University of Liège - Faculty of Applied Sciences

Master Thesis

Style Transfer on Face Portraits

Graduation Studies conducted for obtaining the Master's degree in Computer Science and Engineering

Author:
Christophe André

Supervisors:
Pierre Geurts
Matthia Sabatelli

Academic year 2018-2019

Abstract

The topic of style transfer and, more specifically, style transfer on head portraits is tackled in this thesis, primarily through the lens of *optimization-based*, *neural* style transfer algorithms. Several techniques from the recent literature are examined and their advantages as well as shortcomings are discussed – both individually and in relation to one another – before elaborating a solution combining the strengths of these various algorithms.

"A neural algorithm of artistic style" [6] is first considered, laying the foundation for most algorithms included in this work with its neural style transfer framework, which is shown to be rather modular in subsequent chapters.

After mixed results are observed with this first algorithm, three localized style transfer techniques, namely *"Combining markov random fields and convolutional neural networks for image synthesis."* [11], *"Semantic style transfer and turning two-bit doodles into fine artworks* [3] and *"Visual Attribute Transfer through Deep Image Analogy"* [13] are reviewed, and prove be a better fit for the task, thanks to their property of performing the style transfer in a context sensitive manner.

Where preceding methods are general, the specialized setting considered in this thesis is addressed with a discussion of the *"Painting style transfer for head portraits using convolutional neural networks"* [19] paper.

Lastly, a solution to the problem is proposed, combining the *Painting style transfer for head portraits* with the *"Deep Painterly Harmonization"* [14] method, which itself incorporates concepts from the methods explained in chapters 2 and 3. Results obtained with this last method are presented for a few different examples; they compare favorably to those resulting from the base *Deep Painterly Harmonization* technique when dealing with face portraits.

The solution's primary limitations are identified in the form of a high memory consumption and significant execution time. Pointers to deal with these issues in future works are also provided.

Acknowledgements

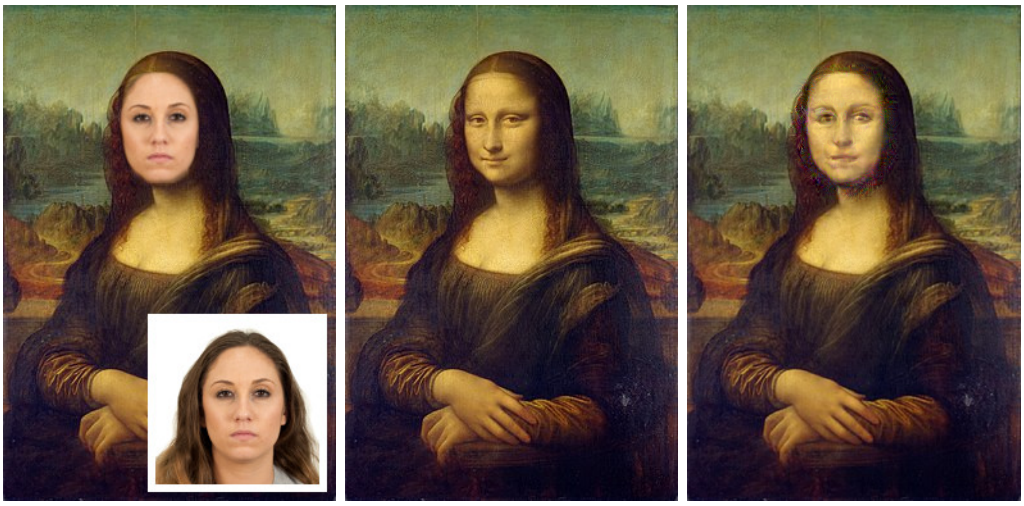
I would like to personally thank my promoters Prof. Pierre Geurts and Matthia Sabatelli for their constant dedication to this thesis. I am especially grateful for their highly valuable feedback as well as their willingness to regularly organize and participate in meetings over the course of several months.

Contents

Chapter 1

Introduction

Style transfer – the process of creating new artworks from two images by applying the *style* of one to the other – while staying faithful to the *content* of this last image – is an increasingly popular research topic, due in no small part to the publication in 2015 of *An algorithm of neural style* by Gatys et al., which provides both a representation of style and exploits the capabilities of classifiers trained on ImageNet to perform a convincing style transfer with an optimization based technique.



(a) content (b) style portrait (c) result

Figure (1.1) Example of neural style transfer applied on face portraits. Style image: *Mona Lisa*, Leonardo Da Vinci. Face appearing in the content image (a) sampled from the Chicago Face Database [15]. (c): output of the style transfer technique proposed in this thesis (and described in chapter 5) for the example at hand.

In this thesis, we are more interested in a specialized form of style transfer: style transfer on face portraits. The idea is to insert the face of an individual in a stylized portrait, the challenge residing in the need to both preserve the subject's likeness while replicating the style of the portrait, to create a believable substitute.

Figure 1.1 exemplifies the process, depicting both the inputs and expected output. It was generated from the solution proposed at the end of this thesis, in chapter 5.

Multiple neural style transfer methods are examined throughout this thesis, for which pros, cons and ideal use cases are discussed. The majority of these techniques is a variation on the optimization based neural style transfer framework first presented in chapter 2, making comparison between these methods possible. Eventually, aspects of these algorithms are combined to produce outputs like that of Figure 1.1.

In chapter 3, three different localized style transfer methods are discussed, two of which are implemented: *"Combining markov random fields and convolutional neural networks for image synthesis."* [11], *"Semantic style transfer and turning two-bit doodles into fine artworks* [3] and *"Visual Attribute Transfer through Deep Image Analogy"* [13]. These techniques contrast with the algorithm by Gatys et al. presented in chapter 2, which relies on globally matching the style of the artwork and generated image.

An extension of neural style transfer specific to the special case of head portraits is presented and implemented in chapter 4, referencing the paper *"Painting style transfer for head portraits using convolutional neural networks"* [19].

Chapter 5 sees the introduction of the *"Deep painterly harmonization"* technique [14] which enables the option of transferring style only on part of the content image, making the result in Figure 1.1 possible. This technique can be seen as a combination of the best features of techniques explained in chapters 2 and 3. Finally, this algorithm is modified to improve its performance when dealing with face portraits, making use of the knowledge gained in chapter 4 to increase the quality of visuals in this specific instance.

Limitations and perspectives for future work are also discussed at the end of the document highlighting the constraints related to memory management and execution time and briefly discussing potential solutions.

Chapter 2

Neural Style Transfer

2.1 A neural algorithm of artistic style

The first algorithm to perform style transfer using convolutional neural networks was presented by Gatys et al. [6] in 2015 . Built on top of the VGG-16 network, trained on ImageNet for image classification, this technique is an optimization problem aiming to apply a certain style to an image while reconstructing its content from intermediate feature maps of the VGG network.

The core idea consists in optimizing a custom loss function by gradient descent, with this loss being computed based on features extracted from intermediate layers of the VGG network, which is fed a content image, a style image and finally the image to optimize (output image), which is initially random noise. This is illustrated by figure 2.1.

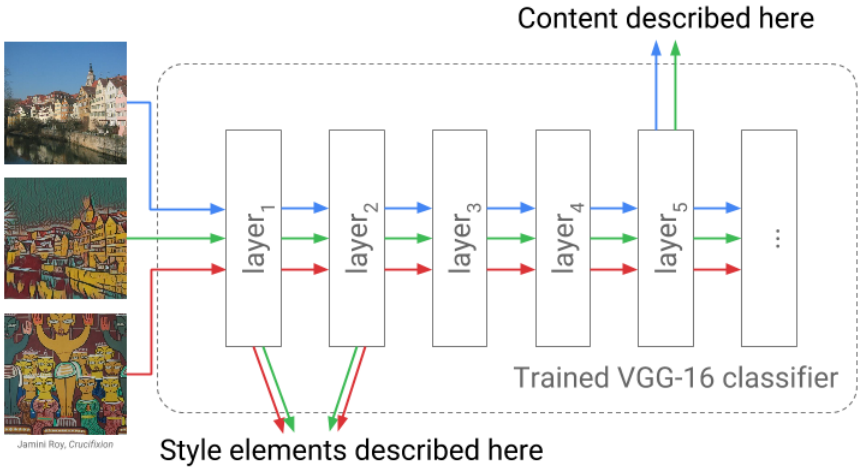


Figure (2.1) Neural style transfer: extraction of content and style features. Image originating from [2]

A main finding of the paper is that content and style appear to be separable, to a certain extent at least, allowing the generation of an image by jointly optimizing with respect to the content and style information. This is done using a total loss function which

is a weighted sum of a content loss and style loss. Weights allow to decide the importance of the content with respect to the style in the resulting image:

$$\mathcal{L} = \alpha\mathcal{L}_c + \beta\mathcal{L}_s$$

2.1.1 Content Loss

The content loss is defined as the mean square error between feature maps of the content image and output image at a given layer (typically a relatively deep layer in the network, as these tend to better encode the content information):

$$\mathcal{L}_c = \frac{1}{2}\sum_{i,j}(F_{ij}^{out} - F_{ij}'^{content})^2$$

2.1.2 Gram Matrix

In addition to the content loss, which is concerned with measuring the difference between neural activations of the generated and reference image in order to preserve semantics, the authors introduce another loss function, this time representing the discrepancy between the style of the new image and of the style image provided as input.

The notion of style is rather difficult to accurately define, especially with regards to the content, which could for instance be represented visually by applying a Laplacian edge detector filter. Style can however be thought of as a combination of color, shapes and texture information. In the paper, the *Gram matrix* is proposed as a way to represent style. This matrix G is defined by

$$G_{ij}^l = \sum_k^{H^lW^l} F_{ik}^l F_{jk}^l \mid i \in \{1, ..., C^l\},\ j \in \{1, ..., C^l\}$$

where H^l,W^l are respectively the height and width of the feature maps and C^l is the associated number of channels (or filters) for the particular layer l. Figure 2.2 illustrates the formation of the Gram matrix is for a feature map composed of four filters (in practice and for the VGG network, the number of filters can be of 64, 128, 256 or 512 depending on the layer at hand).

The gram matrix, based on the definition, encodes for each pair (i, j) of filters, a measure of the correlation between activations for this couple of channels. This is achieved by flattening these two filters into vectors of length HW and taking their dot product, as shown on figure 2.2. Additionally, the Gram matrix discards spatial information. For instance, shuffling pixels in the feature map will result in the same feature map. The Gram matrix thus encodes global statistics about the image and does not take content into account which makes it a good candidate as a representation of style.

2.1.3 Style Loss

The style loss is then defined as the difference between *Gram matrices* of feature maps of the style and input image at the layers selected to extract the style features:

$$\mathcal{L}_s = \sum_l \frac{1}{(2\ C^l H^l W^l)^2} \sum_{i,j} (G_{ij}^{out,l} - G'_{ij}{}^{style,l})^2$$

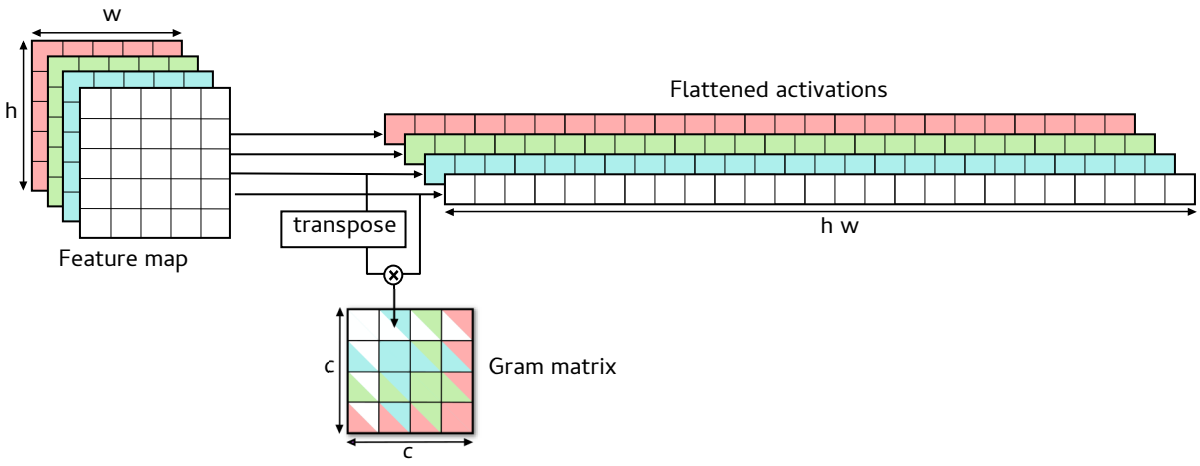


Figure (2.2) Computation of the Gram matrix corresponding to a feature map. Each element of the resulting matrix is a measure of the correlation (due to the use of a dot product) between the activations of two filters (channels). c is the number of channels, while h and w are the vertical and horizontal dimensions of the feature map.

2.1.4 Optimization procedure

A formalization of the neural style transfer algorithm is provided in the form of procedure 1. For the sake of clarity and simplicity, some operations are represented in a sub-optimal fashion: for instance, the network is parsed once to extract features, and these features are subsequently parsed again to compute a loss, which is a different behavior from the actual implementation where the computation of losses is integrated in the forward pass. As shown in procedure 1, the technique is mainly an optimization problem, where a loss is computed at each iteration and optimized trough gradient descent, each iteration ending with a backpropagation step.

Procedure 1 defines a framework which will be reused, with some adjustments both minor and significant, in the following chapters. These modifications are for the most part related to the objective functions. In this chapter, the content and style losses used are those which were defined by Gatys et al. and described earlier. They are detailed in procedures 2 and 3 respectively.

2.1.5 Choice of style layers

The quality of the results is affected by the choice of layers for which are taken into account when calculating the style loss, as layers at different depths in the architecture lead to different style reconstructions[1] as shown by Gatys et al [**?**]. Therefore, the influence

[1] *i.e.* the images obtained by, starting from random noise, minimizing the style loss only

Procedure 1 Neural Style Transfer as an optimization problem

parameters: The reference images $I^{content}$ and I^{style}; $content\_layers$, $style\_layers$ the layers used to compute the content and style losses
return: I^{out}, the generated image
function NEURAL-STYLE-TRANSFER($I^{content}$, I^{style}, $content\_layers$, $style\_layers$)
    ▷ preprocessing
    $content\_feats \leftarrow$ ExtractFeatures(VGG-19, $I^{content}$)
    $style\_feats \leftarrow$ ExtractFeatures(VGG-19, I^{style})
    ▷ initialize the new image to the content; white noise can alternatively be used
    $I^{out} \leftarrow$ **copy** $I^{content}$
    ▷ optimization by gradient descent
    for $i \leftarrow 1$ to n_iter **do**
        $out\_feats \leftarrow$ ExtractFeatures(VGG-19, I^{out})
        $L_{content} \leftarrow 0$
        $L_{style} \leftarrow 0$
        for $l \in \{content\_layers \cup style\_layers\}$ **do**
            if $l \in content\_layers$ **then**
                $L_{content} \leftarrow L_{content} +$ ContentLoss($feats[l]$, $content\_feats[l]$)
            if $l \in style\_layers$ **then**
                $L_{style} \leftarrow L_{style} +$ StyleLoss($feats[l]$, $style\_feats[l]$)
            $L_{total} \leftarrow \frac{\alpha L_{content}}{\text{Length}(content\_layers)} + \frac{\beta L_{style}}{\text{Length}(content\_layers)}$
            ▷ encourage consistency between neighbouring pixel values
            ▷ so as to reduce noise (optional)
            $L_{total} \leftarrow L_{total} + \gamma$TotalVariationDenoisingLoss(I^{out})
            Backpropagate(L_{total}, L-BFGS)
    return I^{out}

of each style layer on the final output is studied. Results are shown on figure 2.3 for a particular content/style pair, after 300 iterations. The generated image is initialized with the content image.

Considering a single layer to represent style leads to mixed results: on one hand layers `relu3-1`, `relu4-1` and `relu5-1` result in a texture resembling that of the original style portrait, although with different levels of granularity, but don't accurately reflect the color distribution of the reference style image. On the other hand, the colors are better preserved for layer `relu2-1`, at the cost of texture.

Combining layers leads to a better outcome, as best seen on figure 2.3.(g) and 2.3.(i). This requires a slightly higher memory consumption, since more gram matrices need to be memorized but both texture and color information are preserved (at least partially). Apart from the comparison for different layers, these images highlight an issue with the proposed algorithm: semantics are ignored when applying the style to the image. Indeed, the style is applied globally, which results in inconsistencies. For instance, hair should appear gray as in the reference style portrait but is instead composed of different colors sampled from

Procedure 2 Content Loss

parameters: F^{out} and $F^{content}$, the activations of the image being generated and content image at the current layer in the network
require: Shape(F^{out}) = Shape($F^{content}$)
return: The contribution to the content loss at the current layer, which is the mean squared error of the activations
function CONTENT-LOSS(F^{out}, $F^{content}$)
    H, W, C = Shape($F^{content}$) ▷ height, width, number of channels
    $L_c \leftarrow 0$
    for $i \leftarrow 1$ to $H \cdot W \cdot C$ **do**
        $L_c \leftarrow L_c + (F_i^{out} - F_i^{style})^2$
    return $\frac{L_c}{2 \cdot H \cdot W \cdot C}$

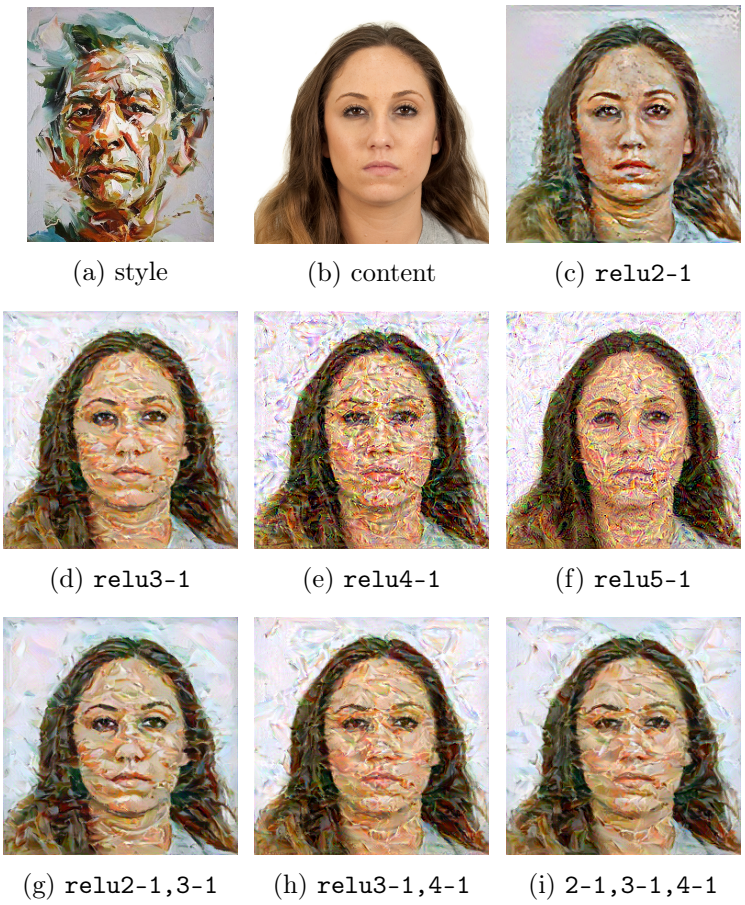
Procedure 3 Style Loss

parameters: F^{out} and F^{style}, the activations of the image being generated and style image at the current layer in the network
require: Shape(F^{out}) = Shape(F^{style})
return: The contribution to the style loss at the current layer (the mean squared error of the Gram matrices, normalized according to the feature map sizes)
function STYLE-LOSS(F^{out}, F^{style})
    $G^{out} \leftarrow$ GramMatrix(F^{out}) ▷ matrix of size CxC (see Figure 2.2)
    $G^{style} \leftarrow$ GramMatrix(F^{style})
    $L_s \leftarrow 0$
    for $i \leftarrow 1$ to C^2 **do**
        $L_s \leftarrow L_s + (G_i^{out} - G_i^{style})^2$
    return $\frac{L_s}{(2 \cdot H \cdot W \cdot C)^2}$

the whole image. This flaw is addressed in later chapters.

2.1.6 Style to Content ratio

The content to style ratio is another parameter to consider when using neural style transfer. As style and content representations are not entirely disentangled, one needs to decide the relative importance to give to content and style for each pair of reference images. Results of style transfer with different orders of magnitude for the style to content ratio are depicted on figure 2.4. As expected, the greater the ratio, the closer the style to the style reconstruction (obtained by ignoring the content loss), shown on image 2.4.(c). Once again, a flaw of the technique is exposed: while colors appear correctly transferred on a global level, there is no incentive to match colors of semantically similar elements (here skin or the background)



(a) style (b) content (c) `relu2-1`

(d) `relu3-1` (e) `relu4-1` (f) `relu5-1`

(g) `relu2-1,3-1` (h) `relu3-1,4-1` (i) `2-1,3-1,4-1`

Figure (2.3) Neural style transfer with different style layers. For (c),(d),(e),(f) the style loss is computed based on a single layer while (g), (h) and (i) combine different layers.

2.1.7 Total Variation Denoising Loss

Making an appearance in Procedure 1, the total variation denoising loss, though absent from the original algorithm by Gatys et al. is part of most recent implementations of neural style transfer. This loss was used in the paper *"Perceptual Losses for Real-Time Style Transfer and Super-Resolution"* [10] by Johnson et al. and has since been used as a regularizer, limiting the variance in pixel values for the output image. This is generally needed as, with an increasing number of iterations, noise tends to appear in the new images as the algorithm tries to minimize the content and style loss, which can lead to unnatural images. The loss can be described by the formula

$$L_{tv} = \frac{\sum_{i=1}^{h}\sum_{j=1}^{w} |I_{i,j} - I_{i+1,j}| + |I_{i,j} - I_{i,j+1}|}{hw}$$

where I is an image of dimensions (h, w). Minimizing this loss encourages a smooth output, as adjacent pixels with similar color values yield a small loss. The degree of

(a) content (b) style (c) style reconstruction

(d) $\beta/\alpha = 1$ (e) $\beta/\alpha = 10$ (f) $\beta/\alpha = 100$

Figure (2.4) Impact of the style/content ratio (β/α). Optimization is performed for 300 iterations, the content loss is computed at layer `relu4-1` and the style loss at layers `relu2-1` and `relu3-1`. (c) corresponds to $(\beta/\alpha) = \infty$.

smoothing is in practice controlled using a weight, similarly to the content and style weights prefixing their respective losses.

2.2 Interpreting neural style transfer

The algorithm presented by Gatys et al. is an innovative solution, allowing the synthesis of an image matching both a certain style and content at once. It is however still hard to provide a convincing interpretation as to why this method actually manages to capture information about the style of an image.

The authors consider the formation of the Gram matrix from features of the VGG network, which encodes correlations between different neurons and suggest that this operation of establishing correlations between neurons could be implemented in some form by the human visual system to obtain a representation of style mostly independent from the content. But this is only speculation and is not an entirely satisfying justification for the use of the Gram matrix.

2.2.1 Neural style transfer and domain adaptation

Further insight is provided in the paper *"Demystifying Neural Style Transfer"* (Li et al., 2017) [12], whose main contribution is the re-contextualization of neural style transfer as a domain adaptation problem[2]. To arrive at this conclusion, Li et al. prove that minimizing the style loss (or, more specifically, the term contributing to it at a particular layer) as previously defined is equivalent to minimizing the *Mean Maximum Discrepancy* (MMD) [7] between the feature maps extracted from the generated image and the style image, the MMD being a popular measure of the difference between two distributions. The aim of neural style transfer is thus to align the distributions of features extracted from the generated and style image and matching the Gram matrices of activations is a way to pursue this goal.

Mathematically (only considering a single layer at once for simplicity) the style loss is re-arranged as follows, where some intermediate steps are omitted for brevity:

$$
\begin{aligned}
\mathcal{L}_s &= \frac{1}{4C^2H^2W^2}\sum_{i=1}^{C}\sum_{j=1}^{C}(\sum_{k=1}^{HW}F_{ik}^{out}F_{jk}^{out} - \sum_{k=1}^{HW}F_{ik}^{style}F_{jk}^{style})^2 \\
&= \frac{1}{4C^2H^2W^2}\sum_{i=1}^{C}\sum_{j=1}^{C}((\sum_{k=1}^{HW}F_{ik}^{out}F_{jk}^{out})^2 + (\sum_{k=1}^{HW}F_{ik}^{style}F_{jk}^{style})^2 - 2(\sum_{k=1}^{HW}F_{ik}^{out}F_{jk}^{out})(\sum_{k=1}^{HW}F_{ik}^{style}F_{jk}^{style})) \\
&= [...] \\
&= \frac{1}{4C^2H^2W^2}\sum_{i=1}^{HW}\sum_{j=1}^{HW}((\sum_{k=1}^{C}F_{ki}^{out}F_{kj}^{out})^2 + (\sum_{k=1}^{C}F_{ki}^{style}F_{kj}^{style})^2 - 2(\sum_{k=1}^{C}F_{ki}^{out}F_{kj}^{style})(\sum_{k=1}^{C}F_{ki}^{out}F_{kj}^{style}))
\end{aligned}
$$

By introducing f_i as the vector of activations for the i^{th} pixel in feature map F and the kernel $K(x, y) = (x \cdot y)^2$ the style loss can be re-written to establish the link with the maximum mean discrepancy:

$$
\begin{aligned}
\mathcal{L}_s &= \frac{1}{4C^2}\left(\frac{1}{H^2W^2}\sum_{i=1}^{HW}\sum_{j=1}^{HW}(K(f_i^{out}, f_j^{out}) + K(f_i^{style}, f_j^{style}) - 2K(f_i^{out}, f_j^{style}))\right) \\
&= \frac{1}{4C^2}\text{MMD}(F^{out}, F^{style})^2
\end{aligned}
$$

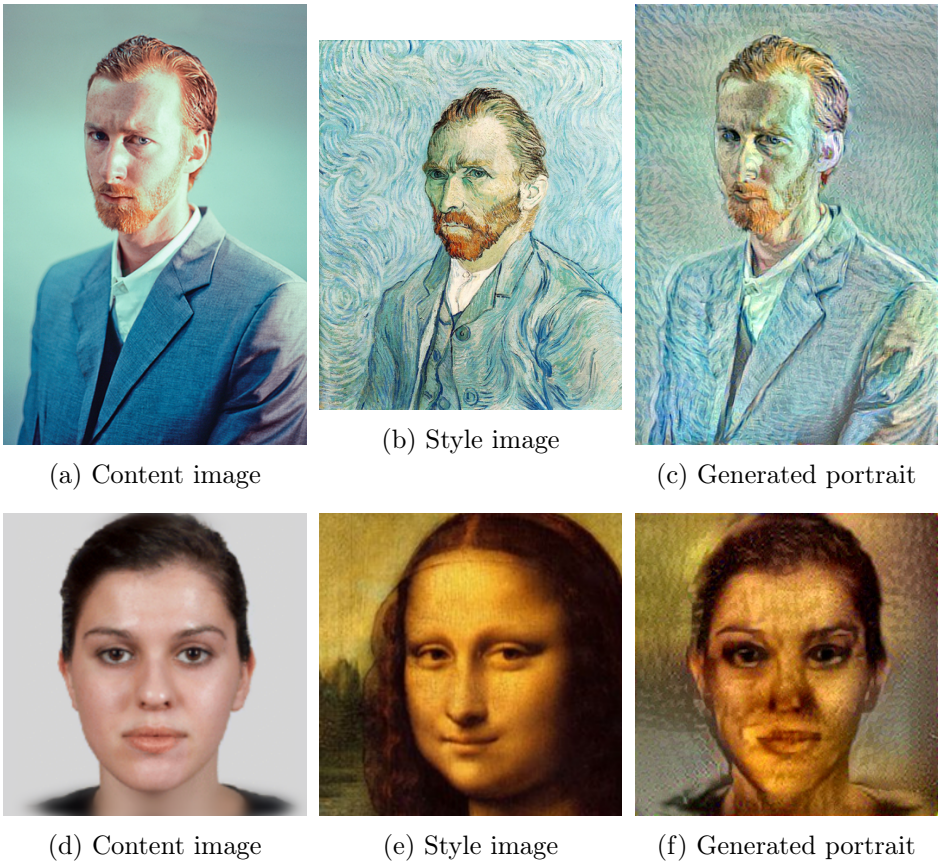
As noted by the authors, this case of domain adaptation is noteworthy as the samples $\{f_i^{out}, f_i^{style} | i \in 1, ..., HW\}$ constituting the two distributions to be matched (F^{out} and F^{style}) describe individual pixels instead of a complete image as is usually the case. This

[2]Domain adaptation is concerned with learning a model on a *source distribution* with the aim to achieve good performance on a given *target distribution*. An example of source distribution is a catalogue of product images photographed in a controlled environment (for instance with a specific orientation, on a white background), while an associated target distribution could be a set of pictures of similar objects taken in a real setting.

last formulation is also interesting as it outlines the fact that the position of each pixel is irrelevant when computing the style loss. This justifies the joint optimization with respect to the content and style, since the measure of style discards spatial information.

Finally, the formulation of the Mean Maximum Discrepancy, which involves a kernel, suggests the possibility of alternatives to the Gram matrix as a means of representing style. Indeed, the authors show the kernel used above ($K(x, y) = (x \cdot y)^2$) can be substituted, for example, by a polynomial kernel $K(x, y) = (x \cdot y + c)^d$ (of which the quadratic kernel is a particular case), including the linear kernel or even a gaussian kernel. These all lead to slight variations in the reproduced style but essentially perform the same role as the Gram matrix.

2.3 Results



(a) Content image (b) Style image (c) Generated portrait

(d) Content image (e) Style image (f) Generated portrait

Figure (2.5) Style transfer using the gram-matrix based optimization method developed by Gatys et al. (a) Photo by Seth Johnson. (b) Vincent Van Gogh, self-portrait. (c) Face portrait sampled from the MR2 Face database [22]. (d) *Mona Lisa*, Leonardo da Vinci.

While not real-time capable, the proposed neural style transfer method is still relevant, making it a good starting point for the style transfer of face portraits. Generated images

with this algorithm are shown on figure 2.5. While portrait (c) is rather satisfying overall, the eyes for instance can look unnatural and seem to resemble the buttons on the coat in the style image. The second portrait is more problematic as while the whole picture is painted in a style similar to the reference image, the face in particular looks quite rough and doesn't match the more refined aesthetic of the one present in the reference style portrait.

Chapter 3

Local Style Transfer

In their review of neural style transfer [9], Jing et al. divide optimization based neural style transfer methods into two broad categories: parametric techniques with *Summary Statistics* and non-parametric methods with *Markov Random Fields* (MRFs). This classification is based on the approach used to model texture and, by extension, style.

A prime example of the first category is the algorithm of neural style presented by Gatys et al. [**?**], where global statistics are extracted from feature activations to describe a texture. A notable shortcoming of this approach is that information about spatial arrangements is discarded when summarizing the style image with in this particular case, the *Gram matrix*.

Not taking spatial configuration into account when computing the style loss is especially problematic when considering head portraits, where deformations are quickly noticed and can ruin the illusion of style transfer.

The second category outlined by Jing et al., differs from the first in that it relies on the assumption that the color distribution of each pixel is only dependent of pixels in its immediate neighbourhood. This non-parametric approach is examined in this chapter, mainly through the lens of the paper *"Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis"* [11] by Li and Wand (2016) who were the first to propose an algorithm based on non-parametric texture re-sampling operating on features extracted by a neural network, rather than on the base images.

3.1 Visual Texture Modelling with MRFs

In *"Texture Synthesis by Non-parametric Sampling"* [5], Efros et al. tackle the problem of texture synthesis, given a sample texture as input by modelling texture using a Markov Random Field model. The purpose of this model is to represent dependencies between the color values of each pixel and the color values of all other pixels in the sample image.

Given the assumption made that a pixel is only dependent from neighbouring pixels, one can build a undirected graph where each pixel is connected to its neighbours[1] (and

[1] In this case, the neighbourhood is defined as a square window centered on the pixel at hand.

only its neighbors). By considering each pixel as a random variable (describing the pixel's color) and imposing a Markov property[2] to the graph, a Markov random field is obtained.

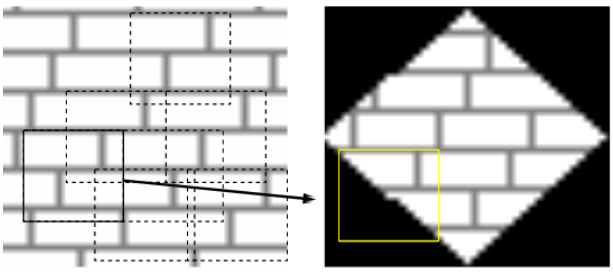


Figure (3.1) Illustration of *"Texture synthesis by Non-parametric Sampling"*. On the left: sample texture. On the right: texture being synthesized. Image taken from [5]

The texture synthesis algorithm is illustrated on Figure 3.1. Starting from a given seed, the new texture is synthesized one pixel at a time. The value of the pixel at hand is determined by matching a square window centered around the pixel with the most similar patch from the sample texture and sampling the pixel located at the center of this patch.

3.2 Patch-based Neural Style Transfer

While the texture synthesis using MRFs method presented above is certainly effective at generating texture such as presented on Figure 3.1, it is rather limited when applied on raw images for style transfer, as patches will be matched partially according to the color information contained in the images rather than only based on the content.

In *"Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis"* [11], Li and Wand greatly increase the versatility of the technique and apply it to the domain of style transfer by operating on features extracted by a neural network pre-trained on ImageNet, here VGG-19 [21]. This approach is, as such, a combination of the neural style transfer algorithm proposed by Gatys et al., which operates on features extracted from a content and style image and the texture modelling with MRFs. Where the algorithm differs with that of Gatys et al. is in the style loss, which is here based on the MRF prior and is defined as follows:

$$\mathcal{L}_s = \sum_i ||\Psi_i(F_{content}) - \Psi_{NN(i)}(F_{style})||^2$$

where Ψ denotes the set of (overlapping) patches extracted from a feature map and $\Psi_{NN(i)}(F_{style})$ represents the patch in $\Psi(F_{style})$ which the most closely matches the i^{th} patch in $\Psi(F_{content})$.

[2]This entails for instance that the value of a pixel is considered independent from the values of pixels outside its neighborhood when the neighborhood is known.

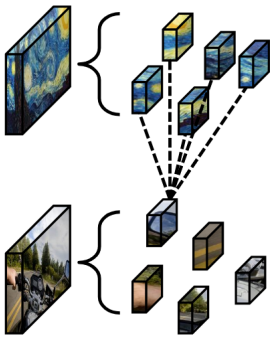


Figure (3.2) Illustration of the patch matching process. Image taken from [4]. While the mapping is here pictured for rgb images, it is in practice performed for feature maps instead. The method is however exactly the same, the only difference being the number of channels, which can be relatively large (between 128 and 512) for the feature maps and slow down the process of comparing patches, also increasing the memory requirements.

Patch matching

To perform the matching process at a given layer, all patches of size k x k x c (where c is the number of channels and k is a parameter) present in the content and style activations are first extracted. Then, for each content patch, the most similar style patch is recorded. This is shown at Figure 3.2 where, for a specific content patch, the closest style patch is selected.

3.2.1 Cosine similarity

There are multiple ways to compare patches, using for instance principal component analysis or scale-invariant feature transform. In this case, cosine similarity is used: Given two vectors $\mathbf{a}$ and $\mathbf{b}$, oriented at an angle θ, the following formula holds:

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| \ |\mathbf{b}| \ cos(\theta)$$

or, similarly:

$$cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| \ |\mathbf{b}|}$$

This cosine can be interpreted as an indicator of the similarity between vectors $\mathbf{a}$ and $\mathbf{b}$. Indeed, for parallel vectors, $cos(\theta) = 1$ while for orthogonal vectors, $cos(\theta) = 0$ with the measure varying in $]0, 1[$ for angles in $]0, \pi[$. Using this concept of cosine similarity, the index of best matching style patch is given by

$$NN(i) = \underset{j}{argmax} \ \frac{\Psi_i(F_{content}) \cdot \Psi_j(F_{style})}{|\Psi_i(F_{content})| \ |\Psi_j(F_{style})|}$$

where patches Ψ_x were flattened so as to represent vectors.

3.2.2 Style loss

Based on the matching process explained above, complete with the definition of cosine similarity, the algorithm used to compute the style loss (or, more precisely, the contribution to the style loss at a layer in the network) is detailed below, in Procedure 4. The content loss is unchanged compared to the classic neural style transfer algorithm.

Procedure 4 Local Style Transfer with Markov Random Fields - Style Loss Computation

parameters: F^O and F^S, feature maps extracted for the generated and style reference image at a given layer
require: Shape(F^O) = Shape(F^S)
return: The contribution to the style loss at the current layer.

function STYLE-LOSS-MRF(F^O, F^S, k)
    ▷ Get patches of size kxkxC centered around each pixel in the feature maps
    $\Psi^O \leftarrow$ ExtractPatches(F^O, k)
    $\Psi^S \leftarrow$ ExtractPatches(F^S, k)

    $\Psi^S \leftarrow$ NearestNeighbors(Ψ^O, Ψ^S) ▷ Mean Squared Error
    for $i \leftarrow 1$ to $n_{patches}$ **do**
        $L_s = L_s + (\Psi^O_i - \Psi^S_i)^2$
    return $\dfrac{L_s}{n_{patches} \cdot k^2}$

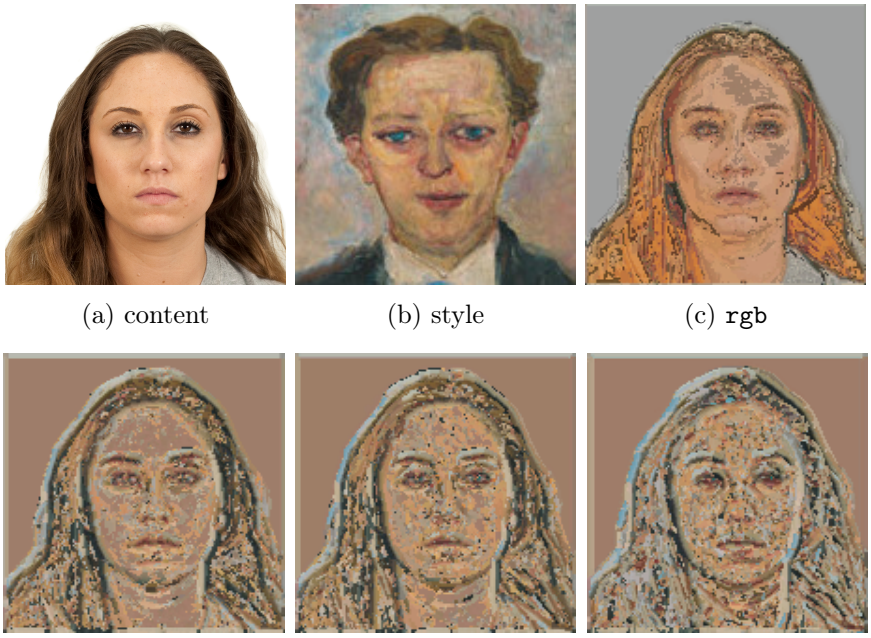
parameters: Ψ^O and Ψ^S, two lists of patches (arrays of pixel values)
require: Shape(Ψ^O) = Shape(Ψ^S)
return: Ψ, a list of patches sampled from Ψ^S s.t. Ψ_i is the nearest neighbor of patch Ψ^O_i in Ψ^S, $\forall i \in 1, ..., n_{patches}$

function NEAREST-NEIGHBOR(Ψ^O, Ψ^S)
    $\Psi \leftarrow$ **copy** Ψ^S
    for $i \leftarrow 1$ to $n_{patches}$ **do**
        $score \leftarrow$ CosineSimilarity(Ψ^O_i, Ψ^S)
        $j \leftarrow$ **arg max** $score$
        $\Psi_i \leftarrow \Psi^S_j$
    return Ψ

3.2.3 Patch matching at different layers

To better visualize the importance of computing the style loss based on MRFs from the activations extracted at intermediate layers of the VGG-19 network instead of directly using the raw style and output image, modified versions of the original style image are presented on Figure 3.3, obtained when the matching process is performed on the raw images or for layers `relu2-1`, `relu3-1`, `relu4-1` of the VGG-19 network. These modified images are obtained by sampling from the original style image: for each pixel of the content image, the kxk patch centered on it is considered and the best match from the style image

is computed. The pixel value is then sampled from the center of the matching patch from the style image.



(a) content (b) style (c) `rgb`

(d) `relu2-1` (e) `relu3-1` (f) `relu4-1`

Figure (3.3) Sampling from the style image when performing the matching directly (c) on the raw content (a) and style (b) images or at intermediate layers of the VGG network (d), (e), (f). The style image is part of the following painting: Oskar Kokoschka, Der Trance-spieler. Oil on canvas, (1909). Content image (a) sampled from the Chicago Face Database [15].

The method used to build these images, by sampling from the style image, is not actually indicative of the full algorithm since in practice, the re-mapped versions of the style image only serve to compute the style loss, with the final output being a reconstruction based on the content and style loss. The results from Figure 3.3 are however fit to demonstrate the advantages of neural patch-based style transfer with respect to the classic method [5]. It can indeed be observed that when the matching phase is done on feature activations, pixels are sampled in such a way that content is taken into account. For instance, when considering layer `relu4-1`, pixels covering the eyes are mostly sampled from the region covering the eyes in the style image, even though there are significant color differences. This does not appear to be the case when working directly on the raw color images, where patches seem to be matched with an emphasis on preserving color. When using feature maps, there appears to be a decoupling with color information, as the plain white background is replaced by a brown flat texture, which was likely selected due to the presence of such a plain patch in the style image when a white or light grey patch could have been selected.

Comparison of shallow and deep layers

In the VGG network, when considering an image of resolution (x, y), the feature maps are of size $(\frac{x}{2}, \frac{y}{2})$, $(\frac{x}{4}, \frac{y}{4})$ and $(\frac{x}{8}, \frac{y}{8})$. To counter this down-scaling and enable a relevant comparison, Figure 3.3 was generated by re-sizing the input images accordingly to synthesize images of resolution 256 by 256. Upon close inspection, the images generated when doing the matching at deeper layers appear crisper (see for instance the forehead, which is synthesized similarly to the background for layer 2 but more closely resembles the skin on the style image for layer 4, or the eyes which better match the style image for deeper layers). This supports the intuition that deeper layers of the VGG network are more fit to model content, with shallow layers mostly focusing on texture information.

Mapping robustness visualization at different layers

(a) content (b) style (c) `rgb`

(d) `relu2-1` (e) `relu3-1` (f) `relu4-1`

Figure (3.4) Heat maps of the cosine similarity between the patch highlighted in red on (a), partially covering the left eye, and all style patches at different layers.

The differences in matchings at different layers can also be observed by choosing a specific patch of the style image and displaying the values of the cosine similarity with respect to patches centered around every pixel of the style image. This visualization is shown at Figure 3.4, and was obtained by performing similar up-scaling operations as for Figure 3.3. Values were normalized before displaying the heat maps. The patch considered mainly covers an eye of the person portrayed on the content image. This location was chosen for symmetry reasons: ideally, style patches for both eyes should be more or less as likely. On raw images, the observed heatmap confirms that similarity values do not vary drastically across the whole image, leading to a somewhat meaningless matching. This is

not the case for feature maps, where patches covering both eyes are more similar to the content patch comparatively to the rest of the image. Differences in similarity are more pronounced for deeper layers, which corroborates previous observations though some areas consistently achieve a relatively high degree of similarity across the three layers (such as the tip of the nose, mouth and parts of the hair).

3.2.4 Multi-resolution approach

(a) content (b) style (c) intermediate results

Figure (3.5) Multi-resolution optimization, from a 64x64 image to a 512x512 image. Style portrait: *Whirlwind*, Paul Wright.

Directly optimizing an image at the desired resolution was observed to be sub optimal, as the process does not always converge within 300 iterations and the performance of the patch matching process is heavily dependent on the resolution of the feature maps: when upscaling feature maps by a factor 2, there are 4 times as many patches to consider. Since the best match must be computed for each patch, the complexity is quadratic, leading to a increase of a factor 16 in computing time. To improve performance, and additionally robustness, the image is optimized in a multi-scale fashion, as depicted on Figure 3.5. In this case, the content image is used to initialize the output to be synthesized. Then, it is down-scaled to a low resolution (here 64x64) and the optimization is carried out for 300 iterations. The output of this first pass is then up-scaled to an image of size 128x128 and serves as input of the next optimization pass. The process is repeated for resolutions 256x256 and 512x512, to finally obtain the final image. Generally only finer details are modified in the later stages of the optimization, with the first phases determining the global shape of the face to be synthesized.

For the latest pass, feature maps are down-scaled by a factor two to compute the style loss, in an effort to preserve memory, at the cost of a slight decrease in visual fidelity. This is done to preserve memory (this was necessary to run the algorithm on a Nvidia K80 graphics card with 12gb or memory).

3.2.5 Choice of style layers

The choice of layers for which the style loss is computed is important, as it was established previously (see Figure 3.4) when comparing the effectiveness of the patch matching process

for different layers. Figure 3.6 presents the images generated with the local style transfer method based on MRFs when computing the style loss at different stages of the network while keeping the same content to style ratio throughout.

(a) content (b) style (c) reconstruction

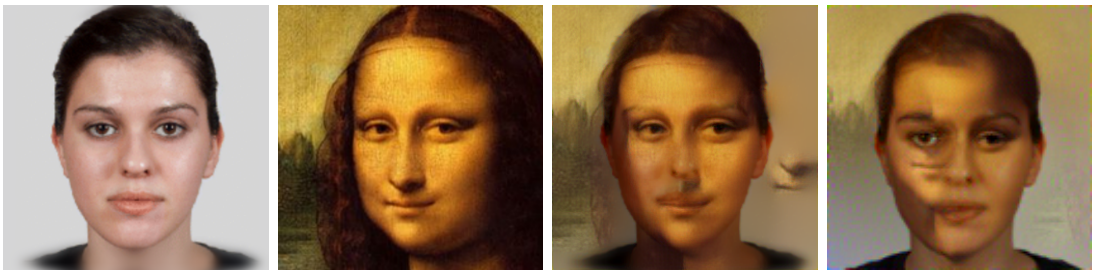
(d) `relu2-2 + 3-1` (e) `relu3-1` (f) `relu4-1`

Figure (3.6) Results with different style layers. (c) is the reconstruction of the style image (b) obtained by ignoring the content loss. (d), (e), (f) are obtained using the multi-resolution approach described earlier.

The distinction between the depicted configurations is accentuated by the abstract painting style and the difference in skin colors. This comparison tends to confirm previous observations about the patch matching process, as deeper layers appear to better preserve the facial expressions (the mouth, for instance, resembles the content image more on example (f) while it appears to match that of the style image for (d) and (e)). A downside to picking deeper layers, which was not previously apparent, is the weaker transfer of color and texture. There thus seems to exist a trade-off between the preservation of the facial expression and similarity to the painting style of the portrait.

Figures 3.5 and 3.6 share the same style portrait. In both cases, it an be observed that the generated images are slightly softer compared to the well-defined edges characteristic of the art style. I interpret this defect as being partially due to the need to downscale the feature maps because of memory limitations when computing the style loss. This phenomenon is better visualized by considering the reconstruction on Figure 3.6.(c) which can be obtained by setting the content loss to zero.

3.2.6 Artifacts



(a) content (b) style (c) result 1. (d) result 2.

Figure (3.7) *Patch-based* style transfer

Figure 3.7 illustrates two generated outputs, showcasing an issue with the local style transfer method. These were both generated using the layer `relu2-2` to compute the style loss and `relu4-1` for the content loss.

It can be observed that the generated portraits match the style image more closely than when using the Gram-matrix based loss function. A major drawback however, visible on both results, is the presence of out of place patches, such as part of Mona Lisa's hair on result 1, blending with the background, as well as part of the face reproduced on the right of the image. On result 2, what resembles a mouth appears beneath the right eye.

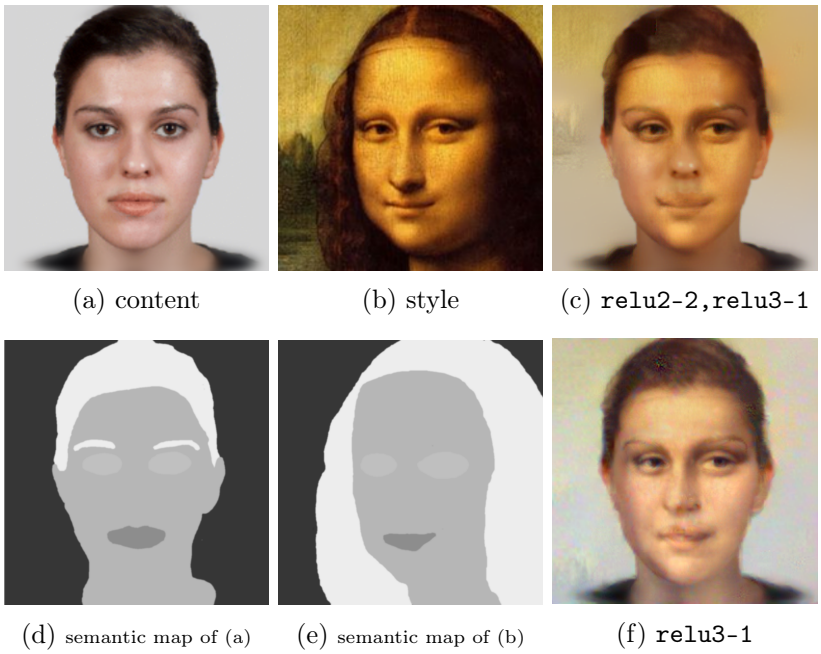
While these glitches are definitely an issue, we can still observe that fine cracks in the original painting also appear in the generated portraits upon close inspection although they are less visible in this case, which might be due to the inclusion of a *total variation denoising* term in the loss function which slightly blurs the image to counter the noise that typically appears during the optimization process.

This approach is more suited for photo-realistic style transfer due to the preservation of spatial configurations. It also works better when the content and style image follow a similar structure, which can be the case for face portraits, provided that both faces in the style portrait and content image are aligned and do not differ radically.

3.3 Semantic Style Transfer

An improvement over the previous method, *Semantic style transfer*, was presented by A. Champandard in the paper *Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artworks* [3]. The idea consists in using semantic maps to guide the patch matching process, making it more likely that a patch of hair, for instance, in the generated image will be matched with another patch of hair in the style image. An example of semantic maps for both content and style image is visible at Figure 3.8.(d,e).

In practice, these semantic maps are concatenated depth-wise with the feature maps at layers used to compute the style loss, as described on Figure 3.9. Semantic maps are multiplied by a weight, representing their importance with respect to the actual features. In my implementation, I have observed that semantic maps were useful with a weight ranging



(a) content (b) style (c) `relu2-2,relu3-1`

(d) semantic map of (a) (e) semantic map of (b) (f) `relu3-1`

Figure (3.8) Semantic style transfer. (c) and (f) were both generated with a semantic weight of 50.

from 40 to 150, with higher values starting to disrupt optimization (mainly resulting in added noise).

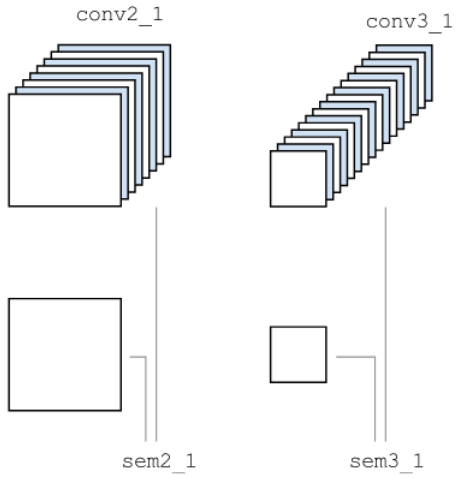
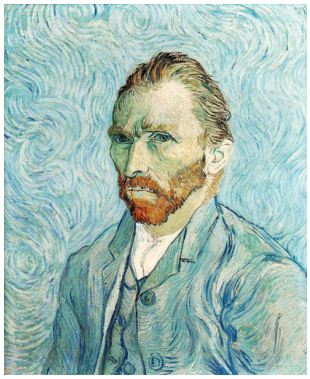


Figure (3.9) Semantic maps and feature maps are concatenated before extracting the patches. Credit: A. Champandard [3]

(a) Content image



(b) Style image

(c) Generated portrait

(d) Semantic map of (a)

(e) Semantic map of (b)

Figure (3.10) Semantic style transfer. The semantic maps were taken from [3]

Results

Results of the method can be observed on Figures 3.10 and 3.8. While content and style features were extracted at layers `relu4-1` and (`relu2-2`, `relu3-1`) of the VGG-19 network as (this generally lead to a more appealing result than other combinations of layers), Figure 3.8.(f) shows the output obtained when extracting style features from layer `relu3-1` as I found it slightly more visually consistent, despite it not reproducing the colors of the style portrait as well as the other version.

It was previously mentioned that the *patch-based* method yielded better results when the content and style portrait had a similar structure. We can also add that conversely, issues arise when the algorithm is unable to find matching patches due to significant differences in the structure of reference images. This can here be observed in parts of Figure 3.8.(c) where the tip of the nose and right cheek seem to copy the content image without applying style. A potential solution to this problem would be to extract patches from scaled

and/or slightly rotated versions of the style image, thus performing a kind of data augmentation. This has however not been tested and could suffer from a major performance cost due to the need to consider more patches during the matching process.

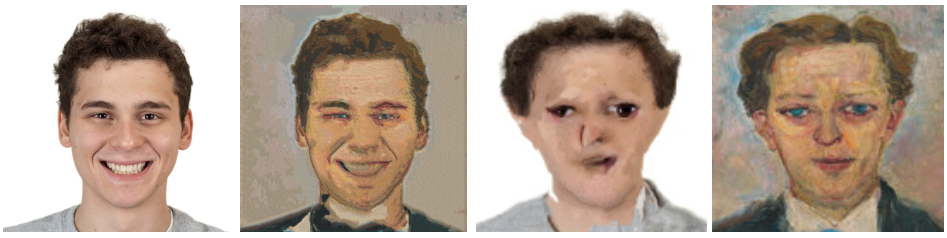
3.4 Related work

In this section, two papers related to local style transfer methods are discussed. The first, *Visual Attribute Transfer through Deep Image Analogy* [13] covers an alternative method of localized style transfer while the second, *"Automatic Semantic Style Transfer using Deep Convolutional Neural Networks and Soft Masks"* [23], is complementary to the Semantic Style Transfer technique and tackles the issue of automatically producing semantic maps, which are required by this algorithm.

3.4.1 Visual Attribute Transfer through Deep Image Analogy

Image Analogies and Style Transfer

A different way to perform local style transfer is *Visual Attribute Transfer through Deep Image Analogy* [13] by Liao et al. As its name implies, the technique is based on the notion of *Image Analogy*, which was introduced by Hertzmann et al. in [8]. The concept can be described as follows: given a pair of images B and B' which are related but belong to different domains (in this case, B would be a photograph and B' a stylized version of B) and an image A belonging to the same domain as B (here, a photo of a different individual), the goal is to find the image A' which is analogous to B', in that it the equivalent to A of B' to B. To clarify, an example of images A, A', B and B' is depicted at Figure 3.11. A' and B' specifically were generated using the code provided by the authors of *Deep Image Analogy*, hence the image B is flawed. One can however imagine that this was a real picture provided as input.



A A' B B'

Figure (3.11) Illustration of an image analogy. Image A' is the analogous of B' as the transformation to obtain A' from A is the same as the operation yielding B' from B. Image A is a sample of the Chicago Faces Database [15]. B': *Der Trance-spieler* by Oskar Kokoschka. A' and B were generated thanks to the *Deep-Image-Analogy* software.

Image analogies can then be used as a supervised form of style transfer, where the style image is in fact an example of the expected output for a specific, known content image. A system can then learn the transformation necessary to obtain the artwork from

the content image and apply this transformation on another photo. The main issue with this procedure is the loss in generality, as the required pair of content and style images is in practice usually nonexistent.

The Deep Image Analogy pipeline

In *Deep Image Analogy* by contrast, no example pair is required, which makes this method just as general as other local style transfer procedures. Instead, both images A' and B are reconstructed, according to a process described below. This is also represented on Figure 3.12.

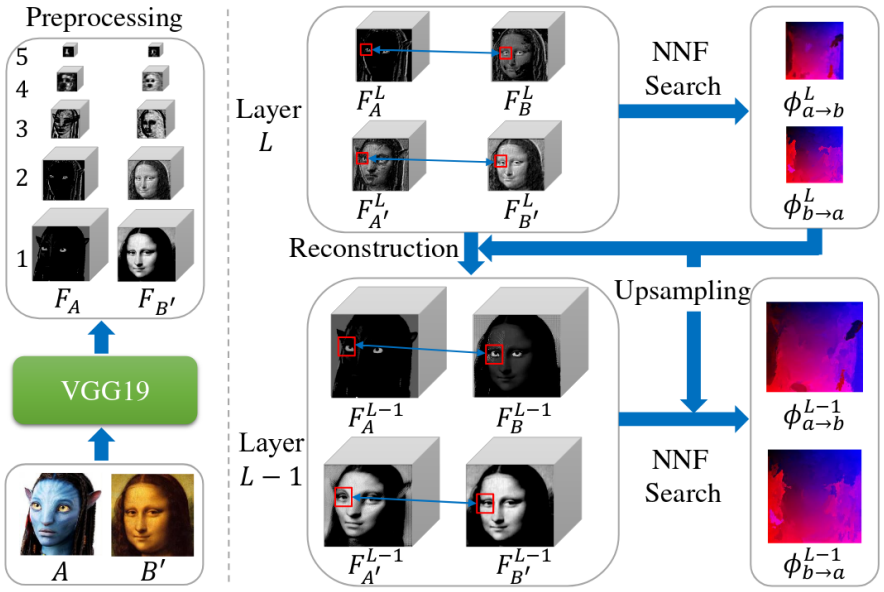


Figure (3.12) Representation of the full *Deep-Image-Analogy* system. Image taken from [13]. A bidirectional matching between images A and B' is created and gradually refined by progressing downwards in the feature maps pyramid. This correspondence can then be used to generate both images A' and B: To retrieve A' for example, one can sample from B' at the pixel locations given by the mapping $\phi_{a\to b}$.

- First, as represented by the Preprocessing step on Figure 3.12, the content image A and style image B' are fed to the VGG-19 classifier in order to extract features at layers `relu1-1`, `relu2-1`, `relu3-1`, `relu4-1` and `relu5-1`.

- Then, the algorithm focuses on the deepest layer considered, *i.e.* `relu5-1`. At this depth, feature maps mostly encode content information and disregard details such as color or texture in the original images. Therefore, the assumption that activations F_A^5 and $F_{A'}^5$ are similar is reasonable (since A and A' share the same content) and the feature map $F_{A'}^5$ for the yet unknown image A' is created, as a copy of F_A^5. Similarly, $F_{B'}^5$ is duplicated to obtain F_B^5.

- Still at layer `relu5-1`, a matching is established between images A and B (the correspondence is the same between A' and B'), as shown at the top of Figure 3.12. This is done by Nearest Neighbor Field Search (see NNF Search on the picture) and is very similar to the matching performed for the style transfer with MRFs: for each pixel p in A, the mapping $\phi_{a\to b}^{L=5}$ returns the index of the pixel q in B such that the kxk patch around pixel q is the most similar to the kxk patch around pixel p, where k is set to 3 at layer `relu5-1`. The same operation is repeated in the other direction to obtain the mapping $\phi_{b\to a}^{L=5}$. This mapping is the key to the style transfer problem: sampling from B' the pixels given by $\phi_{a\to b}$ results in the image A' which is (part of) the desired output, assuming that $\phi_{a\to b}$ is accurate. $\phi_{a\to b}^{L=5}$ is however a very coarse and unrefined matching which, if used after being upsampled, does not lead to a convincing result. As such, the next steps are concerned with progressively descending the pyramid of feature maps – reconstructing the activations of A' and B in lower layers in the process – to refine $\phi_{a\to b}$ and $\phi_{b\to a}$.

- To obtain the matching at layer $L-1$ by Nearest Neighbor Field Search, the feature maps of B and A' must be estimated[3]. Unlike at the deepest layer, the assumption that the feature maps for A and A' are similar does not hold anymore. The feature map $F_{A'}^{L-1}$ must thus be reconstructed. Without delving into details, $F_{A'}^{L-1}$ is approximated by a weighted sum of F_{A}^{L-1} (which, along with some superficial details, captures the content information that $F_{A'}^{L-1}$ should also contain) another image[4] $R_{B'}^{L-1}$ which, at least partially captures information about the style of B'. This other image is constructed by warping the feature map $F_{B'}^{L}$ using the mapping $\phi_{b\to a}$ and applying a transposed convolution to it (essentially adapting it to the $L-1^{th}$ layer). This last image $R_{B'}^{L-1}$ was created to share the approximate shape of F_{A}^{L-1} thanks to the mapping.

  To summarize, $F_{A'}^{L-1}$ is obtained by combining two latent representations, respectively holding some information about the content of image A and style of image B. F_{B}^{L-1} is created in a similar fashion. With approximations of $F_{A'}^{L-1}$ and F_{B}^{L-1} now known, the patch matching step (Nearest Neighbor Field Search) can be carried out. This time, the matching at layer L, once upsampled, is used to make informed decisions when selecting the best matching patches.

- The last step, explained here above, is repeated up to the first layer, `relu1-1`. This eventually yields the final mappings $\phi_{a\to b}$ and $\phi_{b\to a}$. Since the horizontal and vertical dimensions of the feature maps at layer `relu1-1` are the same as for the reference images, A' and B can directly be reconstructed with the mapping ϕ. The value of each pixel p of A' (which is the most interesting output in this case, as it is the output of the transformation from a photo to an artwork) can for instance be retrieved by applying

$$A'(p) = \frac{1}{k^2} \sum_{x \in N(p)} B'(\phi_{a\to b}(x)),$$

[3]While the matching could theoretically be determined using only A and B', this is difficult due to the large variations between the images, hence the matching is computed between A and B (and between A' and B') as this is an easier problem to solve.

[4]In the general sense, as this image contains more than 3 channels.

where $N(p)$ is the patch of size kxk centered around pixel p. This amounts to assigning the average value of the best matching patch in B' to the center of each patch in A and is reminiscent of the operation performed at Figure 3.3.

Figure (3.13) Local style transfer performed using the *Deep Image Analogy* method. The results were generated using the code released by the authors of [13]. The leftmost column contains the style images while the second and third columns are the output of the style transfer for the photos presented at Figures 3.6 and 3.5 respectively.

Results

Figure 3.13 shows the application of *Deep Image Analogy* to images which were earlier used to demonstrate the capabilities of the localized style transfer methods based on Markov Random Fields. The input images satisfy the conditions for the method to properly function since they are semantically very similar, even though they are rather different on a purely visual level. Overall, it appears that, compared to the patch-based and semantic style transfer, style is more closely matched while content is less accurately portrayed. Similarly to previous algorithms, several artifacts appear (see for instance the duplication of the nose and mouth on the first row, the seemingly random additions of eyes on the middle image) though the quality of results, which appears to be style-dependent, can be impressive, in particular for the first row. The issue of potential semantic mismatches between the images is also highlighted: on row 2, for instance, a beard is added when there was none for both subjects, suggesting that the algorithm is not prepared to deal with instances where specific elements appear in one image and not in the other.

3.4.2 Automatic Segmentation

A disadvantage of semantic style transfer compared to the original neural style transfer method with Markov Random Fields is the need to provide semantic maps for both the content and style images, which can be a time consuming task if done manually. This issue is addressed in the paper *"Automatic Semantic Style Transfer using Deep Convolutional Neural Networks and Soft Masks"* [23] by Zhao et al.

The generation of semantic maps is carried out in multiple steps, the first of which involves roughly segmenting the pictures: using an architecture named CRF-RNN, probability maps are built to encode the likelihood of each pixel to correspond to a specific object or category. The second stage is concerned with skin detection, which is useful since it, when combined with the segmentation information, allows to separate the skin from features such as hair. This step is only performed for realistic images (*i.e.* the content) as the variability in color and texture in stylized images make this technique unreliable. Finally, faces are finely segmented into the different characteristic regions. This is achieved partially through the use of facial landmarks, which are detected using OpenFace [1]. In the case of photos, the output of step two is used to improve the fidelity and robustness of the last operation.

An interesting consideration is the use by the authors of soft masks (as opposed to binary masks) for the segmentation. These allow for a representation of uncertainty which is welcome in such an automated task. Soft masks were shown to yield better results visually. The paper also highlights the challenge of segmentation with artistic imagery, as while results appear convincing for the style images used for testing, performance is usually inferior when compared to photorealistic content.

3.5 Discussion

In this chapter, different methods of local style transfer (as opposed to style transfer with summary statistics, examined in chapter 2 with the algorithm by Gatys et al.) were studied and presented.

First was the patch-based style transfer, with Markov Random Fields. This technique is promising as it is, unlike the first neural style transfer algorithm, context sensitive, leading it to produce better results, at least when the reference images are reasonably similar in structure. Some artifacts were however shown to occur in parts of the output where the content had no close match in the style image.

The number of artifacts was reduced when adding semantic maps to the algorithm, effectively guiding the patch matching process to achieve better results. This jump in quality however comes at the cost of more complexity, since accurate semantic maps of both artworks and photos must be provided. An automated method to create these semantic maps was briefly introduced to address this specific issue.

Finally, Deep Image Analogies [13] were introduced. This technique shares some defining traits with the semantic style transfer, like the patch matching in the latent space and the use of semantic correspondences between inputs. It is however adheres to a rather different paradigm: In Deep Image Analogy, the matching phase is performed to

build reconstructions of the desired outputs' activations at different layers, while in patch-based/semantic style transfer, the matching is used to compute a loss function. semantic style transfer is indeed an optimization problem like the algorithm by Gatys et al, while Deep Image Analogy solves the problem differently, by creating a matching to apply to the original images, which is initialized at a deep layer and refined progressively when progressing to shallower layers. An implication of these diverse ways of proceeding is that in semantic style transfer, the mappings at different layers are independent, while they are connected in Deep Image Analogies. This link between mappings at different layers appears to have an affect on visual quality, and this idea of having a consistent matching across layers is in fact used in chapter 5.

Chapter 4

Style Transfer For Head Portraits

The techniques explored in previous chapters are general methods, in the sense that the content to be inserted in an art piece is not restricted to a particular type of object. The subject considered, style transfer for head portraits, is on the other hand more specialized. Making use of this additional constraint, the possibility of improving the algorithm in the specific case of portrait style transfer is studied.

A paper by Selim et al. (2016) [19] introduced a modified version of the original neural style transfer algorithm by Gatys et al., designed specifically for head portraits. The difference with this project comes from the fact that the synthesized image is composited onto an arbitrary background. The main contribution of this paper is the modification of features used in the content-related term of the loss function, designed to better maintain the integrity of facial features. In this chapter, this algorithm and the results it produces are discussed, alongside an alignment preprocessing step required by the technique. The implementation constitutes an extension of the PyTorch code developed for the neural style transfer technique in chapter 2 and makes use of the *face morpher* [18] python library.

4.1 Gain maps

Since only the content loss is affected (the style loss is the mean squared error of the gram matrices of feature maps for a set of layers) by modifications with respect to the algorithm by Gatys et al., this technique can be combined with the *deep painterly harmonization* method (where the content loss is shared with the original algorithm).

The modified content loss is as follows:

$$\mathcal{L}_c = \frac{1}{2} \sum_{i,j} (F_{ij}^{content} - M_{ij}^{output})^2$$

where $F^{content}$ are the features extracted at the content layer(s) for the original content image and M^{output} are the following modified features:

$$M^{output} = F^{content} \cdot G$$

where F^{output} are extracted similarly to $F^{content}$, but this time for the image being optimized (*i.e.* the current output) and G, which the authors call a *gain map* is defined as

$$G = \frac{F^{style}}{F^{content} + \epsilon}$$

with $\epsilon = 10^{-4}$ to avoid dividing by zero. With this description, the modified feature maps actually become the extracted features of the style image at the content layer(s), which negates the point of the content loss. To preserve at least partially the content, the gain maps have to be constrained. For this reason, values of G are limited to the interval $[g_{min}, g_{max}]$. With g_{min} and g_{max} close to 1, content is better preserved while a large interval favors the style features. In the paper, default values are 0.7 and 5.0 respectively.

4.2 Alignment

The addition of gain maps to the optimization process helps to better retain the original painting's texture when creating the modified portrait. Because the faces in the content and style images are not aligned however, directly computing the gain maps using features extracted from the reference images is problematic. For instance, texture information linked to the eyebrows in the style image could end up being applied on the forehead of the inserted face, which would be detrimental to the preservation of facial integrity. A pre-processing step is thus required to match the visages present in the content and style image and avoid the formation of artifacts. This is done in three steps: *identification of facial landmarks*, *triangulation* and *warping* which are detailed below.

4.2.1 Facial landmarks detection

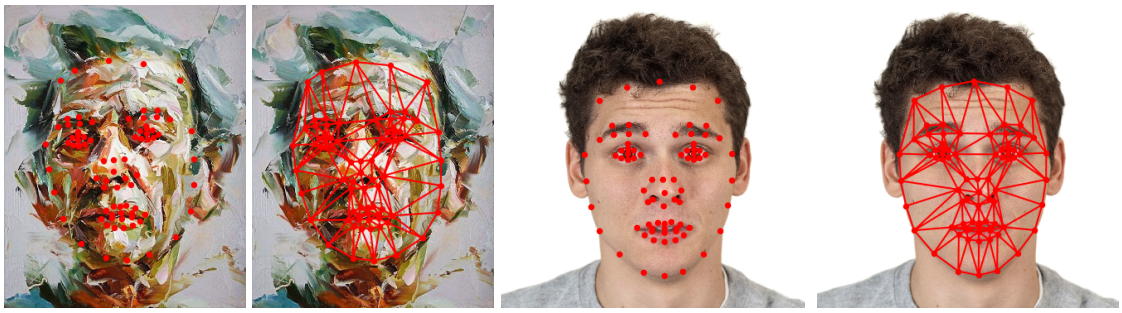
The alignment procedure involves first identifying key landmarks on the subjects' faces (both in the content and style image). These landmarks serve the purpose of delimiting the area of the painting to be selected when creating the gain maps as well as to establish a correspondence between the two pictures. This correspondence is later used to warp the face in the style image in such a way that both sets of landmarks are aligned.

The *face morpher* python library [18] is used to perform the alignment. The package relies on the *stasm* [16] software to locate 79 facial features, covering the outline of the face, mouth, nose, eyes and eyebrows. These are ordered: the n^{th} point in the sets of landmarks for the two considered faces refers to the same feature (*i.e.* the tip of the nose).

Figures 4.1.(a) and 4.1.(c) show the data points obtained for a set of two reference images. The result for this example is slightly off for landmarks related to the nose on the stylized image although this doesn't seem to impact the quality of the alignment (see Figure 4.2) and such imperfections are to be expected for non photo-realistic inputs.

4.2.2 Triangulation

Given the available sets of landmarks for both faces, a possible strategy to carry out the alignment is to first link the data points related to each image by creating two meshes. The



(a) (b) (c) (d)

Figure (4.1) (a), (c): Detection of facial landmarks. (b): Delaunay Triangulation of the landmarks identified in Figure (a). (d): Reproduction of the triangulation obtained in (b) for the set of landmarks relative to Figure (c).

idea is to first generate the mesh for one of the inputs (for instance, the style image) and then replicate it for the second input (meaning that two landmarks linked in one image are also linked in the second). Once these structures are defined, the mesh corresponding to the style image can be then warped to match the content image, which is explained at the next step.

A sound technique to create these meshes is the Delaunay triangulation. This yields a mesh of triangles with the following property: no point (landmark) is located inside the circumscribed circle of any triangle in the mesh (apart from the points forming said triangle). This translates to a triangulation which maximizes the smallest angle of each triangle, and in doing so avoids thin and elongated triangles, which in turn is beneficial to the next phase as stretching the triangles involves a re-sampling operation that can be less robust when dealing with elongated triangles.

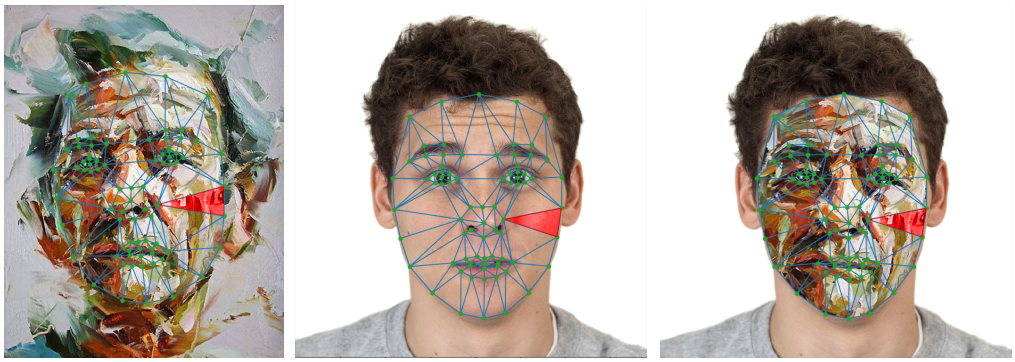
Figures 4.1.(b) illustrates a Delaunay triangulation computed from the landmarks at Figure 4.1.(a). Since it is necessary to use the same sets of triangles for the two images in order to establish a matching, the landmarks of the content image are linked according to the triangulation relative to the style image. The result is shown on Figure 4.1.(d).

4.2.3 Warping

Once the triangulations are available, the style image (more specifically, the face within it) can be distorted such that its facial landmarks are aligned with those of the content image. This is achieved by applying an affine transform to each triangle. Because pixels within the triangles are either stretched or compressed, it is necessary to re-sample from the style image. In the implementation used for the task, this is achieved by the classic bilinear interpolation. The warping stage is depicted on Figure 4.2.

4.2.4 Failure case

The alignment phase is a crucial step of the style transfer with head portraits, and is necessary to produce accurate results. Figure 4.3 presents a failure case, directly linked to



(a) (b) (c)

Figure (4.2) Visualization of the alignment phase. (a), (b): Triangulations obtained at the previous step. (c): The face in portrait (a) is warped to match face (b) by an affine transformation, based on the triangulations. For instance, the triangle highlighted in red on (a) is warped to match the coordinates of the corresponding triangle in (b).

the alignment procedure, highlighting its importance. In this case, the defect is due to the detection of facial landmarks, which failed to correctly identify part of the face contour (as visible along the jaw area). I suspect this was due to the combination of an rather elongated shape for the face, combined with a strong stylization for the portrait. This issue was not encountered for other, more realistic artworks, though this case invites the use of a face detection software better suited to such difficult conditions.

(a) (b) (c) (d)

Figure (4.3) Illustration of a failure case. (a): Self-Portrait by André Derain. (b): content. (c): aligned image (fed to the network to generate gain maps at the appropriate layers). (d): result of the style transfer.

4.3 Related Work

This work is concerned with transferring the style of artworks to photorealistic images. Another interesting application, however, is the transfer of style from one photo to another. In particular, the paper *"Style transfer for headshot portraits"* [20] by Shih et al,

this problematic is addressed for facial portraits. This publication, which may have been an inspiration for [19], also makes use of gain maps. Convincing results are achieved by decomposing the two photos into 'Laplacian stacks', filtering the two inputs photos to extract high and low frequency images. For every pair of extracted images, a dense matching is performed to generate the gain maps, evoking the alignment procedure detailed in section 4.2. The gain maps are applied at each level in the stack to the content image, transmitting information about contrast and lighting, and the resulting pictures are then aggregated to generate the final result. The authors show how their separation of images into high and low frequency content allows the transfer by gain maps of detailed textures (skin, beards) while capturing coarser features (appearance of the eyes, various highlights).

4.4 Results

Images generated with the technique at hand are portrayed on Figure 4.4. To show the impact of this method of style transfer for head portraits, the figure includes observed outputs for three pictures of the same individual, each depicting a distinct facial expression. A point of comparison is provided with the portraits obtained from the classic neural style transfer method of chapter 2. Additionally, the application of gain maps is visualized (although in rgb space, and not for the feature maps, where gain maps are used in the implementation). These appear to dictate rather strongly the placement of highlights and, as expected, the facial expressions observed on row 2 (impact of gain maps) are highly correlated with those of row 4 (neural style transfer for head portraits).

Compared to the classic neural style transfer algorithm by Gatys et al., some minor deformations are observed (see the mouth in columns 1 and 3 for instance). The overall facial expressions are however well preserved and it is easy to tell apart the three images. It is also noteworthy that the faces (the regions to which the gain maps were applied) closely resemble the style portrait and stand out from the background. By contrast, the other outputs (in row 3) lack a sense of depth, giving the impression that a filter was quite uniformly applied on top of the input photograph, which leads to a more artificial appearance.

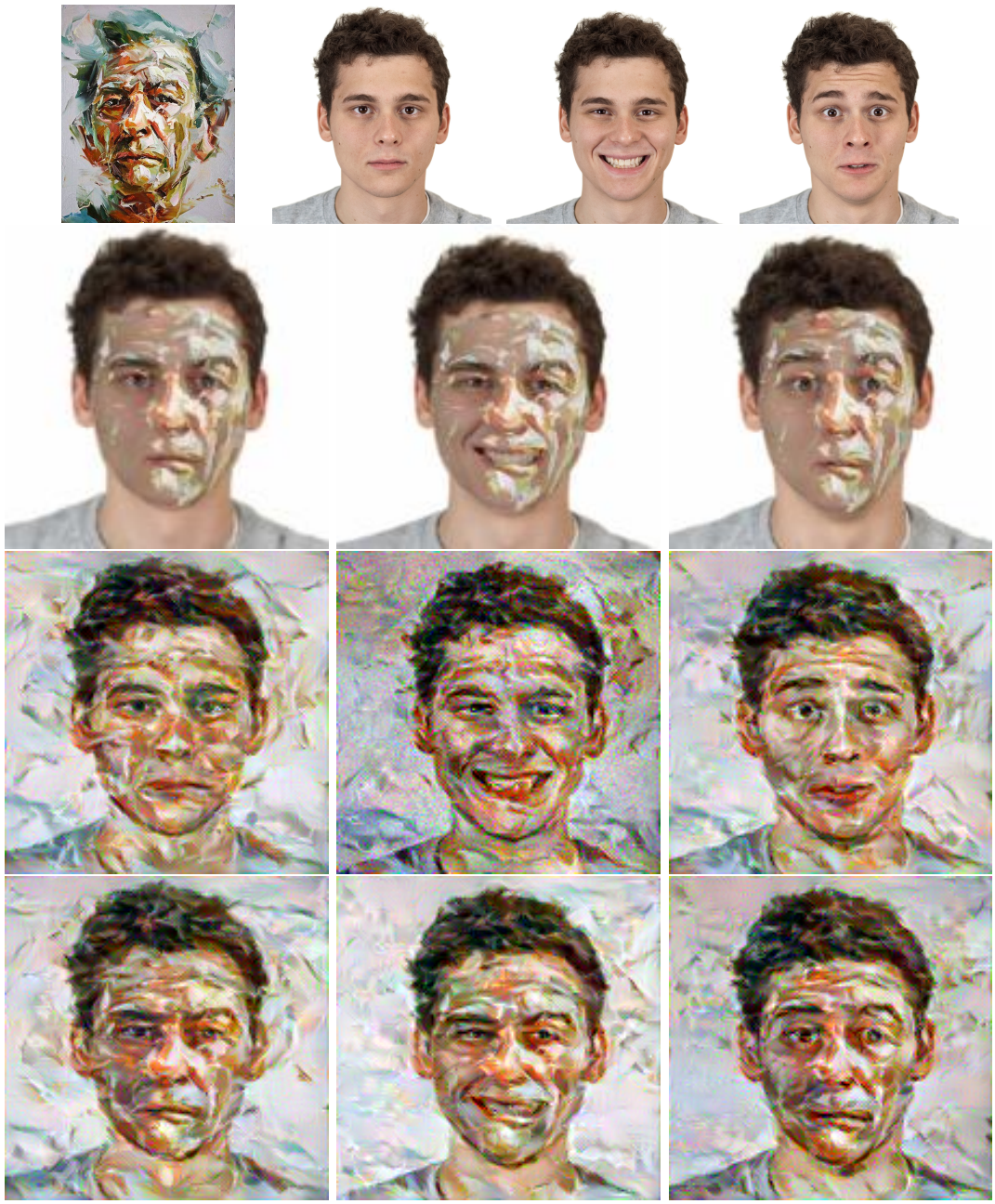


Figure (4.4) Style transfer for head portraits with different facial expressions. First row: style and content reference images. Second row: content images multiplied by the gain maps. Third row: neural style transfer (Gatys et al.). Fourth row: style transfer for head portraits (the content features are multiplied by the gain maps). Style image: *Whirlwind* by Paul Wright. Content images from the Chicago faces database [15]

Chapter 5

Portrait Style Transfer With Background Integration

The core part of this work revolves around substituting faces in stylized portraits by faces extracted from real photographs, while aiming to keep the style of the original portrait (painting) intact. This can prove challenging as typical style transfer techniques are designed to match global statistics of the style image and don't perform as well locally.

In this chapter, the paper *"Deep Painterly Harmonization"* (Luan et al., 2018) [14] is first examined, as it is particularly relevant with respect to the tackled subject. The technique presented by the authors enables the user to paste a photorealistic object onto a painting, and transform the resulting image to match the general appearance of the reference painting.

Subsequently, a combination of Deep Painterly Harmonization [14] with the style transfer for head portraits presented in chapter 4 is discussed and tested, with the goal of improving the quality of the generated artworks in the particular case of head portraits thanks to this extension. To this end, outputs of this last method are compared to those of Deep Painterly Harmonization for different pairs of photo and style images.

5.1 Deep Painterly Harmonization

The key to the Deep Painterly Harmonization [14] algorithm's success appears to be its use of two consecutive optimization passes (also called phases in this document) designed to produce both robust and detailed results. In this section, both passes and their objective are first discussed. Then, results obtained using a custom implementation of the algorithm described in [14] are presented.

The first pass aims to generate a coarse, but robust approximation of the final result, and is mostly related to the patch-based method presented in chapter 3. This phase's sole purpose is to generate a good initialization for the image to be generated in phase 2.

The second pass does indeed take the first output as starting point to create a more refined portrait, with a focus on preserving texture and fine details, which is achieved partly through the use of features extracted from shallow layers in the network.

In both phases, a mask is introduced to indicate where the image should and should not be modified. This mask is regularly up and down sampled to match the resolution of feature maps at different layers. Additionally, it is important to mention that the mask does not need to perfectly fit the inserted content. Instead, it is better to have the mask cover slightly more than the inserted face, in order to allow the algorithm to blend it with the rest of the painting, hiding the rough edges in the process.

5.1.1 Phase 1

The first pass is mainly based on the patch-based method proposed by Li et al. [11] which the authors found to be more appropriate than the classic technique by Gatys et al. [**?**] based on Gram matrices, due to the patch-based technique being content-aware. This assertion is further supported by the observations made in chapter 3. In this implementation, the dimension of the activations patches is 3x3xC, C being the number of channels.

The primary difference with Li et al.'s approach stems from the fact that the style loss is not computed by directly measuring the distance between matches patches in the input and style feature maps. The style loss, for each considered layer, is instead obtained by comparing the Gram matrices of feature maps related to the input image and modified feature maps for the style image. These modified maps are obtained by replacing each patch extracted from input feature maps with its corresponding patch from the style feature maps, and then pooling[1] from the resulting set of patches, which results in new style feature maps (sharing the shape of the originals).

While applying the gram matrices is an optional step, which was not originally considered by the authors of the paper, it is shown in [14] that comparing the Gram matrices, instead of the feature maps directly (which would be equivalent to applying the patch-based method developed by Li et al. [11] and reviewed in chapter 3) is beneficial, as it reduces artifacts and deformations otherwise observed.

The feature maps, once transformed, are multiplied by a resized copy of the mask, in order to only modify the image where content was inserted. Applying this mask after re-mapping the style feature maps is crucial as the selected style patches can then originate from other parts of the paintings. For instance, a patch located on the forehead in the content image could be matched to a patch of skin anywhere on the style image. Intuitively, matching global statistics about the activations should indeed produce less artifacts since using such an abstract representation of style provides less incentive for the network to simply copy entire patches from the style image. Instead, the technique can be viewed as a localized version of the algorithm by Gatys et al. [**?**], where style is still represented based on global statistics but those statistics are computed based on a feature map which more closely resembles the content feature map.

Independent Mapping

An important characteristic of the first pass is that the mapping used to transform the style feature maps – and align them with the content image – is different at each layer, which

[1] The pixel at the center of each patch is extracted

makes the first pass quite robust (compared to performing the mapping once at a specific layer and upsampling or downsampling it to the others). During the first pass, the style layers are layers `relu3-1`, `relu4-1` and `relu5-1` of the VGG-19 network. Shallow layers such as `relu1-1` and `relu2-1` are not included, which is important since these lower layers tend to encode information about the finer visual details present in images. This omission then leads to images somewhat lackluster when it comes to the color and texture transfer. This is however intentional as the first phase must focus on the content (substance over form) in order to be robust and output an artifact-free image.

Style Loss

Procedure 5 Phase 1 - Style Loss

parameters: F^O and F^S, lists of feature maps extracted for the generated and style image at the all the style layers. The Mask indicating where to perform the style transfer. k, A list holding the size of patches chosen for all style layers.
require: $\text{Shape}(F_l^O) = \text{Shape}(F_l^S)\ \forall l \in [1, \text{Length}(F^O)]$
return: The style loss for the algorithm's first phase.

function STYLE-LOSS($F^O, F^S, Mask, k$)
  $L_s \leftarrow 0$
  for $l \in style\_layers$ **do**
    $H_l, W_l, C_l \leftarrow \text{Shape}(F_l^O)$
    $Mask \leftarrow \text{Downsample}(Mask,\ H_l, W_l)$
    ▷ Retrieve the patches covered by the mask and perform the matching
    $\Psi_l^O \leftarrow \text{ExtractPatches}(F_l^O,\ k_l,\ Mask)$
    $\Psi^S \leftarrow \text{ExtractPatches}(F^S,\ k_l)$
    $\Psi_l^S \leftarrow \text{NearestNeighbors}(\Psi_l^O,\ \Psi_l^S)$
    ▷ Build a modified feature map F_l^S, such that it aligns with F_l^O in the area
    ▷ covered by the Mask. This is achieved by sampling each pixel in this area
    ▷ from the center of the best matching style patch.
    $F_l^S \leftarrow \text{Resample}(F_l^S, \Psi_l^O,\ \Psi_l^S, Mask)$
    ▷ The reconstructed style feature map and output feature map are used to
    ▷ compute a loss based on the comparison of Gram matrices. Masked out pixels
    ▷ are ignored (set to zero) for the computation.
    $G_l^O \leftarrow \text{GramMatrix}(F_l^O \bigodot Mask)$
    $G_l^S \leftarrow \text{GramMatrix}(F_l^S \bigodot Mask)$
    for $i \leftarrow 1$ to C_l^2 **do**
      $$L_s \leftarrow L_s + \frac{(G_{l,i}^O - G_{l,i}^S)^2}{\text{Length}(\Psi_l^O) \cdot (2 \cdot H \cdot W \cdot C)^2}$$
  return L_s

The core part of the algorithm's the first pass is undoubtedly its style loss function and is indicative of both the shared features and differences with previous optimization techniques. This loss is presented below with procedure 5. The pseudo-code, is written

differently from [14] to both emphasize the style loss and remain consistent with previous notations, thus ensuring that the algorithms are easily comparable.

5.1.2 Phase 2

With the first pass completed, the obtained image is used to initialize the second phase. During this second pass, style layers are `relu1-1`, `relu2-1`, `relu3-1` and `relu4-1`. Choosing these lower layers makes it easier to transfer color and texture properties, compared to higher layers which better represent semantics. This decision to use shallower layers is sound as while semantics are already well captured from the first phase – and, anticipating on the results section, do not tend to vary much afterwards – the texture and color of the style image are however generally not well transferred at this point.

Consistent Mapping

During this pass, a reference layer (`relu4-1` was picked by the authors) is chosen to perform the mapping of style features. This mapping is then upsampled to transform the other style feature maps, which leads to a more consistent output, as the modifications brought to each layer are consistent. This unified mapping echoes chapter 3, where the paper *Visual Attribute Transfer through Deep Image Analogy* [13] implements a similar consistent mapping between layers. In fact, a unified matching was shown to be superior to performing the matching differently at each layer in the specific case of this publication.

On the implementation side, it can be noted that `relu3-1` is used instead of `relu4-1` in the PyTorch implementation of the algorithm developed for this work, as it yields more visually appealing results. This discrepancy might be the result of an interpretation of the paper which may slightly differ from the authors' original vision.

Refined Matching

A common issue with the matching is the lack of control over which style patch is selected as the best fit for a certain patch in a feature map extracted from the image being created. Patches are matched individually, without considering their immediate surroundings: it is possible that two adjacent patches in the style features (for instance both covering part of an eye) are not matched to the adjacent locations in new image due to the existence of another, distant patch achieving a slightly better similarity score.

To remedy this issue, the mapping is modified to remove outliers before being up or down sampled, with the goal of trying to ensure that adjacent style patches remain adjacent after the mapping is performed when possible. The function responsible for generating and refining the mapping is presented in Procedure 6. Once again, the algorithm is presented a little differently when compared to the paper, to both preserve consistency between the notations and better reflect the actual implementation.

To obtain the modified – and improved – mapping, the following steps are followed: for each patch of activations related to the new image and covered by the mask, all of its direct neighbors are considered. For each of these neighbors and the original patch, the best fits are determined, using the classic nearest neighbor matching technique. For

Procedure 6 Phase 2 - Reference Matching

parameters: F^O_{ref} and F^S_{ref}, feature maps extracted for the generated and style image at the designated reference layer. The Mask indicating where to perform the style transfer. k_{ref}, the size of patches for the reference layer.
require: Shape(F^O_{ref}) = Shape(F^S_{ref})
return: Ψ^O_{ref}, Ψ^S_{ref}, respectively the patches covered by the mask and the associated best matching style patches.

1: **function** REFERENCE-MATCHING($F^O_{ref}, F^S_{ref}, Mask, k_{ref}$)
2: ▷ First perform the patch matching process for a single reference layer
3: $H_{ref}, W_{ref}, C_{ref} \leftarrow$ Shape(F^O_l)
4: $Mask \leftarrow$ Downsample($Mask$, H_{ref}, W_{ref})
5: ▷ Only consider the patches in the relevant area (covered by the mask)
6: $\Psi^O_{ref} \leftarrow$ ExtractPatches(F^O_{ref}, $k_{ref}, Mask$)
7: $\Psi^S_{ref} \leftarrow$ ExtractPatches(F^S_{ref}, k_{ref})
8: $\Psi^S_{ref} \leftarrow$ NearestNeighbors(Ψ^O_{ref}, Ψ^S_{ref})
9: ▷ Increase the mapping's robustness
10: **for** $i \leftarrow 1$ to Length(Ψ^O_{ref}) **do**
11: ▷ Initialize a set of other possible matches for the patch at hand: $\Psi^O_{ref,i}$
12: $candidates \leftarrow \varnothing$
13: ▷ Consider $\Psi^O_{ref,i}$ and all patches adjacent to it in F^O_{ref}, in 8 directions
14: **for** $dx \leftarrow \{-1, 0, 1\}$ **do**
15: **for** $dy \leftarrow \{-1, 0, 1\}$ **do**
16: ▷ Get the index in Ψ^O_{ref} of the patch located at offset (dx, dy) from $\Psi^O_{ref,i}$
17: $j \leftarrow$ Neighbor($\Psi^O_{ref}, i, (dx, dy)$)
18: ▷ Get the index in Ψ^S_{ref} of the patch located at offset -(dx, dy) from $\Psi^S_{ref,j}$
19: ▷ *i.e.* apply the translation from line 17 in reverse to the style patch
20: ▷ which was determined to be the best match for $\Psi^O_{ref,j}$ earlier
21: $k \leftarrow$ Neighbor($\Psi^S_{ref}, j, (-dx, -dy)$)
22: ▷ Add the newly obtained style patch to the list of candidates
23: $candidates \leftarrow candidates \cup \{\Psi^S_{ref,k}\}$
24: ▷ Select the best candidate as the new match for patch $\Psi^O_{ref,i}$. The best
25: ▷ candidate is the one which, on average, resembles the style patches associated
26: ▷ to the neighbors of $\Psi^O_{ref,i}$ the most.
27: $\Psi^S_{ref,i} \leftarrow \underset{c \in candidates}{\arg\min} \sum_{dx=-1}^{1} \sum_{dy=-1}^{1} ||c - \Psi^S_{ref}[\text{Neighbor}(\Psi^O_{ref}, i, (dx, dy))]||^2$
28: **return** Ψ^O_{ref}, Ψ^S_{ref}

every style patch obtained in this way, its neighbor[2] is added to a set of candidates for

[2]The mentioned neighbor for the style patch is selected by applying the translation required to go from the original patch to its neighbor in the reverse direction, as is best described in procedure 6 from line 12 to line 23.

the new best fit. Then, the best candidate is identified. This candidate is the one which, on average, resembles the style patches associated to the neighbors of the original content patch the most.

Style Loss

With the reference matching explained in procedure 6, procedure 7 describes the style loss computation for the algorithm's second phase. The idea is essentially the following: first compute the mapping for the reference layer (in this case `relu3-1`); then, for every other layer (here `relu1-1`, `relu2-1` and `relu4-1`), apply this matching, taking into account the differences in resolution between the different layers. Finally, compute the style loss for each layer as the mean squared error of the differences between best matching activations patches, normalizing as needed based on the patch and feature map sizes.

5.1.3 Results

The results obtained for a set of different individual and paintings are included in Figures 5.1, 5.2 and 5.3. The generated images are presented both for the official implementation by the authors (which was publicly released) and for my implementation, written in PyTorch and based on my interpretation of the paper. It can be observed that results are not exactly identical, with those of Figure 5.1 proving especially distinct. The matter is rather subjective but I do think that while different, those results are of a similar quality to those created using the released software. A detail in image 5.1.(c), generated using the code of Luan et al. is however slightly problematic, as the make-up from the photograph also appears in the output, which makes the image feel unnatural; but these imperfections tend to appear on a case by case basis and might not be representative of the full spectrum of possibilities.

5.2 Face segmentation

Generating compositions such as those depicted at Figures 5.1, 5.2 and 5.3 can prove to be time consuming if done entirely manually, due to the need to extract the face from the input photograph. To greatly increase the ease of generating these compositions, an automated face segmentation solution is used.

The *face morpher* library [18], used in chapter 4, was initially considered as a candidate for this task. However, the observation of a few failure cases such as the one depicted in chapter 4 prompted the use of a more robust solution.

To this end, a solution for *"Deep segmentation in extremely hard conditions"* [17] [code] by Nirkin et al. was instead used. The code outputs the segmented face, which is then applied on a transparent background, making its insertion in the style portrait feasible. All modified portraits from the results section were generated through the use of this technique, without manual segmentation. It is in fact rather easy to tell that this segmentation was done automatically in figure 5.3.(c) as the contour of the face is not perfectly segmented, which is made more apparent due to the blue background. Thankfully, these imperfections

Procedure 7 Phase 2 - Style Loss

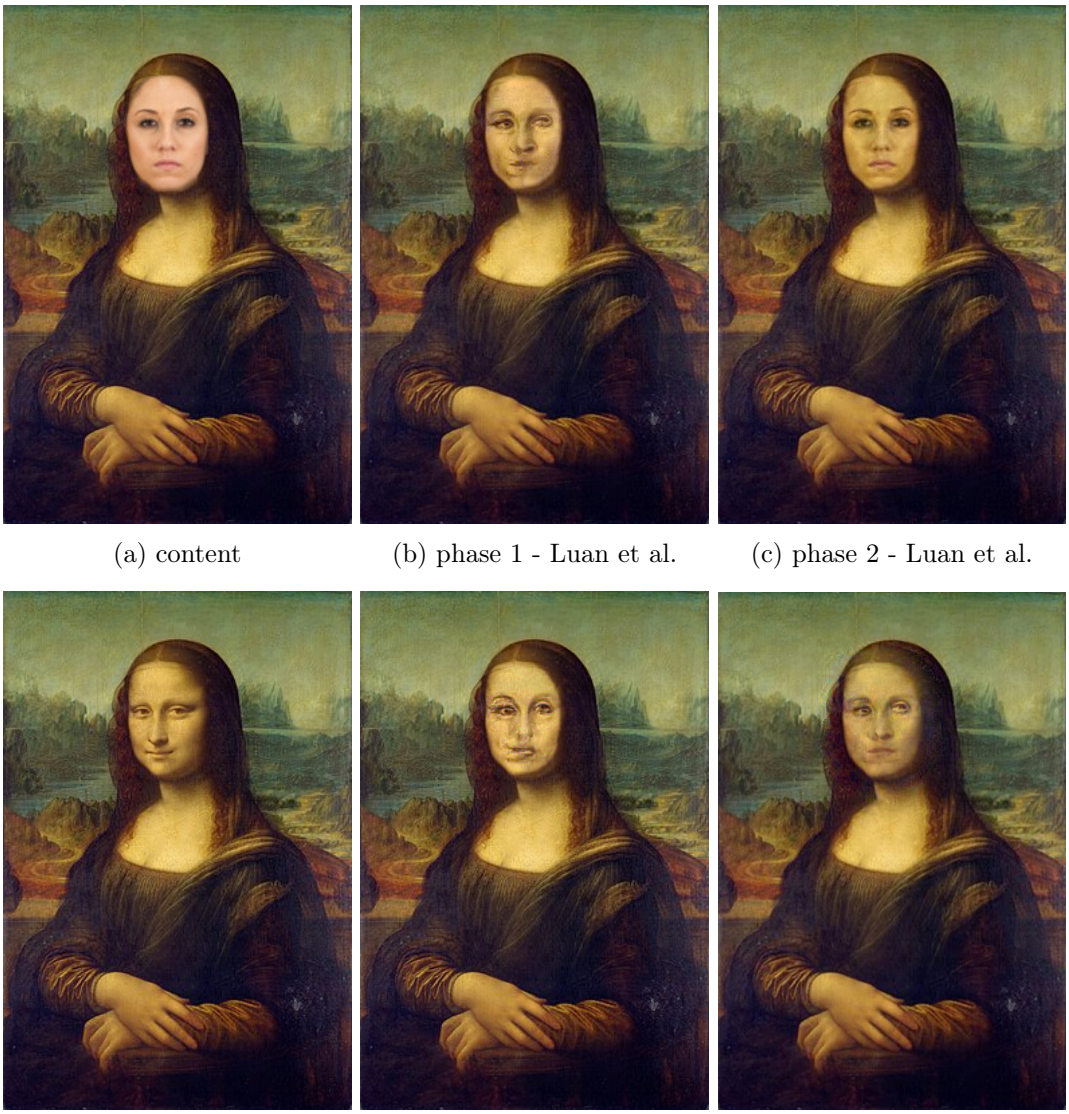
parameters: F^O and F^S, lists of feature maps extracted for the generated and style image at the all the style layers. The Mask indicating where to perform the style transfer. k, A list holding the size of patches chosen for all style layers. ref, the id of the reference style layer.
require: $\text{Shape}(F_l^O) = \text{Shape}(F_l^S)\ \forall l \in [1, \text{Length}(F^O)]$
return: The total style loss for the algorithm's second phase.

function STYLE-LOSS($F^O, F^S, Mask, k, ref$)
  $\Psi^O_{ref}, \Psi^S_{ref} \leftarrow$ReferenceMatching($F^O_{ref}, F^S_{ref}, Mask, k_{ref}$) ▷ see procedure 6
  $H_{ref}, W_{ref}, C_{ref} \leftarrow \text{Shape}(F_l^O)$
  $L_s \leftarrow 0$
  for $l \in style\_layers$ **do**
    $H_l, W_l, C_l \leftarrow \text{Shape}(F_l^O)$
    $Mask \leftarrow \text{Downsample}(Mask,\ H_l, W_l)$
    ▷ Retrieve the patches covered by the mask
    $\Psi_l^O \leftarrow \text{ExtractPatches}(F_l^O,\ k_l,\ Mask)$
    for $i \leftarrow 1$ to $\text{Length}(\Psi_l^O)$ **do**
      ▷ Reuse the reference mapping for other style layers
      if $l \neq l_{ref}$ **then**
        ▷ Retrieve the index of the patch centered at the same location in the
        ▷ reference layer, which can be computed knowing both layers' dimensions
        $j \leftarrow$GetPatchIndex$(\Psi^O_{l,i}, (H_l, W_l), (H_{ref}, W_{ref}))$
        ▷ The corresponding style patch, in the reference layer is then $\Psi^S_{ref,j}$.
        ▷ The equivalent style patch at layer l can then be recovered:
        $s \leftarrow$GetPatchIndex$(\Psi^S_{ref,j}, (H_{ref}, W_{ref}), (H_l, W_l))$
      else then
        $s \leftarrow i$
      ▷ Update the loss value
      $L_s = L_s + \dfrac{(\Psi^O_{l,i} - \Psi^S_{l,s})^2}{\text{Length}(\Psi_l^O) \cdot k_l^2}$
  return $\dfrac{L_s}{2 \cdot \text{Length}(style\_layers)}$

disappear starting at the first phase, demonstrating that there is room for relatively small defects in the input portraits.

5.3 Enhancement for head portraits

Leveraging the observations and comments formulated throughout this work, the deep harmonization technique is modified with the aim of generating better results for face portraits specifically. The style transfer for head portraits is an ideal fit as it is geared towards this specific purpose. Additionally, deep painterly harmonization and the style transfer for



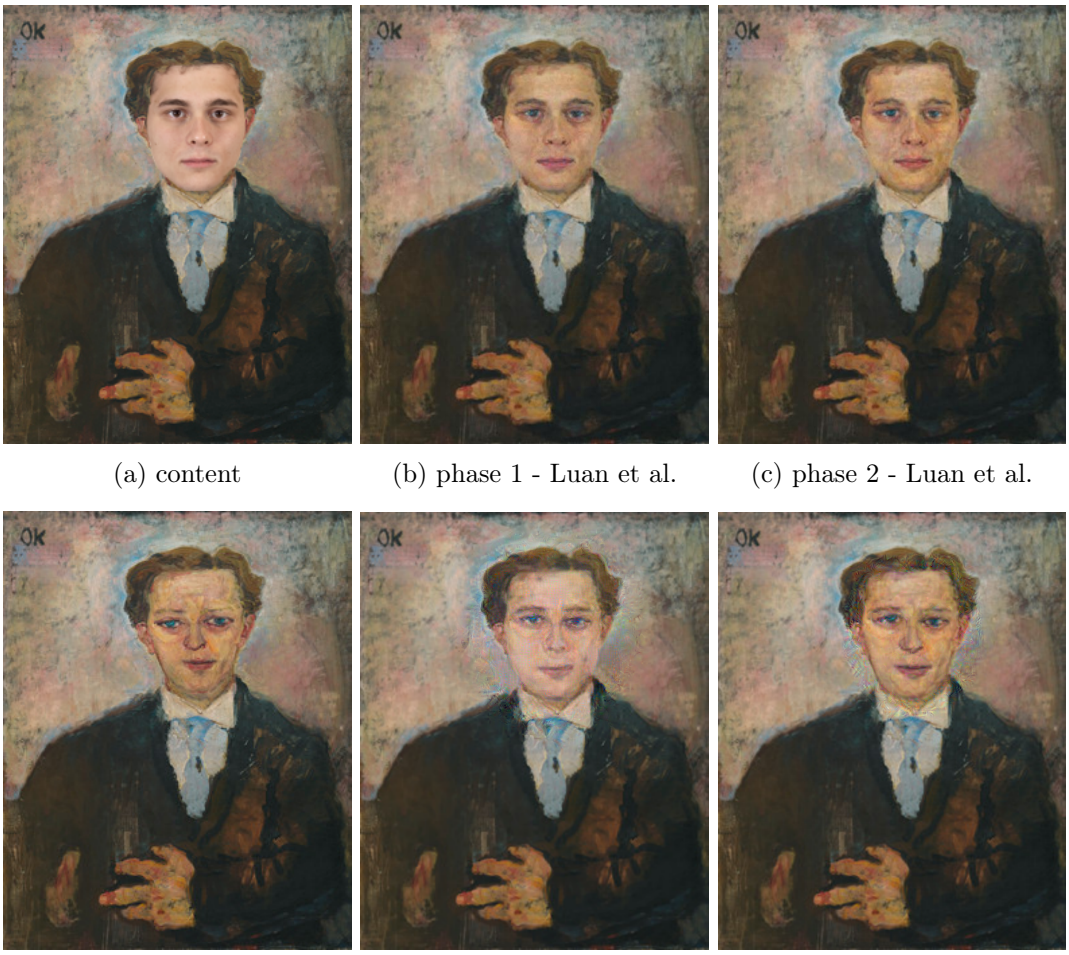
(a) content (b) phase 1 - Luan et al. (c) phase 2 - Luan et al.

(d) style (e) phase 1 (f) phase 2

Figure (5.1) Style transfer applied on a portrait. Style image: *Mona Lisa*, Leonardo Da Vinci. Face appearing in the content image (a) sampled from the Chicago Face Database [15]. (b,c): results obtained by *deep painterly harmonization* (Luan et al.) [14]. (e,f): portraits generated with my implementation of the algorithm

head portraits are quite compatible. Indeed, both can be seen as belonging to the same family of neural style transfer algorithms (optimization based) and most importantly, while the former essentially presents a unique style loss formulation at its core [3], the latter is mainly concerned with a modified content loss. In short, the modularity of the neural style transfer framework is exploited to combine the best features of a few methods reviewed in

[3] As underlined by procedures 5 to 7



(a) content (b) phase 1 - Luan et al. (c) phase 2 - Luan et al.

(d) style (e) phase 1 (f) phase 2

Figure (5.2) Style transfer applied on a portrait. Style image: Oskar Kokoschka, Der Trancespieler. Oil on canvas, (1909). Face appearing in the content image (a) sampled from the Chicago Face Database [15]. (b,c): results obtained by *deep painterly harmonization* (Luan et al.) [14]. (e,f): portraits generated with my implementation of the algorithm

this thesis, namely the *neural algorithm of artistic style* [**?**], *combination of markov random fields and convolutional neural networks for image synthesis* [11], *style transfer for head portraits* [19] and finally *deep painterly harmonization* [14]. The use of masks is integrated to the style transfer for head portraits part of the implementation to preserve the ability to seamlessly insert faces in portraits .

5.3.1 Results

Figure 5.4 illustrates the results of combining the deep painterly harmonization method with the style transfer for head portraits. Comparison with the base deep painterly harmonization technique is made possible by the vignettes added to each portrait, showing

(a) content (b) phase 1 - Luan et al. (c) phase 2 - Luan et al.

(d) style (e) phase 1 (f) phase 2

Figure (5.3) Style transfer applied on a portrait. Vincent van Gogh, Self-Portrait. Oil on canvas(1889). Face appearing in the content image (a) from a photo by Seth Johnson. (b,c): results obtained by *deep painterly harmonization* (Luan et al.) [14]. (e,f): portraits generated with my implementation of the algorithm

the related results from my implementation of the base deep painterly harmonization technique.

Several observations can be made: facial integrity is better preserved for pictures 5.4.(b) and 5.4.(h) in particular. The jump in fidelity is comparable to the one observed in chapter 4 when comparing the style transfer for head portraits technique with the algorithm of Gatys et al. Eyes are now symmetrical and the shape of the mouth appears more natural, which goes a long way to make the portraits more believable, bearing in mind our capacity to notice the smallest of deformations on faces. Impressively, the facial structure is preserved when going from the first to the second phase, yielding a more natural final output.

Overall, generated images appear improved from the base technique, as image 5.4.(c) in particular is vastly enhanced while picture 5.4.(i) shows an increased resemblance to the content image, while still being faithful to the style portrait. Figure 5.4.(f) appears on par with the previous result, which was already satisfactory – as could reasonably be expected since the content and style images are rather similar in this case.

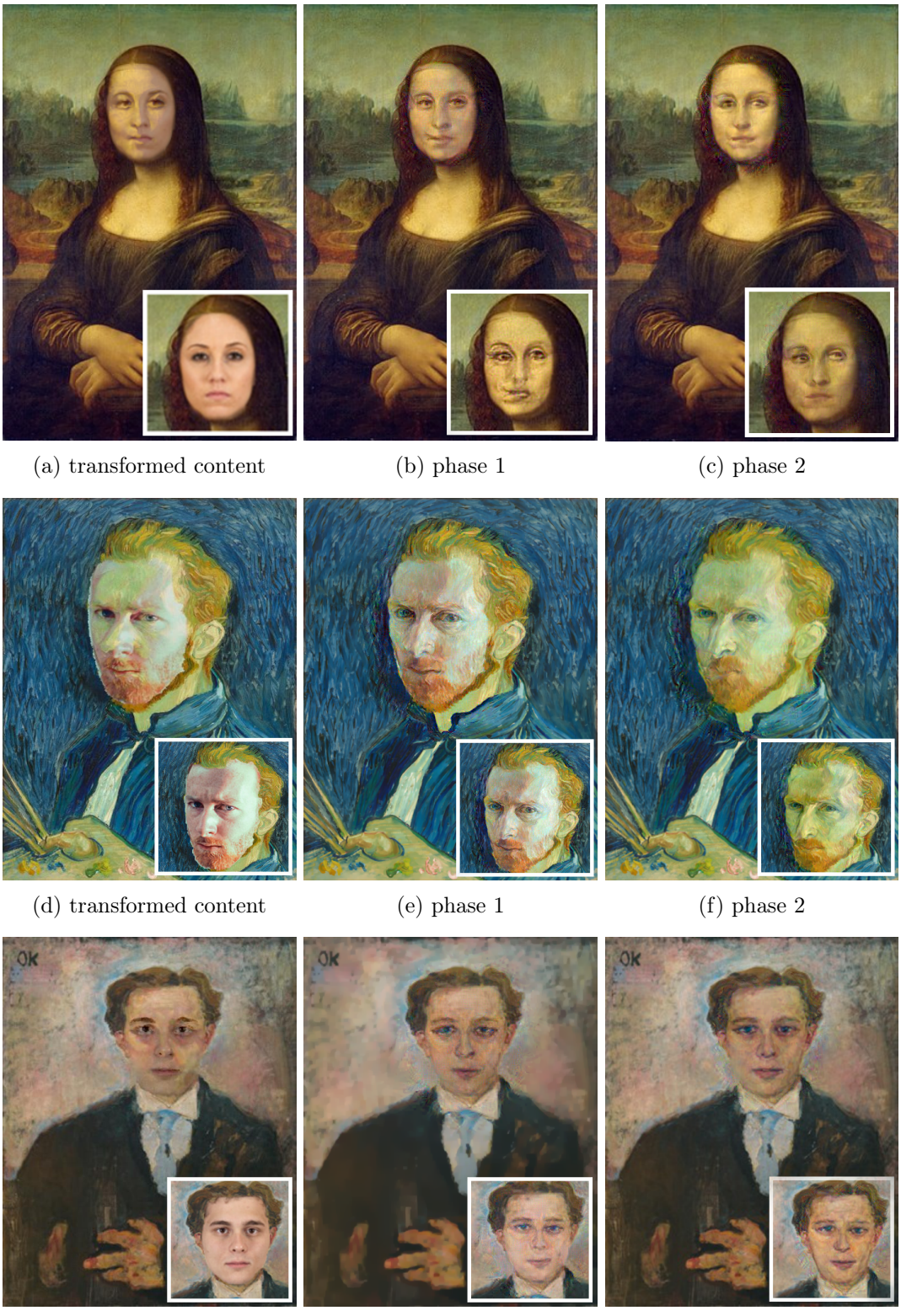
5.3.2 Limitations

Memory consumption

Ignoring the visual quality of the results for a moment, a major limitation of the proposed system for neural style transfer is its memory usage. On a Nvidia K80 GPU with 12gb of memory, which was used to run the final code, the resolution of results is limited at roughly 800 by 500 pixels (or equivalent). Increasing this value means it is likely to run out of memory at some point during this execution. While this issue could certainly be alleviated with a renewed on memory management, ensuring that memory is used optimally or through the use of costly high end computing resources complete with a large memory, it is important to note that, at each layer, the memory consumption increases linearly with the number of pixels and thus quadratically with respect to the images' height and width – thus making the problem difficult to solve.

Execution time

The second constraint is time: it can take from 10 to 15 minutes to generate a single image of size 800 by 500 using the proposed technique depending on the number of iterations, which is far from a desirable real-time solution. There is however hope to be had in this department, as Fast Style Transfer techniques such as the one developed in [10] could be combined with the presented method to obtain real-time results. This technique in particular, by Johnson et al., involves training a feedforward network for a given style with a large dataset of natural images, using the VGG classifier as a loss network to iteratively build a function that approximates the style transfer procedure. These fast style transfer methods typically require a long training phase to approximate the transformations obtained via optimization based techniques, and generally only work for a specific style or limited set of styles at once but ultimately produce near-instantaneous results once the setup is achieved, which is helpful if the objective is for instance to generate a large number of portraits from different photos with a single popular artwork.



(a) transformed content (b) phase 1 (c) phase 2

(d) transformed content (e) phase 1 (f) phase 2

(g) transformed content (h) phase 1 (i) phase 2

Figure (5.4) Style transfer for head portraits. The first column is obtained by multiplying the content image by the corresponding gain map (making use of the style image); this is provided to illustrate the effect of gain maps. In practice, gain maps are applied to the content features and not directly to the image itself. The smaller images are sampled from Figures 5.1, 5.2 and 5.3 to make the comparison with the base algorithm possible.

Chapter 6

Conclusion

In this thesis, the topic of style transfer in the particular case of face portraits was approached through a thorough exploration of optimization based neural style transfer techniques. These algorithms allow end users to automatically generate new artworks from photographs by replicating the style of a specific painting while staying faithful to the provided content.

The *Neural algorithm of artistic style* [6] by Gatys et al. was first reviewed in chapter 2, laying the foundation for later findings and observations. A framework for neural style transfer was introduced, unveiling a procedure based on the iterative, joint optimization of so-called content and style losses by gradient descent – with theses losses being computed using latent features, extracted by the pretrained VGG-19 image classifier from the input images.

Valuable insight was gleaned in chapter 2 regarding the inner workings of classifiers such as VGG-19 and the information it captures at various layers, through experimentation with the multitude of adjustable parameters of the algorithm. This was made possible thanks to a custom implementation, using the PyTorch framework, of the method. From these experiments, it appeared that the technique was not particularly well suited for style transfer when applied on head portraits, even though it is capable of impressive results for more general applications.

The primary culprit proved to be the lack of awareness from the algorithm with regards to context: there was no incentive to apply style information differently for the face and background, ultimately resulting in rather artificial images.

A different approach was necessary to ensure that the style transfer was carried out in a context sensitive fashion. To this end, local style transfer methods were reviewed in chapter 3, namely *"Combining markov random fields and convolutional neural networks for image synthesis."* [11], *"Semantic style transfer and turning two-bit doodles into fine artworks* [3]. Where the former technique introduced a brand new, *patch-based* procedure to compute a different style loss (compared to chapter 2), the latter improved upon it with the addition of *semantic maps*, providing guidance for the patch matching process in [11] to enhance results.

Both methods were implemented in PyTorch, re-using the aforementioned neural style transfer framework to generate new examples of style transfer. These outputs proved more promising than those of chapter 2 when the content and style images were reasonably similar in their structure.

Some artifacts were however observed where the reference images differed too significantly in the case of [11], artifacts which were thankfully reduced and alleviated by guiding the method with user provided semantic maps thanks to the semantic style transfer algorithm [3].

Still, the semantic style transfer method did not produce fully believable results, prompting the search of a different solution, ideally specialized to deal with face portraits. This was provided with the *"Painting style transfer for head portraits using convolutional neural networks"* [19]. This method, built on top of the algorithm developed by Gatys et al., makes use of *gain maps* to preserve facial integrity. A custom implementation was also provided, complete with a required facial alignment solution, developed using the *face morpher* [18] library. Convincing results were produced, addressing the concerns evoked in chapter 2.

Finally, a method to perform the style transfer while correctly inserting the face in the input image in place of that of the style painting, was provided in chapter 5. Satisfying results were obtained by combining the *"Deep painterly harmonization"* [14] pipeline – itself inspired by the techniques described in chapters 2 and 3 – with the method tailored to head portraits of chapter 4. The obtained results were shown to improve on previous examples. the jump in quality can be attributed to the two step process introduced in Deep painterly Harmonization, with a first phase designed to generate a robust, coarse image and a second phase refining the result.

Combined with a face segmentation algorithm [17], this last contribution to the thesis provides a sound and convincing technique to seamlessly insert the faces of individuals in portraits, generating new artworks in the process with a minimal amount of effort required from the end user.

6.1 Avenues for future work

As briefly mentioned in chapter 5, the main limitations of this work come from the constraints in memory (the patch matching process used in the solution proved to be memory intensive), which limits the ability to generate new artworks with high fidelity. I believe this problem could be solved by introducing parallelism to the presented style loss computation, delegating the task of matching activations patches to multiple computing resources.

The second limiting factor is time: the proposed solution is far from being real-time capable. This could however change by combining it with a Fast Style Transfer framework, such as the one proposed by Johnson et al. in the paper *"Perceptual losses for real-time style transfer and super-resolution"*. Such a solution could, at the cost of training a feedforward network for set of styles, enable the mass production of new artworks from existing paintings with ease, providing a handy tool for interested artists.

Finally, one could imagine extensions catering to other specific situations such as the style transfer between two photographs, a use case which was briefly discussed in chapter 4.

Bibliography

[1] Tadas Baltrusaitis, Peter Robinson, and Louis-Philippe Morency. Openface: An open source facial behavior analysis toolkit. *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–10, 2016.

[2] Fred Bertsch. Multistyle pastiche generator. 2016.

[3] Alex J. Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. *CoRR*, abs/1603.01768, 2016.

[4] Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. *CoRR*, abs/1612.04337, 2016.

[5] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1033–, 1999.

[6] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[7] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13(1):723–773, March 2012.

[8] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 327–340, New York, NY, USA, 2001. ACM.

[9] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *arXiv preprint arXiv:1705.04058*, 2017.

[10] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.

[11] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. *CoRR*, abs/1601.04589, 2016.

[12] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Demystifying neural style transfer. *CoRR*, abs/1701.01036, 2017.

[13] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *CoRR*, abs/1705.01088, 2017.

[14] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep painterly harmonization. *CoRR*, abs/1804.03189, 2018.

[15] Wittenbrink Ma, Correll. The chicago face database: A free stimulus set of faces and norming data. 47, 1122-1135, 2015.

[16] S. Milborrow and F. Nicolls. Active Shape Models with SIFT Descriptors and MARS. *VISAPP*, 2014.

[17] Yuval Nirkin, Iacopo Masi, Anh Tuan Tran, Tal Hassner, Medioni, and Gérard Medioni. On face segmentation, face swapping, and face perception. In *IEEE Conference on Automatic Face and Gesture Recognition*, 2018.

[18] Alissa Quek. Face morpher library - tool to warp, morph and average human faces.

[19] Ahmed Selim, Mohamed Elgharib, and Linda Doyle. Painting style transfer for head portraits using convolutional neural networks. *ACM Trans. Graph.*, 35(4):129:1–129:18, July 2016. URL: http://doi.acm.org/10.1145/2897824.2925968.

[20] YiChang Shih, Sylvain Paris, Connelly Barnes, William T. Freeman, and Frédo Durand. Style transfer for headshot portraits. *ACM Trans. Graph.*, 33(4):148:1–148:14, July 2014.

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[22] Gray K. Chituc V. Heffner J. Schein C. Strohminger, N. and T.B. (in press) Heagins. The mr2: A multi-racial mega-resolution database of facial stimuli. *Behavior Research Methods*.

[23] Huihuang Zhao, Paul L. Rosin, and Yu-Kun Lai. Automatic semantic style transfer using deep convolutional neural networks and soft masks. *CoRR*, abs/1708.09641, 2017.