id}\left[\boldsymbol{\alpha} \cdot \boldsymbol{\sigma}_{id}\right] + \boldsymbol{\mu}_{\text{exp}} + \mathbf{E}_{\text{exp}}\left[\boldsymbol{\delta} \cdot \boldsymbol{\sigma}_{exp}\right] \qquad (1)$$

where $\boldsymbol{\mu}_{id} \in \mathbb{R}^{N\times 3}, \boldsymbol{\mu}_{exp} \in \mathbb{R}^{N\times 3}$ are mean neutral geometry (facial identity) and mean facial expression; $\mathbf{E}_{id} \in \mathbb{R}^{N\times 3\times 30}$ and $\mathbf{E}_{exp} \in \mathbb{R}^{N\times 3\times 20}$ are principal components for neutral geometry and facial expression with their standard deviations $\boldsymbol{\sigma}_{id} \in \mathbb{R}^{30}, \boldsymbol{\sigma}_{exp} \in \mathbb{R}^{20}; \boldsymbol{\alpha} \in \mathbb{R}^{30}$ and $\boldsymbol{\delta} \in \mathbb{R}^{20}$ are latent parameters we need to estimate.

We firstly use uniform samples in the range (-1, 1) for both the parameters α and δ which leads to getting results similar to normal face geometries as we can observe in Fig.6. Subsequently, we use larger ranges in uniform distribution, and consequently, the face geometry in these cases is away from the real faces geometry (Fig.7) which is undesirable for our model.

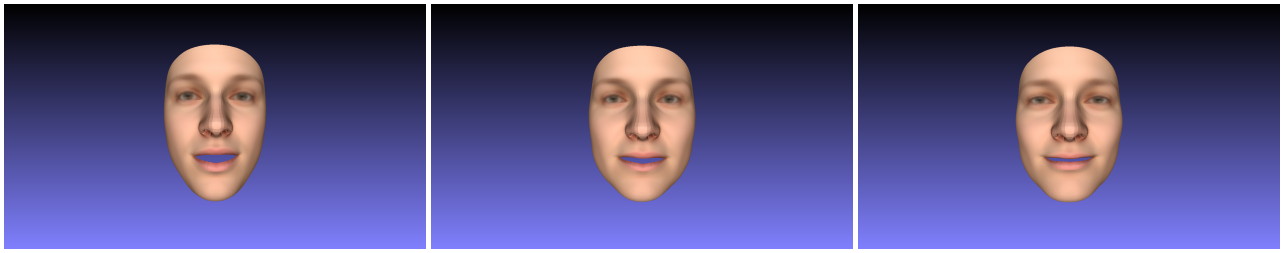

Figure 6: Face geometry generated by uniformly sampled parameters α and δ in range $[-1, 1]$.

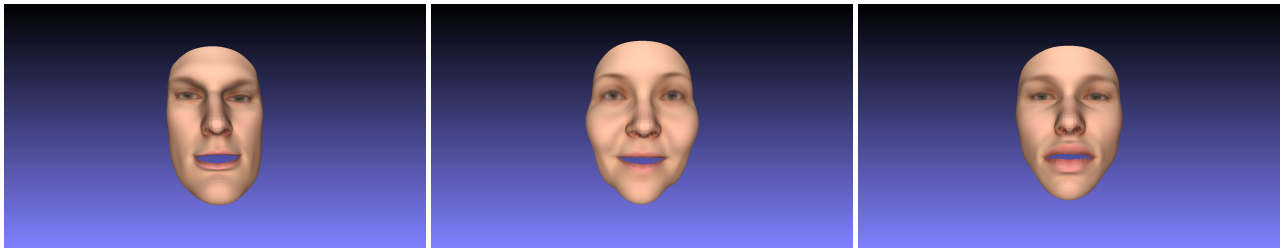

Figure 7: Face geometry generated by uniformly sampled parameters α and δ in random ranges.

3.2 Pinhole camera model

To transform (i.e. translation and rotation) a 3D model we need to apply a transformation matrix on the face vertices $\{x, y, z\} \in \mathbf{G}(\boldsymbol{\alpha}, \boldsymbol{\delta})$. The rotation matrix and translation matrix are respectively:

$$R = \text{raw pitch yaw} = \mathrm{R_x}(\theta_x)\ \mathrm{R_y}(\theta_y)\ \mathrm{R_z}(\theta_z), \quad T = \begin{bmatrix} 1 & 0 & 0 & \mathrm{D_x} \\ 0 & 1 & 0 & \mathrm{D_y} \\ 0 & 0 & 1 & \mathrm{D_z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where,

$$R_{\mathrm{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_{\mathrm{x}} & -\sin\theta_{\mathrm{x}} & 0 \\ 0 & \sin\theta_{\mathrm{x}} & \cos\theta_{\mathrm{x}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathrm{R_y} = \begin{bmatrix} \cos\theta_{\mathrm{y}} & 0 & \sin\theta_{\mathrm{y}} & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_{\mathrm{y}} & 0 & \cos\theta_{\mathrm{y}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathrm{R_z} = \begin{bmatrix} \cos\theta_{\mathrm{z}} & -\sin\theta_{\mathrm{z}} & 0 & 0 \\ \sin\theta_{\mathrm{z}} & \cos\theta_{\mathrm{z}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To apply the transformation matrix to the 3D model we need to transform it in homogeneous coordinates by adding one more dimension with ones. The results we get for right and left rotation of 10° around Oy are visualized in the Fig.8

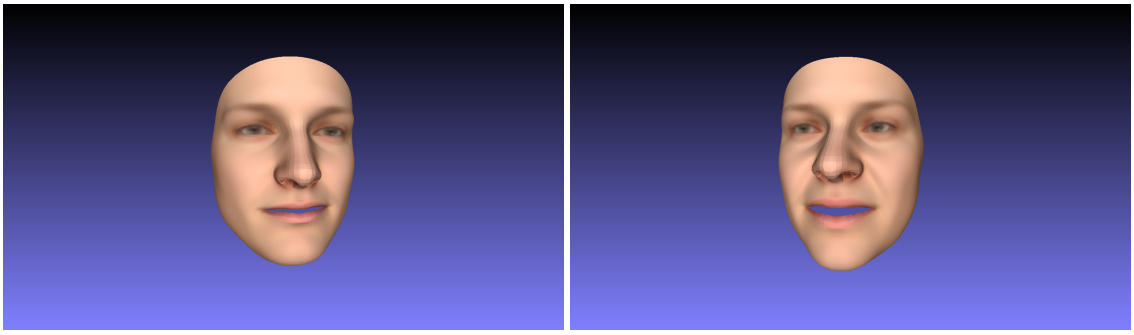

Figure 8: First image shows the rotation of 10° around Oy for while the second the rotation of -10° around the same axis.

After we have applied the transformation (rotation and translation) to the 3D object we can model into a camera plane using a pinhole camera model $\Pi \in \mathbb{R}^{4\times4}$

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ \hat{d} \end{bmatrix} = \underbrace{[\mathbf{V}] \times [\mathbf{P}]}_{\Pi} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

where the viewpoint matrix and the perspective projection matrix are respectively,

$$\mathbf{V} = \begin{bmatrix} \frac{r-l}{2} & 0 & 0 & \frac{r+l}{2} \\ 0 & \frac{t-b}{2} & 0 & \frac{t+b}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad P = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

The matrix elements are calculated by the following equations, where FOV is set to 0.5 and the values of the n (near) and f (far) are selected after some trial and error process.

$$aspect\ ratio = \frac{width}{height}, \quad top = \tan(\frac{FOV}{2}) * near, \quad bottom = -top$$

$$right = top * aspect\ ratio, \quad left = bottom = -top * aspect\ ratio$$

We then get the U,V projection by dividing the $\hat{x}$ and $\hat{y}$ coordinates by the homogeneous coordinate. Using the given file of vertex indexes annotation we obtain the desirable 68 face landmarks of which the visualization is the following (Fig.9):

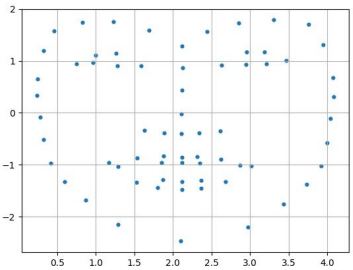

Figure 9: The 68 face landmark of the transferred 2D object.

3.3 Latent parameters estimation

In this part we firstly visualize the ground truth landamarks of a neutral image using the Dlib library and the given function detect_landmark (Fig. 10). Now we build a model to estimate the parameters α, δ, ω and t using Energy minimization. We build a loss

Figure 10: Ground truth landmarks on a neutral image.

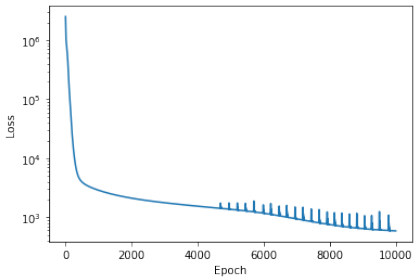

Figure 11: Loss curve of latent parameter estimation on a logarithmic scale

function function where we compare the ground truth landmarks (see previous paragraph) with the current estimation. The energy can be optimized by

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

$$\mathcal{L}_{\text{lan}} = \frac{1}{68}\sum_{j=1}^{68}\left\|\mathbf{p}_{k_j} - \mathbf{l}_j\right\|_2^2, \qquad \mathcal{L}_{\text{reg}} = \lambda_{alpha}\sum_{i=1}^{30}\alpha_i^2 + \lambda_{\text{delta}}\sum_{i=1}^{20}\delta_i^2$$

where $\mathbf{p}_{k_j} = \left\{u_{k_j}, v_{k_j}\right\}$ is a 2D projection of a landmark point k_j from Landmarks68_model2017-1_face12_nomouth.anl and l_j is its ground truth 2D coordinate. Furthermore, we build a model with parameters that should be optimized and we train it using the gradient descent procedure, while the model is differentiable. The forward function of the model uses all the functions we created in previous subsections of this assignment and it returns the U, V projection - corresponding 2D pixel coordinate of each 3D point. Finally, we are fine-tuning the model hyperparameters to obtain the most desirable result. After experimenting, we found that we get the best results by randomly initializing α and β according to $(N)(0, 1)$, initializing translation with $(0, 0, -200)$ and initializing rotation with $(0, 0, 0)$. λ_{alpha} and λ_{delta} were both set to 1. We use a learning rate of 0.01 and run for 10000 epochs. The loss curve in figure 11 shows an asymptotic descent in the first epochs, after which it decreases further in the remaining epochs. The jittering in the later epochs can be explained by the static learning rate. The parameters get over-adjusted which causes the loss to jump up, after which they are corrected again. In figure 12 the optimisation is visualized over three time steps. It is evident that the process aligns the estimation landmarks to the ground truth. The first image shows the effects of the initialization: The prediction is flipped and near the ground truth, but differs greatly in scale, which corresponds to a high loss. The intermediate image at 1000 epochs shows that the scale and rotation have been estimated relatively well, but the shape and expression parameters are noisy. This is also reflected by the loss curve, as it is still relatively high compared to later epochs. In the third image it is clear that the latter parameters approach the ground truth relatively well. Textured results are shown in figure 25. It is shown that the X and Y axes correspond relatively well to a face, but points on the Z axis appear warped. This could be fixed by more extensive hyperparameter tuning, especially the regularization parameters. The colors do not correspond to the actual face, which could be due to a faulty implementation of bilinear interpolation.

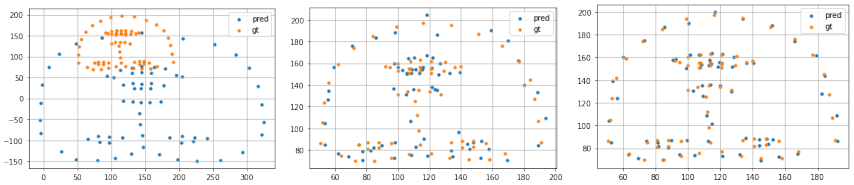

Figure 12: Parameter estimation result compared to ground truth at 0, 1000 and 10000 epochs

3.4 Energy optimization using multiple frames

Using, the optimized parameters from the previous section, we call the morphable model and the pinhole camera model to obtain the 2D projection of the whole image and not only of the landmarks. Subsequently for each channel of a neutral image we obtain the vertex color of each point of the 2D projection by bi-linear interpolation of the neighborhood pixels. Finally, we call the render function to visualize the results.

3.5 Face Swapping

4 Face-swapping on Video

In this section, we intend to use deep learning based methods to perform the task of face swapping.

5 Blending

Since face swapping is not always perfect using traditional methods as obtained in our previous section, we now make use of a blending pipeline that is based on Generative modelling. Specifically, it is named as FS-GAN.

5.1 GAN Training Pipeline and Losses

In this section, we implement the GAN for face swapping and train it on the given dataset and make use of training loss named 'Poisson blending functions' as proposed in [2]. We obtain the results as shown below -

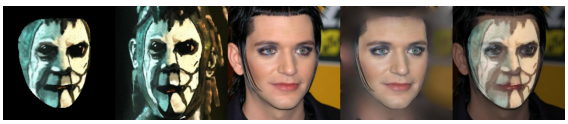

Figure 13: Example of generated images using the GAN based blender.

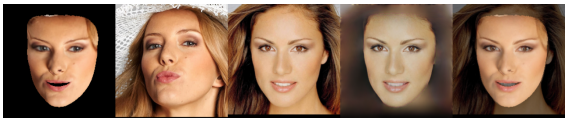

Figure 14: Example of generated images using the GAN based blender.

In the next section we remove the discriminator of the GAN and only use the pretrained generator. Contrary to the our belief, the generator doesn't output just noise but does give reasonable results as shown in 16 17 18. The reason why we the outputs are not just random noise is because the generator is initialized with pretrained weights. The discriminator in theory is responsible to give feedback to the generator about how good/bad the image is. Without it with epochs the results in theory should not move closer towards the target instead keep on repeating the same things that are being done by the generator.

In addition, we observe from 19 due to absence of discriminator that between the 5th and 15th epoch there is no major difference between the image that are generated since there is no feedback from the discriminator the generator just tends to produce the images without any supervision.

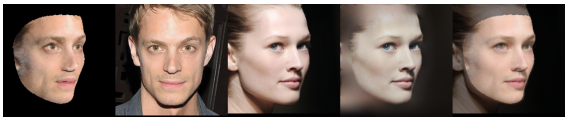

Figure 15: Example of generated images using the GAN based blender.

Figure 16: Example of generated images using the Gan without the discriminator

5.2 Losses

For the initial task done in the previous section, we make use of traditional Poisson blend loss, however in this section we try to experiment with new loss functions such as VGG loss, GAN loss and use the default hyperparameters as described in the given assignment.
We investigate with the following techniques apart from integrating the loss functions as described: Alpha blending : In Alpha blending we overlay a foreground image with transparency over a background image. We swap the ground truth generation with the alpha blending function and make use of OpenCV function to implement this. We obtain the results as shown in Fig. 21 Fig. 22 Fig. 20. We observe that are some artefacts in the generated images.

Laplacian Pyramid Blending : Since not always we work with image of the same size, and we also tend to operate with images in different resolutions we make use of Image pyramids in such cases to overcome this issue. They can be either of the type: Gaussian Pyramid or Laplacian Pyramid. Similar to previous sub section, we introduce Laplacian Pyramid Blending to generate the ground truth results and obtain the following results. Image pyramids tend to give good results unlike other blending since other methods usually do not function well and produce discontinuities between the images. We obtain the results as shown in Fig. 23 and Fig. 24.

5.3 Analysis

5.3.1 Compare traditional method and deep learning method

We observe that when we compare the two methods the GAN based methods is well off at producing better results for face swapping. Since we make use of a huge deep learning network for the task of optimization the GAN model performs better than traditional CV methods. Usually traditional methods require large amount of prepossessing such as noise removal from the image, aligning the image, and often require hyperparameters that are be tuned every time unlike a Deep learning model . Deep learning are usually good at learning these hyper parameters and often do well on train-test data when trained in an appropriate metric, loss function and should achieve convergence. However, it is worthy to note that note always deep learning models can be trained for the given task since usually defining the objective function itself is vague and measuring how well it is performing by a metric is not easy. In addition, GANs are known to generate high fidelity images when compared to traditional methods.

5.3.2 Ablation Study

The network makes use of LG loss, L-idd loss, L-reconstruction loss. In this section, to analyse how absence of one loss function impacts the image generation process we perform an ablation study and obtain results as shown in Fig. 27 and Fig. 28.

Note: For all the ablation study due to lack of compute we run it for 2 epochs and compare the outputs obtained.

From the figures we observe for $\mathcal{L}_{id} = VL\left(I_{fake}, gt\right)$ where it uses VGG based loss between the image generated by the generator and the ground truth, the image generated doesn't degrade so much, probably because other losses such as discriminator based loss are actually better at giving it feedback about what the output should look like.

For L-idd loss, Similarly we do not observe much difference, possibly again because of the same reason as stated.

For L-Reconstruction, we observe that the image quality degrades here in terms of the contrast, luminescence, etc.

Figure 17: Example of generated images using the Gan without the discriminator

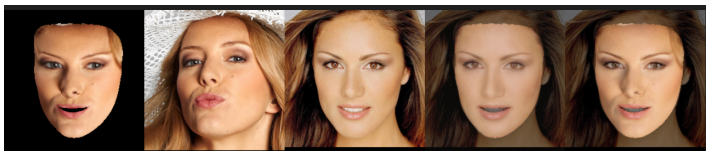

Figure 18: Example of generated images using the Gan without the discriminator.

5.3.3 FID scores

FID and inception distance are two good metrics that are able to capture the distance between given any two distributions. Since FID is a distance metric the score should idelly be as lower as possible. The activations from the Inception V3 model are taken to summarize each image.
We make use of the repository as stated in the assignment PDF to obtain the FID scores. For the case of test images and images generated by the GAN without the discriminator we obtain FID value of 70.40.
For the test images and the images generated by the GAN using alpha blending we obtain FID score of 68. For the test images and the images generated by the GAN using poisson blending we obtain FID score of 120. For the test images and the images generated by the GAN using laplacian blending we obtain FID score of 115.
Good metric to capture realness : From [1] we know that FID performs usually well in terms of robustness, discriminability and also computational efficiency. It considers the first two order moments of the two given distributions. When compared to Inception score fid is more consistent with human judgements and is more robust to noise than inception score. In addition, FID can detect intra class mode dropping. **We have uploaded all the model files at this link - Drive link**

6 Bringing it all together

Throughout the report we have analysed several face swap methods, ranging from out-of-the-box deep learning methods, optimization based methods with hand-engineered features and GANs. For ease of use, we recommend the out-of-the-box method as its straightforward in its usage. This comes at a cost of customizability, however, the optimization-based and GAN-based approaches require expert knowledge for them to work properly. GANs are known for their training pitfalls.

Usually traditional methods require large amount of prepossessing such as noise removal from the image, aligning the image, and often require hyperparameters that are be tuned every time unlike a Deep learning model . Deep learning are usually good at learning these hyper parameters and often do well on train-test data when trained in an appropriate metric, loss function and should achieve convergence. However, it is worthy to note that note always deep learning models can be trained for the given task since usually defining the objective function itself is vague and measuring how well it is performing by a metric is not easy. In addition, GANs are known to generate high fidelity images when compared to traditional methods.

7 Conclusion

In this assignment, we make use of deep learning based methods such as Generative model to perform the task of face swapping. By the usage of GAN with appropriate loss functions and obtaining ground truth using various blending functions such as Poisson, Laplacian we achieve the task of face swap. We observe through an ablation study how various losses are responsible and how exactly they impact the results visually and we quantify the results quantitatively and qualitatively.

8 Self-Evaluation

We shared the points equally and worked collaboratively on questions that we found difficult to implement individually.

Figure 19: We observe no major difference between the images of 5th and 15th epoch due to absence of discriminator

Figure 20: Example of generated images using the Gan with the alpha blending

Figure 21: Example of generated images using the Gan with the alpha blending

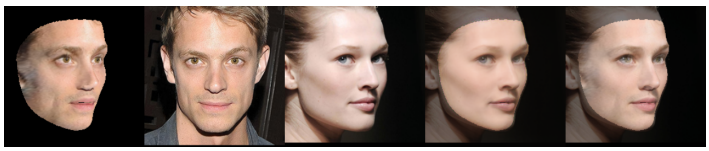

Figure 22: Example of generated images using the Gan with the alpha blending

Figure 23: Example of generated images using the Gan with the alpha blending

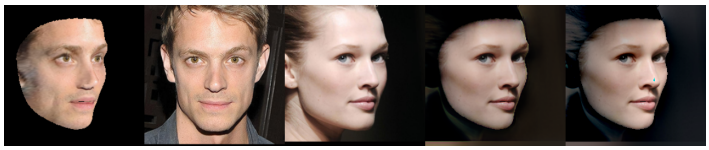

Figure 24: Example of generated images using the Gan with the alpha blending

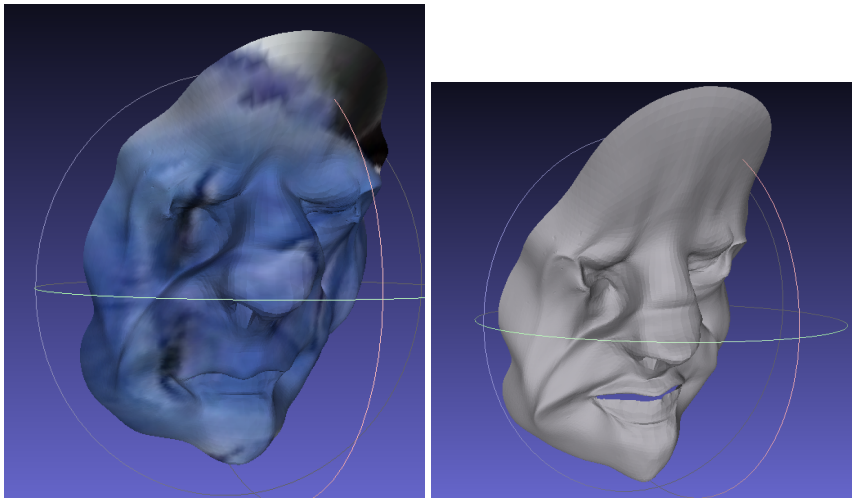

Figure 25: Textured faces with PCA based face reconstruction method

Figure 26: Image obtained by the GAN based method

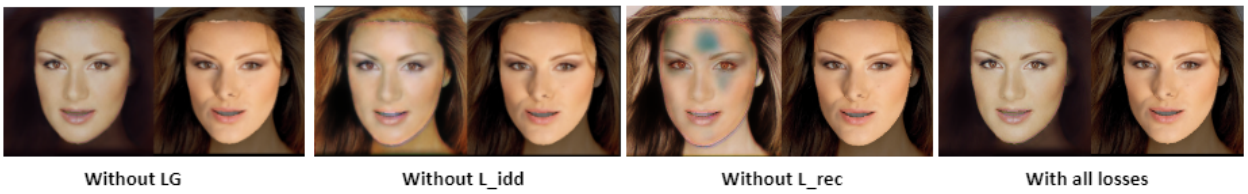

Figure 27: Ablation study of losses

Figure 28: Ablation study of losses

References

[1] A. Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[2] Y. Nirkin, Y. Keller, and T. Hassner. FSGAN: Subject agnostic face swapping and reenactment. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7184–7193, 2019.