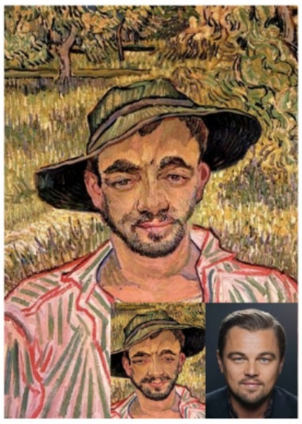
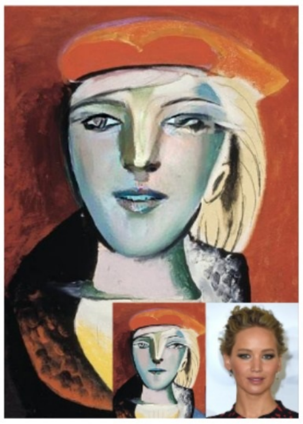
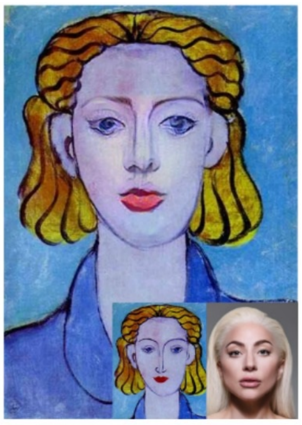
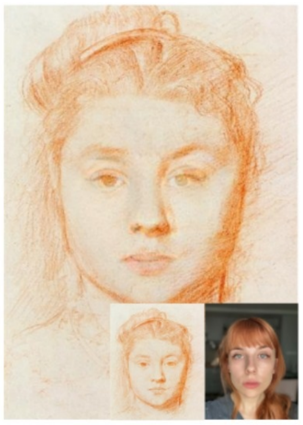
Portrait Face Swap & Style Transfer


Leonardo DiCaprio
by Vincent Van Gogh


Jennifer Lawrence
by Pablo Picasso


Lady Gaga
by Henri Matisse


Ola Piętka
by Edgar Degas

1 Introduction

This project is an innovative exploration that intertwines art and technology. My goal was to transcend the usual two-person face swap, extending its functionalities by swapping a face with one from a renowned painting while retaining the style of the painting. This concept was inspired by the widespread use of image filters in social media platforms like Snapchat, Instagram, and TikTok.

2 Approach

The fundamental steps in the program are *Face Swap*, that entails replacing the face in the portrait with the one on the input photograph, and *Style Transfer* that preserves the original color and texture of the painting onto the swapped face.

2.1 Face Swap

The face swap serves as a preparation step before the style transfer process, as it allows us to switch the face in the portrait (later called target or style) with one from an input image (later called source or content). This procedure is crucial in reducing noise and preparing a clean foundation for the subsequent operations, ensuring that the original attributes of the portrait are preserved.

2.1.1 Keypoints Detection

The first step in the face swap process is the detection of facial keypoints. Typically, we identify 68 facial keypoints that mark the boundaries of the face, eyes, mouth, and nose. However, to reduce potential distortions, particularly around the mouth during the warping process, I adjusted this approach. I chose to utilize only 47 keypoints, intentionally excluding those associated with the lips. This decision helps to create a more authentic face swap. For instance, if the one image had a smiling face while the other had a neutral expression, using all the keypoints could lead to noticeable distortions during the warping process. By refining the keypoints used, we can ensure a more realistic result.

2.1.2 Triangulation

Following keypoint detection, we move on to the crucial stage of triangulation in the face swap process. Here, we take the detected keypoints from both the source image and the target image and compute the Delaunay triangulation. This technique helps later align the source face with the structure of the portrait face.

2.1.3 Affine Transforms

Once we have computed the triangulations, the next step is to calculate the affine transformations for each corresponding triangle. These transformations are essentially mathematical functions that help to manipulate the source face so it aligns with the corresponding triangles of the target face. By applying these transformations to each triangle, we warp the source face to match the portrait's structure and perspective. This process results in an image of the warped source face that fits seamlessly into the portrait, preserving the original look and feel of the artwork while incorporating the new facial features.

2.1.4 Blending

The final step in the face swap process is blending, which includes both color correction and seamless cloning. For color correction, we scale each pixel by the ratio of pixel intensities between the portrait and the warped face for each color channel. This technique helps to harmonize the color tones of the new face with those of the original portrait. After the color correction, we apply seamless cloning, which essentially blends the boundaries of the warped face with the portrait. This process creates a final face-swapped image that is later use as an input to style transfer.

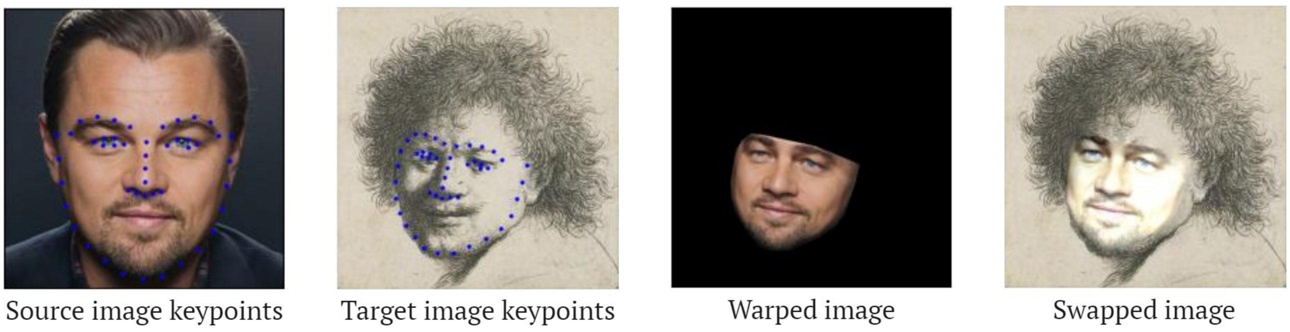


Figure 1: Visualization of stages in Face Swap process.

2.2 Style Transfer

The core part of this project is the style transfer, which applies the original style and texture of the portrait onto the replaced face. My method is composed of two distinct stages, and it is unique in its blending of different approaches, which, to my knowledge, have not been attempted previously.

2.2.1 Stage 1

The first stage of style transfer is all about maintaining the unique style and content characteristics of the original portrait. To achieve this, I create a Laplacian pyramid (Burt and Adelson, 1983) for the source image, the portrait (target), and the generated output image (which I initially set as the content image, but it can also be defined as a noise image).

Starting from the smallest image at the top of the pyramid, I extract features for each image using the VGG19 network (Simonyan and Zisserman, 2015). These features are then used to minimize the loss (Equation 1) at each level. The loss in this stage is a combination of content loss (Equation 2) and style loss (Equation 3), both of which I'll explain later. Since the style loss should be computed only on the area of the swapped face, I use a mask and resize it to match the shape at the current pyramid level.

The output image from one level of the pyramid is then bilinearly upsampled and used as the input for the next computation. I repeat this process of training iterations at each pyramid level until all levels and iterations are completed. The output from this stage then forms the input for the next stage, allowing for a seamless transition between the two stages.

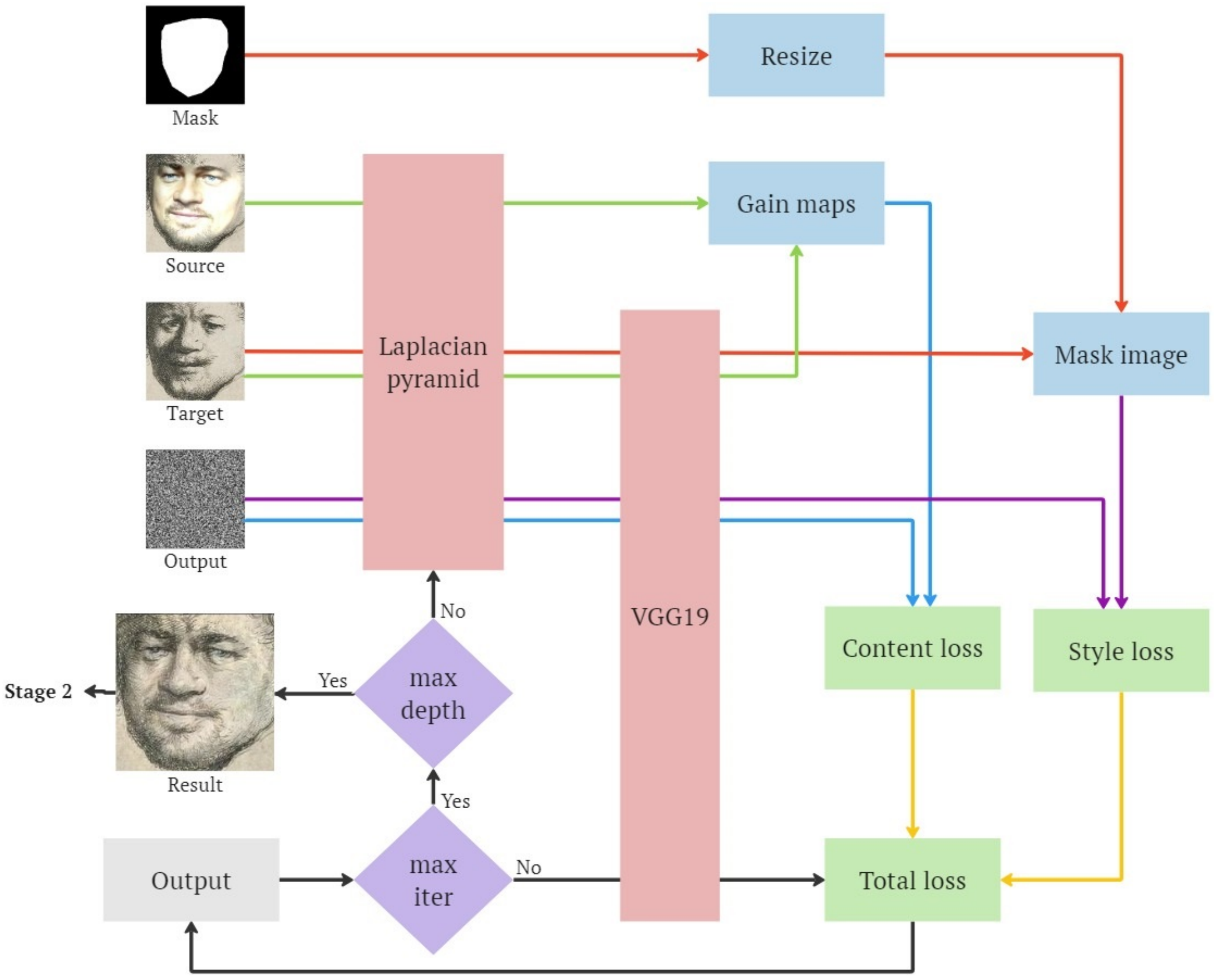


Figure 2: Architecture of stage 1 of the style transfer process.

2.2.2 Stage 2

The second stage of the style transfer is aimed at refining the details. This stage starts with the output from the first stage as the input (source) image. Unlike the first stage, we don't use a Laplacian pyramid here; instead, we run a fixed number of training iterations on the provided images.

In this stage, the computation of the histogram loss (Equation 4) and total variation loss (Equation 5), is added to the existing content and style loss calculations. These additional computations are designed to correct any instabilities that might have occurred during first stage and to further enhance the overall quality of the image. Both the histogram and style loss calculations are carried out only on the face swapped region, necessitating the use of a mask image here as well.

By introducing these additional loss components, we allow the model to make more refined adjustments to the image, resulting in improved details and overall coherence in the final output. This stage runs for a specified number of iterations and produces the final output of the style transfer process.

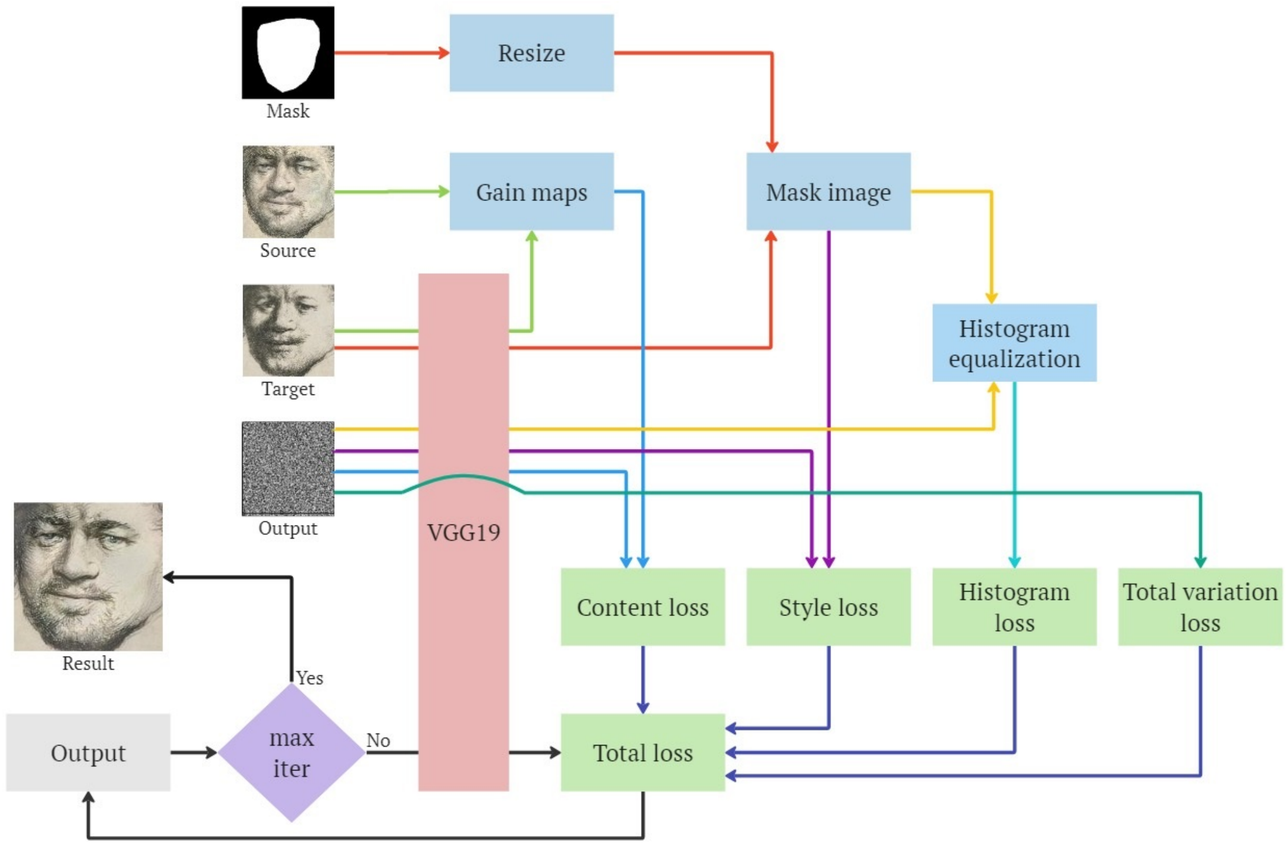


Figure 3: Architecture of stage 2 of the style transfer process.

2.2.3 Loss Function

The loss at each iteration is computed based on four components, each multiplied by its weight:

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style} + \gamma \mathcal{L}_{hist} + \theta \mathcal{L}_{tv} \tag{1}$$

Content Loss plays a crucial role in maintaining the content information of the image during the style transfer process. It is formulated as the Mean Squared Error (MSE) between the modified feature map of the content image and the feature map of generated output image at a specific layer. This concept was based on the idea of adjusting the content image features based on the style image to improve the quality of style transfer (Selim et al., 2016).

In mathematical terms, the content loss function is defined as:

$$\mathcal{L}_{content} = \sum_{l=1}^{L} \|O_l - M_l\|_F^2 \tag{2}$$

$$where \;\; M_l = C_l * F_l \;\; , \;\; F_l = \frac{S_l}{C_l + \varepsilon} \;\; , \;\; \varepsilon = 1e-4$$

Here, O_l represents the feature map of the output image at layer l, M_l represents the modified feature map of the content image at layer l, C_l is the feature map of the content image at layer l, S_l is the feature map of the style image at layer l, L is the number of layers considered for calculating the content loss, and $\| * \|_F^2$ denotes the squared Frobenius norm, which measures the element-wise difference between the two feature maps.

The content loss thus measures the differences between the feature representations of the output image and the modified content image at different layers of the network. Here, the modification involves a multiplication with F_l, which adjusts the feature maps of the content image based on the style image. This ensures that the content of the original (portrait) image is preserved in the final output.

Style Loss plays a significant role in the style transfer process by ensuring that the output image adheres to the artistic style of the provided style image. It measures the difference between the style representation of the generated image (output) and the input style image. It is computed as the Mean Squared Error (MSE) between the Gram matrices of the style image and the output image at each layer of the network, but only within the masked region. This method was proposed by Gatys et al. (2015) in their seminal work on neural style transfer.

The formula for calculating style loss is:

$$\mathscr{L}_{style} = \sum_{l=1}^{L} \|G(S_l) - G(O_l)\|_F^2 \tag{3}$$

$$where \;\; G(X) = XX^T$$

Here, $G(X)$ is the Gram matrix of X, which is a measure of the correlation between different feature maps in X, capturing the style information, O_l represents the feature map of the output image at layer l, S_l denotes the feature map of the style image at layer l, L is the number of layers considered for calculating the style loss, and $\| * \|_F^2$ denotes the squared Frobenius norm, which measures the element-wise difference between the two feature maps.

The style loss thus quantifies the difference in style between the style image and the generated output image. By minimizing this loss during training, we encourage the network to generate an output image that closely matches the artistic style of the style image.

Histogram Loss is another component that contributes to the optimization process, ensuring the preservation of activation histograms. This in turn improves the overall quality of the generated image, reduces artifacts such as ghosting, and accelerates the convergence of the algorithm. This technique is based on the work presented Risser et al. (2017). It is computed as the Mean Squared Error (MSE) between the feature maps of the output image and the histogram-matched version of the style image at each layer of the network, but only within the masked region.

The formula for calculating histogram loss is:

$$\mathscr{L}_{hist} = \sum_{l=1}^{L} \|O_l - H_l\|_F^2 \tag{4}$$

$$where \;\; H_l = histmach(O_l, S_l)$$

Here, H_l is the histogram-matched version of the feature maps of the style image at layer l, O_l represents the feature map of the output image at layer l, S_l denotes the feature map of the style image at layer l, L is the number of layers considered for calculating the histogram loss, and $\| * \|_F^2$ denotes the squared Frobenius norm, which measures the element-wise difference between the two feature maps.

The histogram loss essentially aims to match the distribution of the feature responses of the output image to the distribution of the feature responses of the style image. By doing so, it promotes a more accurate style transfer by preserving the statistical characteristics of the style image in the output image.

Total Variation Loss is used to reduce high-frequency artifacts, effectively smoothing out the noise that can result from the optimization process. This type of loss was introduced in the paper Johnson et al. (2016). It is computed as the sum of the absolute differences (AE) between neighboring pixel values in the output image. Specifically, it sums over both the horizontal and vertical directions, calculating the squared differences between each pixel and its neighbor.

The formula for calculating total variation loss is:

$$\mathscr{L}_{tv} = \sum_{x,y} |O_{x,y} - O_{x,y-1}| + \sum_{x,y} |O_{x,y} - O_{x-1,y}| \tag{5}$$

Here, $O_{x,y}$ denotes the pixel value at position (x, y) in the output image.

In essence, total variation loss encourages spatial smoothness in the generated image. This helps to reduce high-frequency noise and makes the output appear more natural and less pixelated.

3 Results

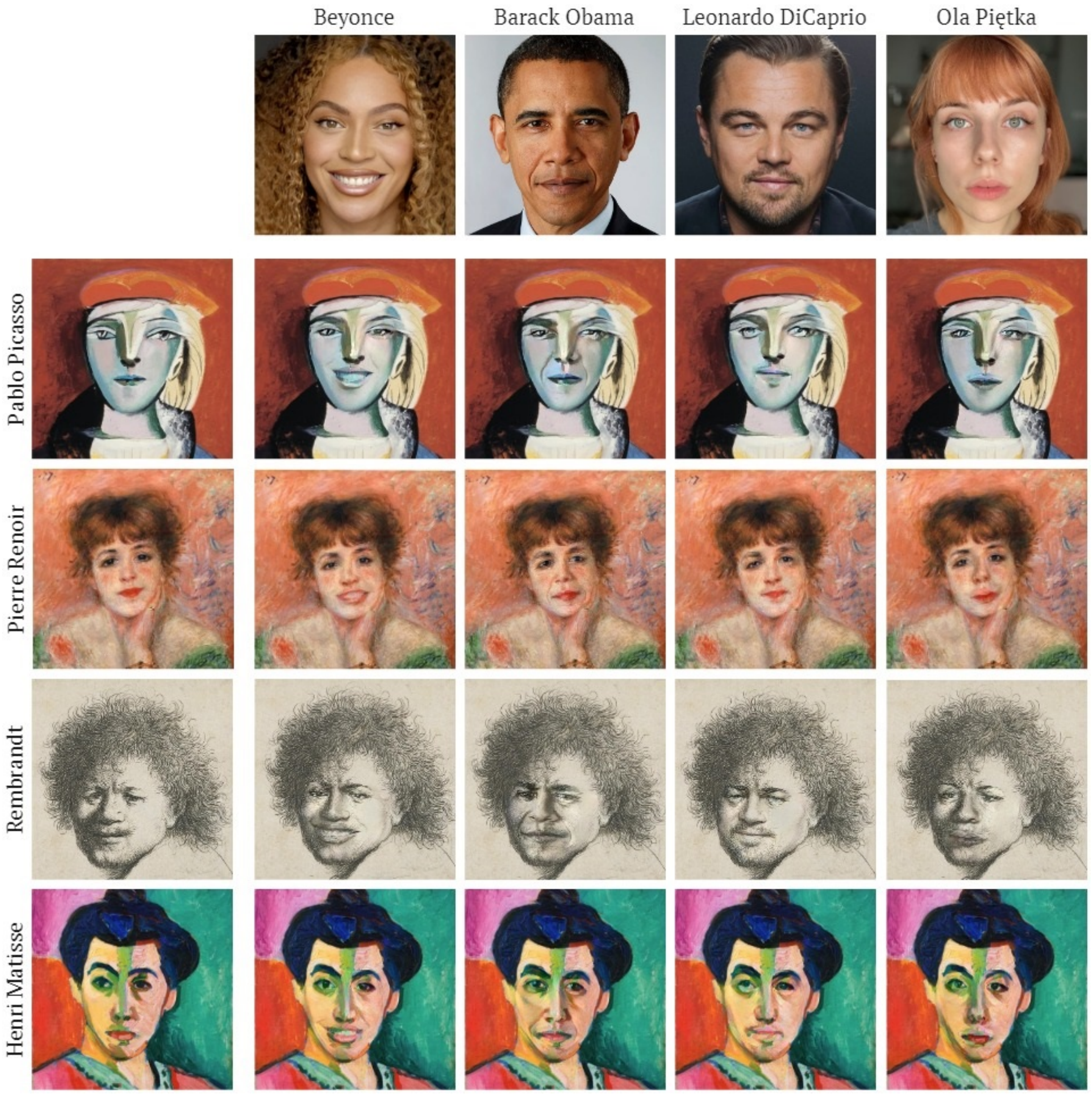


Figure 4: Final portrait face swap results with style transfer. Original portrait (style) images are in the first column, and original face (content) images are in the first row.

Finding the optimal combination for the loss function was a complex task that required thorough research and studying various scientific publications. The process involved much trial and error, eventually leading to a unique approach for style transfer that had not been explored before. The division of the process into two distinct stages notably enhances the overall quality of the produced image and boosts the effectiveness of the style transfer (Figure 5).

The sophisticated process of swapping faces in portraits has produced remarkable results across a wide variety of examples (Figure 4). The style transfer technique successfully maintains the original style of the painting, resulting in a convincing face swap. It effectively retains key features such as smiles or beards from the content image in the output, while also maintaining the style of the portrait. Even when dealing with black and white portraits, colors are transferred accurately.

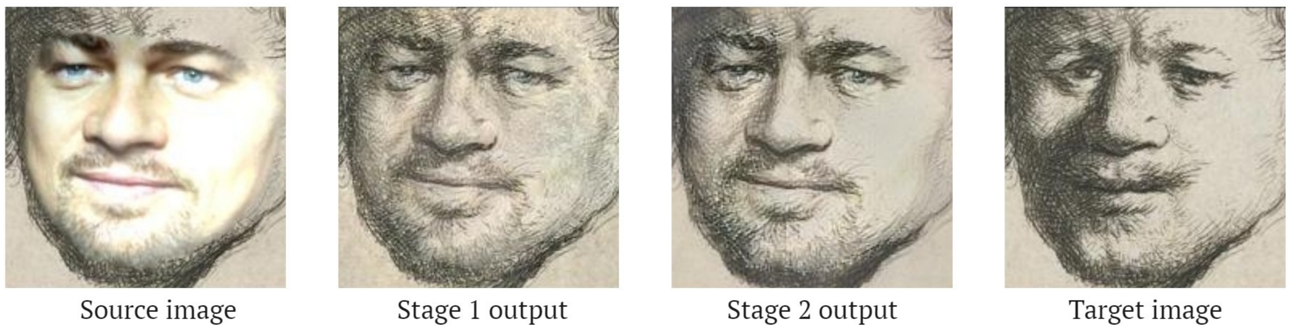


Figure 5: Comparison of outputs from stage 1 and 2 w.r.t. source and target images.

4 Details on Implementation

For style transfer, I utilized a pre-trained VGG19 network (Simonyan and Zisserman, 2015) on the ImageNet dataset to extract features from intermediate layers. The loss function was then minimized using the Adam optimizer (Kingma and Ba, 2017) on a GPU V100.

Content loss was computed at the 'relu' layers *conv1_1*, *conv2_1*, *conv3_1*, *conv4_1*, and *conv5_1* of the original VGG19 network, using weightings of 0.4, 0.2, 0.1, 0.2, and 0.1 respectively. I set the alpha parameter, which serves as the weight term, to 5e+1 for both stages.

For style loss, I computed it at 'relu' layers *conv3_1* and *conv4_1* of the original VGG19 network with weightings of 0.4 and 0.6 respectively. The beta parameter, another weight term, was set to 1e+4 for both stages.

The histogram loss was calculated at 'relu' layers *conv1_1* and *conv4_1* from the original VGG19 network, both weighted at 0.5. The theta parameter, or weight term, was set to 2e+3 for the second stage (set to 0 for first stage).

For the computation of total variation loss, I set the weight gamma to 5e-2 for the second stage (set to 0 for first stage).

In the first stage of the process, I set the number of iterations to 250 (per pyramid level) and the learning rate to 2.0. In the second stage, the number of iterations was increased to 500 and the learning rate was decreased to 1.0.

For the input portraits, I used images from an open-source art gallery WikiArt.org. For test cases, I chose photos of well-known figures: Beyonce, Barack Obama, Jennifer Lawrence, Lady Gaga, and Leonardo DiCaprio.

5 Future Improvements

Moving forward, I plan to incorporate a deep face recognition approach to improve the precision of facial feature mapping. Certain unconventional portraits, such as those by Pablo Picasso, currently present challenges due to difficulty in recognizing facial features. One possible solution could be to develop or employ a pre-trained model specifically for detecting keypoints.

The time-intensive nature of the current process is another aspect that needs optimization. Presently, the entire operation takes approximately twenty minutes on a GPU, which isn't suitable for real-time applications. Hence, streamlining the computational process to reduce overall processing time will be an essential goal.

Finally, a noticeable limitation of the existing approach is the potential for exaggerated distortions when the face image's resolution is much lower than that of the portrait. In the future, this issue could be mitigated by introducing more robust pre-processing steps or implementing super-resolution techniques to enhance image quality.

6 Conclusion

In conclusion, I am delighted to have undertaken such a fascinating project that combined my love for art, technology, and computational photography. The project shows how computational photography can transform our understanding and interaction with art. It was a challenging yet rewarding journey of discovery, filled with learning and problem-solving. I'm excited about the future, where art and technology can synergize to open even more avenues of creativity.

7 Acknowledgments

This project is built upon the final group project created for the graduation class ”Computational Photography”. Special thanks to Brayden Bingham and Brian Reinbold for their significant contributions to the Face Swap (Section 2.1) process.

References

P. Burt and E. Adelson. 1983. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2015. A neural algorithm of artistic style.

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

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Eric Risser, Pierre Wilmot, and Connelly Barnes. 2017. Stable and controllable neural texture synthesis and style transfer using histogram losses.

Ahmed Selim, Mohamed Elgharib, and Linda Doyle. 2016. Painting style transfer for head portraits using convolutional neural networks. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, pages 129:1–129:18.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition.

WikiArt.org. Wikiart -visual art encyclopedia.