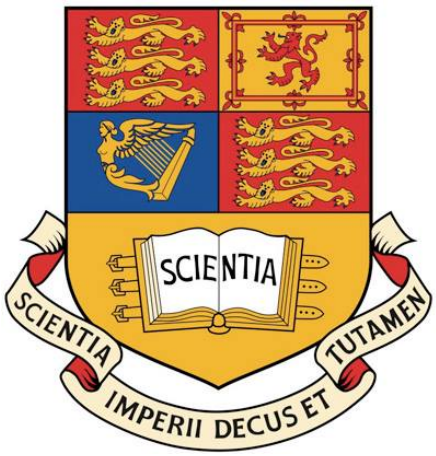
Imperial College London

Department of Electrical and Electronic Engineering

MEng Final Year Project Report



Project Title:	**Hybrid System for Controlled Style Transfer**
Student:	**Cao An Lê**
CID:	**01122387**
Course:	**EEE4T**
Project Supervisor:	**Prof. Pier L . Dragotti**
Second Marker:	**Dr. T. Stathaki**

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my project supervisor Professor Pier Luigi Dragotti, for giving me the opportunity to work this exciting project, and for his steady guidance and invaluable help at every stage of the project.
I would also like to thank my friends and family for their unconditional love and support throughout my entire academic journey, and for being my greatest source of motivation and inspiration all along.

Abstract

Recent advances in Deep Learning and their applications in Computer Vision generated a renewed interest in the general image processing task of artistic style transfer. The aim of this project is to study and analyse existing methods for style transfer, and to propose a novel algorithm that merges those different ideas. First, a detailed background section provides a literature review and some theory on the topic, then an implementation and evaluation section are laid out to discuss design choices, and assess the quality of the work undertaken. Finally some extra considerations and suggested future work are provided at the end of the report.

Contents

List of Figures

1. Introduction

Style transfer is defined as the image processing task of migrating the style of a given image[1] to the content of another[2], thereby creating an artistic combination of the two (see Figure 1.1 for examples of style transfer). Over the years, many algorithms aiming at accomplishing this goal have been proposed, all of which are based on different principles. However, generally speaking, the literature offers two main strategies.

Figure 1.1: An example of style transfer from [1].

A) Content image B) Output using The Shipwreck of the Minotaur by J.M.W. Turner. C) Output using The Starry Night by Vincent van Gogh. D) Output using Der Schrei by Edvard Munch.

Firstly, we find the generalisation of classic texture synthesis methods, which as the name suggests, consists in synthesising a texture image that resembles the style image's texture. The term *generalisation* denotes the application of this technique in the context of style transfer, where the texture synthesis is constrained to the overall shape of the content image.
The second common strategy makes use of the increasingly popular Convolutional Neural Networks (CNNs), and more specifically, CNNs trained for object recognition [3]. These networks are able to extract high-level image information in generic feature representations and use them to independently model the style and content of natural images.
In this work, we put forward the strengths and weaknesses of each method. The final aim is to propose a novel solution that groups the best aspects of both strategies, allowing for an output image with high perceptual quality, as well as a transfer process that is easily tunable in terms of perceptual factors. The success of the task is based on custom quality factors defined later in section 6.

[1]The style image
[2]The content image

2. Background

This section of the report details some necessary background to the project and related work on artistic style transfer. Moreover, specific examples of the two methods mentioned in the introduction are studied in greater depth.

2.1 Motivation

Editing photographs and pictures is becoming an increasingly popular and accessible activity for the general public with the rise of smartphone ownership and usage [4]. From applying coloured filters or fixing colour balance to editing the actual content of the photograph, editing options are becoming numerous and the image processing research sector is constantly trying to find new processing techniques.
Recently, there has been a renewed interest in the task of artistic style transfer, thanks to the impressive technological advances in Deep Learning. In fact, some mobile applications [5] have already been developed to serve that purpose and are already available for iOS and Android devices. However, all of them greatly lack means of controlling the outcome in order to satisfy end-user's taste, and merely offer to input two images to produce a third one, with no additional option.
Ultimately, an automated style transfer system with greater parameter granularity would be benificial for both commercial use (giving users more freedom in producing artistic images), as well as for research purposes (expanding the ranges of tests and results that are possible with only two images), and thus the incentive of this study is to develop an improved solution,with the aforementioned goal in mind.

2.2 Related Work

The early work related to the topic of artistic style transfer mainly consisted of non-parametric algorithms to resample pixels of a given texture image and as such, was called texture synthesis. A paper by Efros and Freeman [6] suggested a patch-quilting approach to this problem, which after generalisation, resulted more in a texture transfer rather than style transfer. Indeed, the transfer was only designed to reproduce the general shades of the given content image. This was then improved by Freeman, with a modification of the patch-merging procedure using a Belief Propagation (BP) approach, which seeks the best patch assignments for a given location among some neighbours.

These works inspired many others on the problem of texture synthesis and led to one particularly successful algorithm developed by Kwatra et. al [7]. Their technique relied on the same principles as in [6] regarding the patch-matching and their aggregation, but differed in the sense that they were attempting to optimise a global minimum rather than local one. Inspired

by the EM (K-means clustering) algorithm, this algorithm is split in two steps. In the first step, the current global image is frozen and the best patch-matches are sought, while in the second stage, the found matches are frozen and the current global image is updated through an aggregation of the patches. They found that applying this synthesis sequentially for different patch sizes, and iterating the EM algorithm multiple times per patch boosted the quality of the resulting image. Morevoer, they also suggested to operate on a Gaussian pyramid [1] of the output image for best output quality.

More recently, Frigo et al. [8] published a paper studying style transfer via texture synthesis which could be considered as a direct follow-up of the work in [6], adapted to include the influence of a content image. However, in contrast, Frigo et al. adopted an adaptive quad-tree division of the image domain by seeking to minimise the distance to the nearest neighbour in the style image and constraining the inner variance of the content patch. Although this work resulted in much better style transfers than those in [6], it still offered poor hallucination capabilities, which again resulted more in a texture transfer rather than an artistic style transfer.

Finally, Michael Elad and Peyman Milanfar [9] heavily based their work on [7] and brought slight modifications and additional elements to the algorithm. The underlying assumption behind their work is that style transfer can be obtained using any good texture syntehsis algorithm and modifying it to take into account the content image. Very interestingly, their style transfer algorithm run over a flat (no content) image simply reduced to a pure texture synthesis process. The key modifications to [7] at the initialisation part are:

- Optionally apply edge-preserving spatial filtering on the content image, so as to remove small details from the final result.
- Pre-process colour transfer from the style image to the content image.
- Build and use a mask image to define importance of the content image regions.
- Initialise the algorithm with the content image added with strong white noise.

Within the iterative part, the modifications are:

- The content image is pushed to the result using a mask for each aggregation step.
- Colour transfer is applied from the style to the result in order to preserve the palette wealth of the style image.
- The Nearest-Neighbour algorithm is paired with a dimensionality reduction technique called Principal Component Analysis and a tree-based search.

A detailed description of their algorithm is provided in section 2.3.1 of this report.

[1] Type of multi-scale signal representation.

Although being quite a recent approach, a substantial amount of work has also been accomplished using CNNs with regards to the style transfer problem. Indeed, late the boom of artificial neural networks gave rise to many exciting research topics, one of which concerned style transfer.

In 2016, Gatys et al. published a paper [1] demonstrating the feasibility of texture synthesis using the VGG-19 network [3], a CNN trained for object recognition. The correlations between feature responses at each layer of the network were computed to obtain a stationary drescription of the image's style. Gradient descent was then used on a white noise image seeking for an output image with the same correlations as the style image. As opposed to classic methods of texture synthesis, this innovative technique allowed for hallucinative components to appear in the output image, which earned Gatys et al.'s work great attention in the texture synthesis research community.

A year later, Gatys et. al again proposed an extension of their work by generalising the algorithm to the context of artistic style transfer [2]. The theory behind this work is very much in the spirit of the previous one, with the CNN also extracting features from a content image. As later described in this study, the features collected for content extraction are those found in the deeper layers of the network. Despite CNNs being incredibly impressive at handling the task of artistic style transfer, the main drawback of using them is speed. The computations that make up the network can be extensively heavy, and grow exponentially with the image dimensions. Recently, a number of works demonstrated the effective use of training a feed-forward CNN to perform stylisation [10, 11, 12] by considerably reducing the time taken.

Finally, a third paper by Gatys et. al [13] proposed a method to allow for more control and flexbility in the generation of the artistic image. Allowed regions on the content image for the style transfer to happen could be defined, colour palettes could be controlled etc... The control of these so-called "perceptual factors" is inevitably crucial for the purpose of a successful style transfer, since the requirements of such a task are not technically clear. Naturally, this paper is of great interest for the development of our proposed solution.

2.3 Background Theory

2.3.1 Patch Matching

This section details the algorithm used in the paper by Elad and Milanfar [9].

Energy Minimisation Problem

In the work of Michael Elad and Payman Milanfar, the core objective of the algorithm is to find the image $\underline{X}$ that would minimise the following series of energy functionals:

$$E_{L,n}\{\underline{X}\} = \frac{1}{c}\sum_{(i,j)\in\Omega_{L,n}} \min_{(k,l)} \left\|\mathbf{R}_{ij}^{n}\underline{X} - \mathbf{Q}_{kl}^{n}\mathbf{D}_{L}^{S}\underline{S}\right\|_{2}^{r} + \left\|\mathbf{D}_{L}^{C}\underline{C} - \underline{X}\right\|_{\mathbf{W}}^{2} + \lambda\rho\{\underline{X}\} \tag{2.1}$$

where $\underline{C} \in \mathbf{R}^{3N_c}$ is the content image, $\underline{S} \in \mathbf{R}^{3N_s}$ is the style image [2] , and $\mathbf{W} \in [0,\infty)^{N_c}$ is a segmentation mask for the content image. The operators $\mathbf{D}_L^C$ and $\mathbf{D}_L^S$ are there to down-scale $\underline{X}$ and $\underline{C}$ in order to operate on the Gaussian-pyramid. Assuming for now that only the native resolution is used, i.e. $\mathbf{D}_1^C = \mathbf{D}_1^S = \mathbf{I}$, the problem reduces to:

$$E\{X\} = \frac{1}{c}\sum_{(i,j)\in\Omega} \min_{(k,l)} \left\|\mathbf{R}_{ij}^{n}\underline{X} - \mathbf{Q}_{kl}^{n}\underline{S}\right\|_{2}^{r} + \|\underline{X} - \underline{C}\|_{\mathbf{W}}^{2} + \lambda\rho\{\underline{X}\} \tag{2.2}$$

The patch extraction of size n from location (i,j) of $\underline{X}$ is denoted by $\mathbf{R}_{ij}^{n}$, and therefore the first term seeks $\underline{X}$ such that all patch of size n extracted at (i,j) is close to a patch of the same size extracted from the style image $\underline{S}$, represented by $\mathbf{Q}_{kl}^{n}$. Note that the matchings are executed over a decimated grid of points to reduce computation, and that local patch errors use a power of $r = 0.8$ instead of a squared term to get a robuts patch aggregation over the overlaps. The two hyper parameters n and L denote respectively the different patch sizes and different scales to be used. This essentially ensures that element of different sizes from the style image are included in the style transfer. The second term takes care of applying the segmentation map to the content image, and the third term ensures that $\underline{X}$ is spatially smooth. The energy functional is minimised sequentially by iterating through the scales and patch-sizes.

EM Optimisation Algorithm

In this section, only one patch-size and the native resolution will be used for simplicity:

$$\min_{\underline{X}} \frac{1}{c}\sum_{(i,j)\in\Omega} \min_{(k,l)} \left\|\mathbf{R}_{ij}^{n}\underline{X} - \mathbf{Q}_{kl}^{n}\underline{S}\right\|_{2}^{r} + \|\underline{X} - \underline{C}\|^{2}\mathbf{w} + \lambda\rho\{\underline{X}\} \tag{2.3}$$

[2] N_c and N_s are the size (number of pixels) of each image. The factor 3 describes the images in the RGB domain.

We first assume that $\underline{X}$ is fixed and known, and look for the best patch matches in $\underline{S}$. In other words, we look for the closest neighbour at (k^*, l^*) of $\mathbf{R}_{ij}^n \underline{X}$ in $\underline{S}$:

$$\{k^*, l^*\} = \min_{(k,l)} \left\| \mathbf{R}_{ij}^n \underline{X} - \mathbf{Q}_{kl}^n \underline{S} \right\|_2 \tag{2.4}$$

which yields the matched patch $\underline{z}_{ij} = \mathbf{Q}_{k^*,l^*}^n \underline{S}$. In the next step, those found patches are frozen and $\underline{X}$ is updated by solving:

$$\min_{\underline{X}} \frac{1}{c} \sum_{(i,j)\in\Omega} \left\| \mathbf{R}_{ij}^n \underline{X} - \underline{z}_{ij} \right\|_2^r + \|\underline{X} - \underline{C}\|_{\mathbf{W}}^2 + \lambda\rho\{\underline{X}\} \tag{2.5}$$

This problem can be re-written by adding a constraint.

$$\min_{\underline{X},\underline{Y}} \frac{1}{c} \sum_{(i,j)\in\Omega} \left\| \mathbf{R}_{ij}^n \underline{X} - \underline{z}_{ij} \right\|_2^r + \|\underline{X} - \underline{C}\|_{\mathbf{w}}^2 + \lambda\rho\{\underline{Y}\} \text{ s.t. } \underline{Y} = \underline{X} \tag{2.6}$$

The Augmented-Lagrangian algorithm can be used to handle the constraint:

$$\min_{\underline{X},\underline{Y}} \frac{1}{c} \sum_{(i,j)\in\Omega} \left\| \mathbf{R}_{ij}^n \underline{X} - \underline{z}_{ij} \right\|_2^r + \|\underline{X} - \underline{C}\|_{\mathbf{W}}^2 + \lambda\rho\{\underline{Y}\} + \mu\|\underline{X} - \underline{Y} + \underline{U}\|_2^2 \tag{2.7}$$

where $\underline{U}$ is the Langrange multiplier vector. Then, using the Alternating Direction Method of Multipliers (ADMM), we update $\underline{X}$, $\underline{Y}$ and $\underline{U}$ sequentially with block-coordinate descent [14], in which we sequentially freeze some of the unknowns and update the others. In the first step, we update $\underline{X}$ while keeping $\underline{Y}$ and $\underline{U}$ as known.

$$\min_{\underline{X}} \frac{1}{c} \sum_{(i,j)\in\Omega} \left\| \mathbf{R}_{ij}^n \underline{X} - \underline{z}_{ij} \right\|_2^r + \|\underline{X} - \underline{C}\|_{\mathbf{W}}^2 + \mu\|\underline{X} - \underline{Y} + \underline{U}\|_2^2 \tag{2.8}$$

To solve this, the Iterative Reweight Least Squares (IRLS) [15] algorithm is used, and we modify the r power to 2 with the help of pseudo-weights. Considering the k^{th} iteration, there is a temporary solution $\hat{\boldsymbol{X}}_k$ and the weights are $w_{ij} = \left\| \mathbf{R}_{ij}^n \hat{X}_k - \underline{z}_{ij} \right\|_2^{r-2}$. The problem becomes:

$$\min_{\underline{X}} \frac{1}{c} \sum_{(i,j)\in\Omega} w_{ij} \left\| \mathbf{R}_{ij}^n \underline{X} - z_{ij} \right\|_2^2 + \|\underline{X} - \underline{C}\|_{\mathbf{W}}^2 + \mu\|\underline{X} - \underline{Y} + \underline{U}\|_2^2 \tag{2.9}$$

and has a closed form solution,

$$\underline{\hat{X}}_{k+1} = \left[\frac{1}{c} \sum_{(i,j)\in\Omega} w_{ij} \left(\mathbf{R}_{ij}^n\right)^T \left(\mathbf{R}_{ij}^n\right) + \mathbf{W} + \mu\mathbf{I} \right]^{-1} . \left[\frac{1}{c} \sum_{(i,j)\in\Omega} w_{ij} \left(\mathbf{R}_{ij}^n\right)^T z_{ij} + \mathbf{W}\underline{C} + \mu(\underline{Y} - \underline{U}) \right] \tag{2.10}$$

Ignoring the Langrangian multiplier μ, this result translates to the aggregation of the patches $\underline{z}_{ij}$ in the overall image weighted by w_{ij}, followed by applying the effects of the content image

mask $\mathbf{W}$. In the second step, we update $\underline{Y}$ and freeze $\underline{X}$ and $\underline{U}$:

$$\min_{\underline{Y}} \lambda\rho\{\underline{Y}\} + \mu\|\underline{X} - \underline{Y} + \underline{U}\|_2^2 \tag{2.11}$$

This corresponds to a denoising procedure on the image $\underline{X} + \underline{U}$ with noise λ/μ. The denoising algorithm used by Elad and Milanfar is the Domain-Transform filtering [16], which is a fast approximation algorithm for the bilateral filter.
In the third and last step, $\underline{U}$ is updated using the Augmented Langrandian approach: $\underline{U} = \underline{U} + \underline{X} - \underline{Y}$. By ommitting the prior ρ, we obtain the optimisation:

$$\min_{\underline{X}} \frac{1}{c} \sum_{(i,j)\in\Omega} \left\|\mathbf{R}_{ij}^n \underline{X} - \underline{z}_{ij}\right\|_2^r + \|\underline{X} - \underline{C}\|_{\mathbf{w}}^2 \tag{2.12}$$

which can be broken into two steps. We first minimise the first term (recall the IRLS method):

$$\underline{\tilde{X}} = \text{Argmin}_{\underline{X}} \sum_{(i,j)\in\Omega} \left\|\mathbf{R}_{ij}^n \underline{X} - \underline{z}_{ij}\right\|_2^r \tag{2.13}$$

and use the temporary result $\underline{\tilde{X}}$ to approximate the general solution $\underline{\hat{X}}$:

$$\underline{\hat{X}} = (\mathbf{W} + \mathbf{I})^{-1}(\underline{\tilde{X}} + \mathbf{W}\underline{C}) \tag{2.14}$$

which happens to be exact when $r = 2$. Finally, we can bring back the prior as a denoising step on $\underline{\hat{X}}$.

Multi-Scale and Multi-Patch Sizes

As mentionned earlier, the above algorithm is applied for each image scale and patch size interatively, starting from the coarsest resolution to the native one and for each resolution iterating from the largest patch size to the smallest one. The essence of this approach is very similar to that of a stochastic gradient descent, as each n and L pair brings its own contribution to the end solution, which helps avoiding local minima. Then for each pair (L, n), 10 inner iterations are performed to solve the IRLS problem, where the optimisation task is initialised with the content image with very strong additive Gaussian noise ($\sigma = 50$), so that patches are matched daringly.
This procedure leads to a gradually refined style-transfer task. Indeed, spatial resolution is improved as the algorithm sweeps across the resolution scales from coarest to finest. Secondly, initial large elements from the style image are gradually refined as the patch sizes become smaller. Lastly, elements from the content image are added to the temporary result at each iteration, guided by the segmentation mask $\mathbf{W}$. This allows control over which regions are allowed to drift and hallucinate more than others.

Segmentation

The role of the segmentation mask $\mathbf{W}$ is to give more or less importance to certain regions of the content image. Several methods of creating such a mask exist, however, the results presented in Elad and Milanfar's work were all obtained using $\mathbf{W}$ to be a constant image of value $0 < \alpha < \infty$. In other words, no real spatial guidance was provided in their algorithm, and the style transfer was operated on the entirety of the content image. Quite predictibly, the value of α influences the strength of the style transfer, and in the extreme cases, we perform pure texture synthesis ($\alpha = 0$) or no style transfer at all ($\alpha \to \infty$).

Colour Transfer

The above algorithm can be adapted to output any colour palette. One crucial step is to pre-process the content and style image $\underline{C}$ and $\underline{S}$ such that they share the same palette. In general, experiments showed that the style image's pallette was much richer than that of the content image, and led to more visually pleasing results. The process of palette matching is also very important during the sequential process of the optimisation, so as to use as much of the richness of $\underline{S}$ as possible.
In Elad and Milanfar's work, the task of palette matching was achieved through histogram matching. The few cases where this technique failed used style images which either were not rich enough or contained large dark regions. This statement suggests that the choice of $\underline{S}$ is crucial to the success of the style transfer, and this represents a direct consequence their algorithm as each element within $\underline{S}$ *can* be considered as a potential styling element (since no spatial guidance segmentation mask $\mathbf{W}$ has been considered).

Fast Nearest Neighbour

Patch-matching constitute the most computer-intensive task. This section describes an approximation of the algorithm presented above, trading an acceptable amount of result quality to achieve better computational performance. In this approximation, for a each (n, L) pair, patches of $\underline{S}$ are arranged in a matrix $\mathbf{P}_{L,n}$ with dimensions $n \times M_{L,n}$, where $M_{L,n}$ represents all possible patches. Then the aim to find the closest column vector in $\mathbf{P}_{L,n}$ to $x_{ij} = \mathbf{R}_{ij}^{n}\underline{X}$.
Before doing so, a dimensionality reduction of $\mathbf{P}$ can be applied using Principal Component Analysis (PCA)[3], which allows for a much faster search of the Nearest Neighbour (NN). Additionally, the NN search can also be quickened and approximated by constructing a clustering tree.

[3]The paper suggested keeping 95% of the data variation.

Summary of Elad and Milanfar's Algorithm

Figure 2.1, directly borrowed from [9], summarises the above algorithm.

Objective: Create the style-transfer image, $\underline{X}$.

Input: Use the following ingredients and parameters:

- $\underline{C}$ and $\underline{S}$ - Content and Style images
- $\mathbf{W}$ - Content segmentation map
- L_{max} - Number of resolution layers
- $n_1, n_2, \ \dots \ , n_m$ - patch sizes
- $d_1, d_2, \ \dots \ , d_m$ - Ω subsampling gaps
- I_{IRLS} - number of IRLS iterations (chosen as 10)
- I_{alg} - number of update iterations per patch-size
- r - robust statistics value to use.

Initialization:

- Apply color transfer from $\underline{S}$ to $\underline{C}$
- Build the Gaussian pyramids of $\underline{C}$, $\underline{S}$, and $\mathbf{W}$
- Optional (for fast and approximate NN): Prepare the patches from the style image for each resolution layer and each patch size as tree structure
- Initialize $\underline{X} = \mathbf{D}_L^C \underline{C} + V$, where $V \sim \mathcal{N}(0, 50)$

Loop Over Scales: For $L = L_{max}, \ \dots \ , 1$ do:

Loop Over Patch-Sizes: For $n = n_1, \ \dots \ , n_m$, minimize $E_{L,n}$ w.r.t.$\underline{X}$, by

Iterate: For $k = 1, 2 \ \dots \ , I_{alg}$ do:

1. *Patch Matching*: For the current image $\underline{X}$, match NN from the style image by solving (16). The style patches should be of the same size, taken from the corresponding resolution layer.
2. *Robust Aggregation*: Compute $\underline{\tilde{X}}$ as the robust patch aggregation result, by solving (14) using I_{IRLS} iterations.
3. *Content Fusion*: Combine the content image $\mathbf{D}_L^C \underline{C}$ to $\underline{\tilde{X}}$ using Equation (15) to obtain the updated $\underline{X}$.
4. *Color Transfer*: Apply color-transfer from $\underline{S}$ to $\underline{X}$.
5. *Denoise*: Filter the obtained $\underline{X}$ by the domain-transform.

Scale-Up: As we move from one resolution layer to the next, scale-up $\underline{X}$.

Result: The output of the above algorithm is $\underline{X}$.

Figure 2.1: Elad and Milanfar's Algorithm

2.3.2 Convolutional Neural Networks

In this second section, the background theory regarding the use of CNNs for artistic style transfer is explained, with a stronger emphasis on the work of Gatys et al [2]. The recent progress in deep learning allowed for very powerful computer vision systems that are able to extract high-level semantic information from natural images in generic feature representations that generalise across datasets. Note that in the rest of this section, it is assumed that the VGG network [3] is considered.

Content Representation

Generally, a layer in a network represents a non-linear filter bank. The responses of a layer l can be arranged in a matrix $F^l \in \mathcal{R}^{N_l \times M_l}$ where N_l is the number of filters at l, M_l is the size of the feature maps and F^l_{ij} is the response of filter i at position j in layer l. In order to see the image content stored at a certain layer of the network, gradient descent can be applied on a white noise, to seek an image with similar feature responses as the original one. Considering the original image $\vec{p}$, the generated image $\vec{x}$ and their respective feature representation P^l and F^l in layer l, we can define the squared-error loss:

$$\mathcal{L}_{\text{ content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F^l_{ij} - P^l_{ij}\right)^2 \tag{2.15}$$

Using its derivative with respect to the activations from layer l,

$$\frac{\partial \mathcal{L}_{\text{ content}}}{\partial F^l_{ij}} = \begin{cases} \left(F^l - P^l\right)_{ij} & \text{if } F^l_{ij} > 0 \\ 0 & \text{if } F^l_{ij} < 0 \end{cases} \tag{2.16}$$

the gradient descent with respect to $\vec{x}$ can be computed using error back-propagation. The process can be repeated until the search image $\vec{x}$ generates a similar response in a given layer as the initial image $\vec{p}$.
In general, CNNs trained for object recognition contain more explicit object representations in higher layers[4] of the network [1], i.e. higher layers extract higher-level content information. Therefore, the content representation model is formed from the feature responses in higher layers of the network.

Style Representation

To find a model for the style of an image, the feature space used is slightly different as it consists of the correlations between the different filter responses, which is what capture texture information instead. Those correlations are computed from the Gram matrix $G^l \in \mathcal{R}^{N_l \times N_l}$ and so G^l_{ij} represents the inner product between the vectorised feature maps i and j in layer l:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \tag{2.17}$$

[4]Deeper layers.

For texture information, many layers are used in the correlation calculation, as it conveniently yields a stationary representation of the input image's style. Similarly, an image can be constructed from gradient descent on a white noise image, seeking to minimise the loss between the correlations of the original image and those of the generated image [1, 17]. If we define the original image $\vec{a}$, the generated image $\vec{x}$ and their respective style representation in layer l A^l and G^l, the loss at l is

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2 \tag{2.18}$$

and the style loss is defined as

$$\mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l \tag{2.19}$$

where w_l determines weights for the loss contribution of each layer. Again, the derivative with respect to the activations in layer l can be calculated:

$$\frac{\partial E_l}{\partial F_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2} \left(\left(F^l\right)^{\mathrm{T}} \left(G^l - A^l\right)\right)_{ji} & \text{if } F_{ij}^l > 0 \\ 0 & \text{if } F_{ij}^l < 0 \end{cases} \tag{2.20}$$

Finally, back-propagation can be used to calculate the gradients of E_l with respect to the pixels of $\vec{x}$.

Style Transfer

Transfering the style of an artwork $\vec{a}$ onto a photograph $\vec{p}$ constitutes a joint minimisation problem, with the following loss function:

$$\mathcal{L}_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{\text{content}}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) \tag{2.21}$$

which is a linear combination of the individual loss functions for content and style. Depending on the ratio of α and β, visual quality of the resulting image is subject to variations, however this matter remains a subjective topic. Thus, the gradient $\frac{\partial \mathcal{L}_{\text{total}}}{\partial \vec{x}}$ paired with the L-BFGS [18] optimisation algorithm can be used to find the right image that combines that style of the artwork the content of the photograph.
Figure 2.2 is directly taken from [2], and presents schematically the algorithm proposed by Gatys et. al on the task of style transfer using CNNs. The two networks working in parallel are clearly shown.

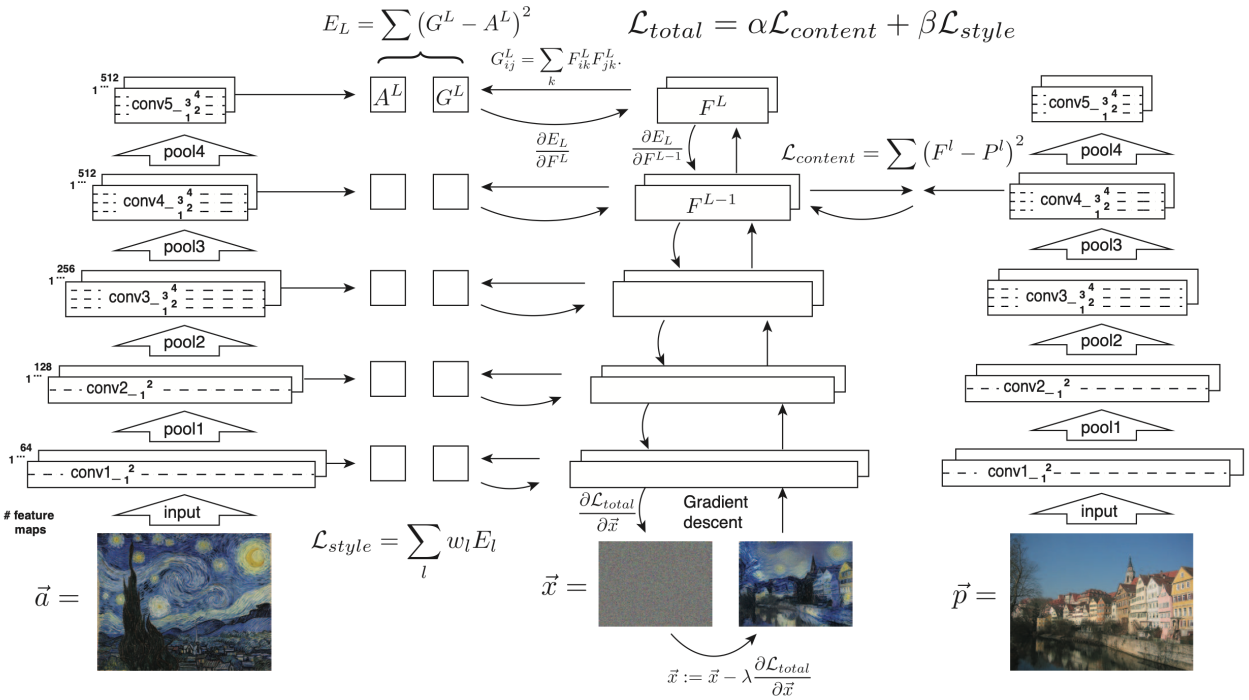


Figure 2.2: Gatys et. al's Algorithm, taken from [2]

2.3.3 Comparison between the two methods

The two methods presented above are fundamentally different. Principally, the non-parametric resampling techniques achieve style transfer on a low-level while the CNN-based methods define a high-level model for both the style image and the content image. Rather than defining a model, resampling methods give instead a mechanistic procedure to randomise a source texture without changing its perceptual properties.

A second difference lies in the randomness of the outcome. Indeed, the texture synthesis-based methods allow for a more predictable and controllable output of the style transfer task, as all the parameters are defined during the process. In contrast, resulting image from CNNs are rather unpredictable, which is a typical scenario for neural network-based solutions, as they often rely on probabilities instead. However, one should note that work has been done on the subject of control of perceptual factors [13] within CNN-based algorithms, and this has been subject to greater investigation in this study. Moreover, results have shown that this property of CNNs is not necessarily undesirable, as the task of producing hallucinative elements is inherently more effective using neural networks, which could be considered as more artistically pleasing. Indeed, since the resampling algorithms are heavily based on patch-matching, stylistic elements in the resulting image will always be taken directly from the style image, rather than be artistically "created".

Lastly, the patch-matching techniques have proven to be considerably faster at producing the hybrid image. Again, work has been done to accelerate the process for CNNs using feed-forward

networks [10, 11, 12], but they still remain slower.
In creating the proposed solution of this study, both methods and their individual properties were considered. Some techniques were borrowed and combined together in order to produce a self-contained system which achieves visually pleasing result whilst offering a good amount of flexibility in the process.

3. Analysis and Design

As mentioned in the introduction section, the end goal of this project is to propose an improved system for the problem of style transfer by combining different ideas from existing ones, and especially those from [2, 13, 9]. The new algorithm should group the strengths of other methods and therefore should ideally allow to produce images of higher perceptual quality, with a higher control over the perceptual factors. The following section details the design choices and expands on the operating principle of individual features that consistutes the final system.

3.1 Base Algorithm

In terms of the backbone of the style transfer mechanism, the decision was directed at the artifical neural network based approach, for two main reasons:

- The first one is that no official software was provided for the texture synthesis paper, which implied a big inconvenience given the time constraints of the project. Indeed, implementing the complex algorithms described in [9] from scratch would not have allowed enough time for work on actual improvement of the system.
- Secondly, as briefly disccused in the background section, the novel CNN-based techniques generally allows for more visually pleasing artistical creations[1], under the assumption that the system accomodates enough flexibility in tuning parameters of the process to better suit the nature of the two given input images.

Therefore, a slightly modified VGG-19 neural model [3] trained for object recognition on the popular ImageNet [19] image dataset was selected to complete the style transfer task. The original network's architecture is represented in figure 3.1.

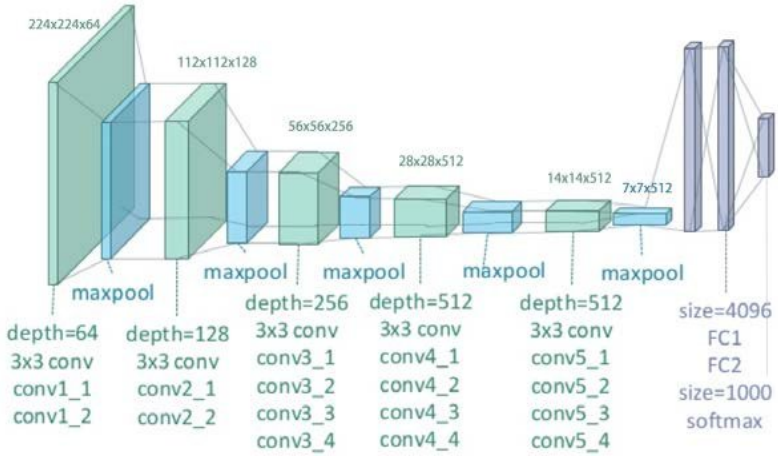


Figure 3.1: Architecture of the VGG-19 Neural Network.

[1] since they produce hallucinative elements in the resulting image.

The network's configuration for the neural style transfer process was set up as follows:

- Optimiser: L-BFGS
- Feature Pooling Type: Average Pooling
- Content Layers: relu4_2
- Style Layers: relu1_1, relu2_1, relu3_1, relu4_1, relu5_1
- Max Iterations: 800

3.2 Tuning parameters

3.2.1 Colour Palette

The choice of colour palette is a feature both described in the non-parametric paper of Elad and Milanfar [9], as well as in the third of paper of Gatys et. al [13], where controllable perceptual factors are put forward. This component of the style transfer mechanism is an important one as it allows the resulting image to possess virtually any colour we desire. In this work, the technique is achieved through histogram matching, implemented as a pre-processing step where the content and style images see their histogram being matched to one which is representative of the desired colours in the output image.

For simplicity, the choice of target histogram in this study was limited to either that of the style image S[2], or that of the content image C, as it made the most sense in the context of this artistic task and was judged to be sufficient as a proof of concept for colour control. In the given scenario of wanting to preserve the content image's colours, the colours in S are transformed to match those in C. This yields a new style image S' that is used as input style image to the neural style transfer algorithm which otherwise remains unchanged.

The specification of a certain histogram for an image can be achieved in two steps. If we consider $p_r(r)$ to be the original probability density function[3], and $p_z(z)$ to be the desired probability density function, the first step involves equalising both histograms so that they are uniformly distributed:

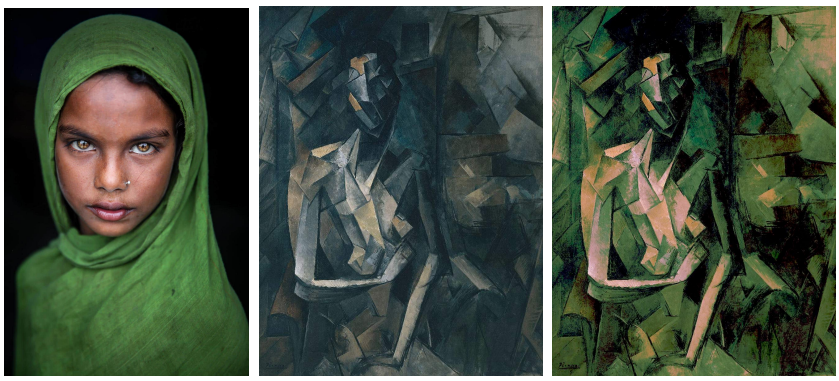
$$s = T(r) = (L-1)\int_0^r p_r(w)dw \tag{3.1}$$

$$v = G(z) = (L-1)\int_0^z p_z(w)dw \tag{3.2}$$

where r lies in the interval $[0, L-1]$. Since the resulting histograms are considered identical, the inverse process $z = G^{-1}(s) = G^{-1}[T(r)]$ can be used to obtain the desired probability density function. While the above example allows the histogram specification of a grayscale image, the same technique can easily be applied separately to the three channels (RGB) of a coloured image.

[2]In the default style transfer case, the output image always inherits the colour palette of the style image.
[3]Image histograms are essentially probability density functions of intensities.

In order to illustrate what a transfer of colour palette achieves through histogram specification, Figure 3.2 and 3.3 shows the result of applying it to two different pairs of images.



(a) Target **(b)** Source **(c)** Matched image

Figure 3.2: Colour Palette Transfer through Histogram Specification eg. 1

A) Photograph: "Mou Aysha" - B) Painting: "Femme Nue Assise"

(a) Target **(b)** Source **(c)** Matched image

Figure 3.3: Colour Palette Transfer through Histogram Specification eg. 2

A) Photograph: "Tubingen" - B) Painting: "The Shipwreck of the Minautor"

3.2.2 Segmentation Mask

Again mentioned in both [9] and [13] but only truly explored in the latter one, a segmentation mask for spatial guidance is useful to dictate where in the content image the transfer is allowed to happen. Likewise, it can also be used on the input style image to define specifically which parts of the artwork are to be used in the transfer process. Therefore, the mask of a given image with dimensions $N \times M$ is defined as another image with the same dimensions whose pixels can take a finite number of values defining a finite number of segments in the segmentation output.

Although Gatys et. al's paper describes how style transfer can be spatially controlled via Guided Gram Matrices, there is no documentation as to how to produce such segmentation masks. Similarly, other demo software implementations of this feature in the community all directly provide ready-made guidance masks alongside their images, with no details on their creation. Moreover, all of those examples masks are binary masks, only defining allowed (white) source regions (in the style image) and allowed destination regions (in the content image).

In this work, spatial control is improved by extending the feature to integrate guidance masks with potentially more than two values. For instance, given input content and style images, and a segmentation mask with N distinct values[4] for each image, the style transfer process will occur exactly where the values of the individual masks are matching, i.e., the style of segmented regions in the style image with values $n_1, n_2, ..., n_N$ are transfered to the regions of the content image that are segmented with values $n_1, n_2, ..., n_N$.
Finally, we create and define our own three methods of segmentation, which leverage this new feature.

Threshold Segmentation

Threshold segmentation constitutes the simplest method of image segmentation, where the goal is to segment the image according to its brightness level. To achieve this, the output space is equally divided into $n_t + 1$ segments where n_t is the number of thresholds to be used. In the most extreme case where $n_t = 1$, and assuming brightness level spans the range $[0, 255]$, the only threshold $t_1 = 128$ splits output space in 2 values (0 where pixels $< t_1$ and 255 where pixels $> t_1$). This simply comes down to a binary type of segmentation and produces a binary mask.
As a summary, our threshold segmentation method involves the following steps:

- Converting the image to grayscale level.
- Equalising the image's histogram.
- Categorize each pixel in the right in the right segment of brightness level.

The initial conversion of the input image to grayscale is essential as only the brightness level of pixels is of interest, and not their colour information. Secondly, an intermediate histogram equalisation step is added in order to span a bigger range of brightness levels as well as obtaining a more equal proportion of each value in the segmented image.
The threshold segmentation technique is shown in Figure 3.4 for $n_t = 1, 2, 3$.

(a) Input image **(b)** $n_t = 1$ **(c)** $n_t = 2$ **(d)** $n_t = 3$

Figure 3.4: Threhsold Segmentation on "Mou Aysha" photograph

[4] N segmented regions.

Colour Segmentation

In this type of segmentation, the intention is to section an image by colour. Similarly as for threshold segmentation, an external parameter n_c is introduced to define the number of colours desired in the segmented image. As opposed to n_t, this directly corresponds to the number of regions in the segmentation mask.

This method is particularly useful for landscape-type of images where relative regions are clearly differentiated by their colour. For instance, separating the green and blue in a photograph of grass terrain and a sky with colour segmentation and parameter $n_c = 2$ is likely to work well.

The approach adopted to perform colour segmentation is described as follows:

- Find the n_c most dominant colours in the image.
- For every pixel in the image, calculate the distance to the n_c most dominant colours.
- Replace that pixel with the colour that had the minimum distance.

In this work's implementation, the n_c most dominant colours are found using the k-means clustering algorithm on the input image (in the 3-dimensional RGB space), from which n_c cluster centers are returned. Then, the standard L2 Euclidean norm is used to calculate the distance from each pixel to the n_c most dominant colours, and the closest colour is defined as the one with the minimum distance. Examples of colour segmentation are provided in Figure 3.5, 3.6, and 3.7.

(a) Input Image **(b)** Segmented Image

Figure 3.5: Colour Segmentation on "Sunflower" photograph ($n_c = 2$)

(a) Input Image **(b)** Segmented Image

Figure 3.6: Colour Segmentation on "Desert" photograph ($n_c = 3$)

(a) Input Image **(b)** Segmented Image

Figure 3.7: Colour Segmentation on "George Moore" painting ($n_c = 4$)

From the above figures, it can clearly be seen how this sort of segmentation can be useful in a style transfer setting, as colour regions can be targeted easily.

Semantic Segmentation

There exist numerous levels of granularity in which machines can acquire an understanding of images.

- *Image classification* - is the most fundamental block in Computer Vision and involves putting a label on a single object in an image.
- *Classification with Localisation* - has the additional task of localising where the object is in the image by defining a bounding box around it.
- *Object Detection* - extends to images containing multiple objects and therefore possibly multiple labels.
- *Semantic segmentation* - assigns a class label to each pixel in a given image, with thus more precise boundaries than simple boxes.
- *Instance segmentation* - is similar to semantic segmentation but also separates the different instances of a same class.

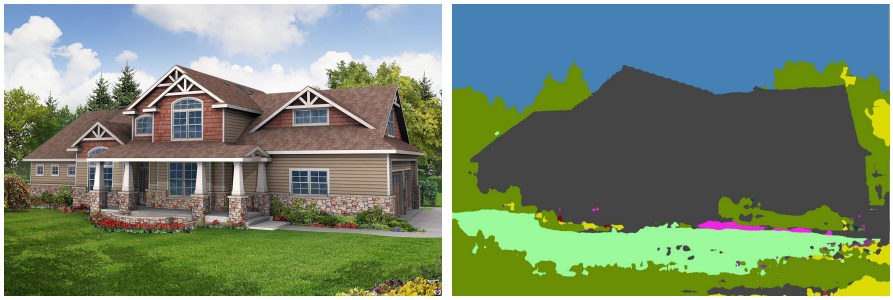
For the purpose of this study, semantic segmentation was explored as the third method of segmentation as it constituted a good fit for the task of artistic style transfer.

Unsurprisingly, semantic segmentation is the achievement of the work of CNNs trained for object recognition. In this implementation, the PSPNet network [20] was used and trained on the famous CityScapes image dataset. The network was pre-trained for exactly 19 labels: *road*, *sidewalk*, *building*, *wall*, *fence*, *pole*, *traffic light*, *traffic sign*, *vegetation*, *terrain*, *sky*, *person*, *rider*, *car*, *truck*, *bus*, *train*, *motocycle*, *bicycle*. The choice of the CityScapes dataset [21] for training was motivated by the fact that style transfer applications are more often paired with

scenery images rather than specific objects[5].

While there has been an attempt at training the PSPNet network with extra labels with custom hand-annotated images in order to widen the model's capability, the number of provided examples was not sufficient, nor was the precision of the labelling masks. The whole study therefore used the initial pre-trained PSPNet model to perform semantic segmentation.

Figures 3.8 and 3.9 show the work of semantic segmentation.



(a) Input Image **(b)** Segmented Image

Figure 3.8: Semantic Segmentation on "House" photograph

(a) Input Image **(b)** Segmented Image

Figure 3.9: Semantic Segmentation on "Road" photograph

3.2.3 Content & Style balance

As previously mentioned in part 2.3.2 in the Background section, Gatys' neural-based algorithm offers a way to manipulate the influence of the content and style images through a linear combination of their loss representation in the global optimisation problem:

$$\mathcal{L}_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{\text{content}}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) \tag{3.3}$$

This feature is likewise exploited in order to offer a greater range a possible artistic style transfer creations.

[5]CityScapes is a dataset containing mainly urban scenery images.

4. Implementation

The implementation of the final system was designed according to three main factors.

- Intuitiveness & flexibility of use.
- Fast computations.
- Easily maintained and modified.

For those previous purposes, the development of a fully self-contained web application (see Figure 4.1) became an evident choice.

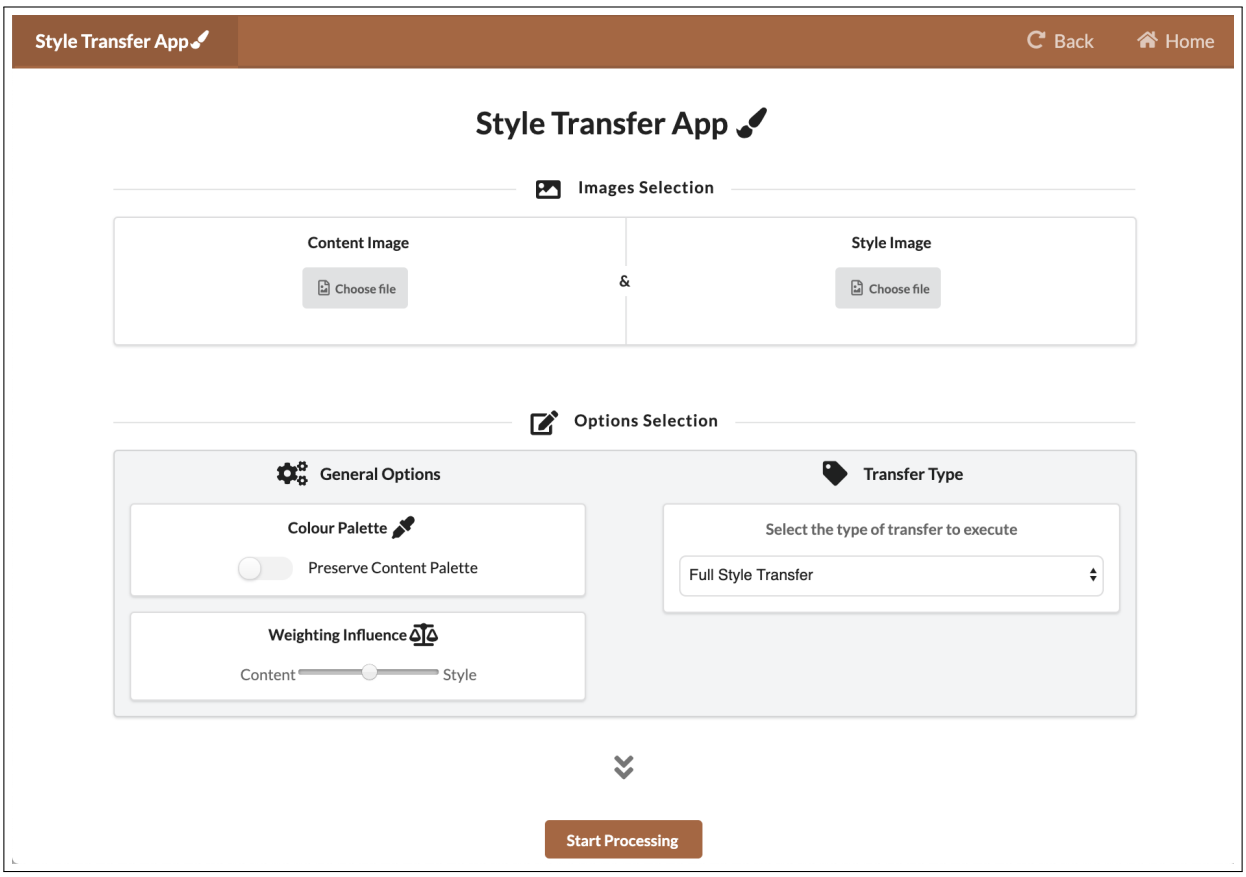


Figure 4.1: Index page of the web application

4.1 Motivations for a Web Application

The following section details the reasons for developing a web application to serve the purpose of this research. Whilst increasing the development time, the final deliverable bear extra benefits which are certainly valuable to the study.

User Experience

A Graphical User Interface (GUI) is always a convenient way to bring life to a program. Indeed, it makes it far more intuitive for anyone (researcher or commercial consumer) to execute the many different type of image operations that the suggested style transfer system offers.

The same cannot be stated about script-based programs that often have to be run from a command-line interface through some specified command. Indeed, an action as simple as choosing input content and style images would require the user to manually type the exact path of both images on their computer. A similar complication can arise when using program parameters. Tweaking them in a script-based application either requires the user to run different versions of the program through different commands, or again manually type in the correct parameters.

On the other hand, a GUI can simply allow to input images through a typical file system window. With regards to tweaking parameters, it can be a matter of ticking checkboxes, or selecting options in a dropdown menu. Furthermore, given the graphical nature of the project, where images play a major role, a GUI conveniently leverages the use of images, as they can be displayed directly on the screen at any stage of the process.

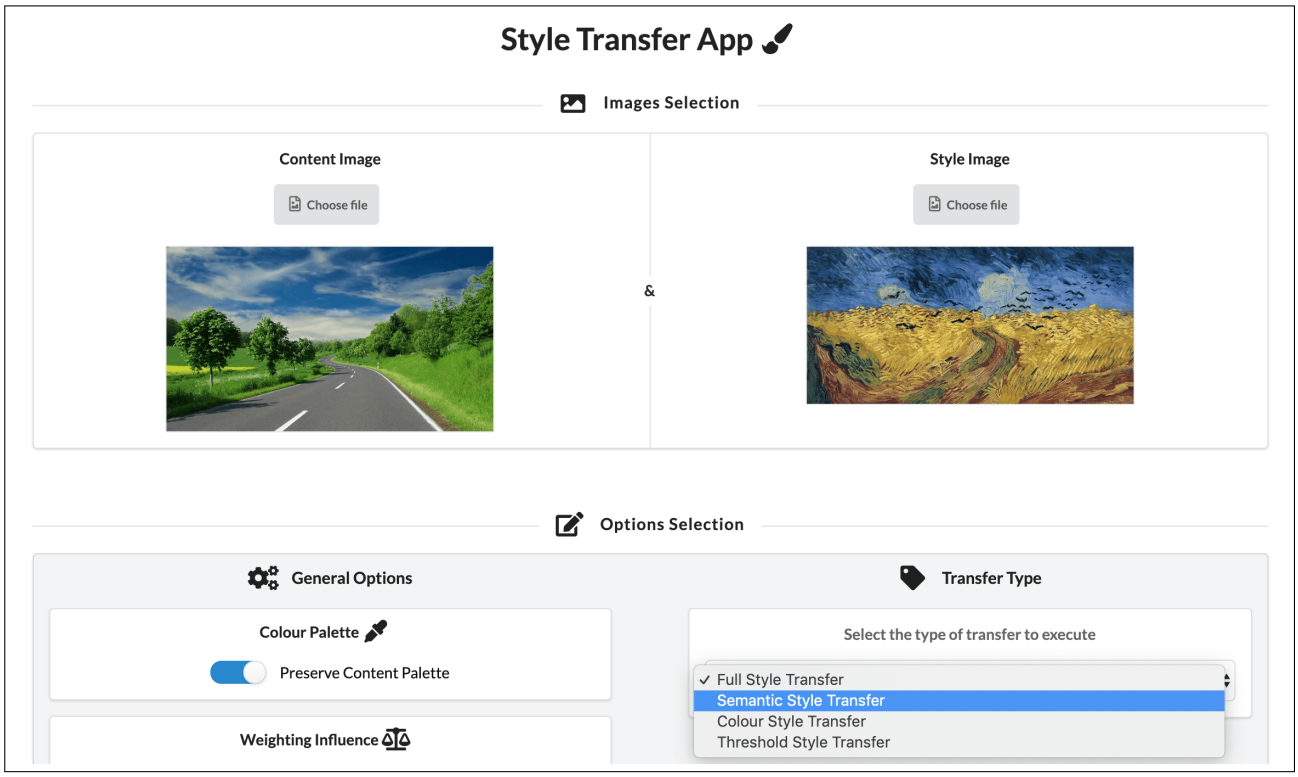


Figure 4.2: Screenshot of the application showing enhanced user experience

Therefore, implementing the system as part of a real web application enhances the experience by making it quicker, easier to use, and visually more appealing, especially for this type of use case.

Computational Performance

Similarly, the speed of the system is of high interest in both commercial settings and in the research field. As a commercial user, speed is highly linked to user experience and quality of the app. In the case of a researcher, it is in their interest to be able to perform tests faster thereby progressing their research more quickly.

Similar to any other process that comprises the training of an artificial neural network, style transfer can take a considerable amount of time to run. Building a web application allowed the program to be easily deployed to a cloud server. As a consequence, the heavy computations that style transfer involves needed not run locally, but rather on a more powerful machine better suited for the task.
For this study, the app was thus deployed to a Google Cloud Compute Engine Virtual Machine instance, with 24 CPUs available and an NVIDIA Tesla V100 GPU. The code was also adapted to account for the use of improved hardware. For example, the code snippet presented in Figure 4.3 represents a part of the colour segmentation process, where individual pixels are replaced with their nearest neighbour amongst the n_c most dominant colours. The Python `multiprocessing` module is used to create a `Pool` of workers that splits the total task to accomplish[1] into `n_cpu` parallel subtasks. In our case, since `n_cpu` = 24, this particular code executes approximately 6 times faster than a typical quad-core personal laptop.

```
import multiprocessing

# Number of CPUs available
n_cpu = multiprocessing.cpu_count()

# Replace every pixel by the nearest neighbor in the dominant colours
with multiprocessing.Pool(processes=n_cpu) as pool:
    output_image = np.array(pool.map(
        self.closest_row,
        [x[0] for x in np.split(image, image.shape[0])]
    ))
```

Figure 4.3: Example of leveraging the multi-core server machine hardware

Similarly, since both tasks of semantic segmentation and style transfer are handled by CNNs, they are mapped very well to the work of GPUs, especially with the GPU support that the `Tensorflow` [22] package for Python offers. A comparison of the performance before and after the deployment can be found in the results and evaluation section.

[1] e.g. for an image with width N and heigh M, this pixel-wise transformation requires $N \times M$ operations.

	Local	Server
Task 1	152.783s	39.776s
Task 2	87.658s	25.852s
Task 3	39,320.127s	285.017s

Table 4.1: Time performances before and after deployment. Task 1: Colour Segmentation of "Starry Night" painting ($n_c = 5$). Task 2: Semantic Segmentation on "House". Task 3: Style Transfer using "Tubingen" as content and "Starry Night" as style.

Maintainability

The structure of the codebase was carefully designed to accomodate transparent and scalable integration of additional features. The web application was developed using the following popular technologies:

- Python (`Flask`, `Tensorflow`, `OpenCV`) for the server-side development.
- HTML, CSS, JavaScript for the client-side development.

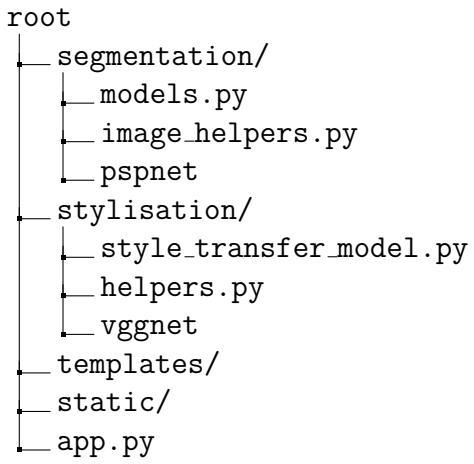


Figure 4.4: Directory Tree Structure of the System. `segmentation/`: directory containing all segmentation related functionalities. `models.py`: segmentation model types. `image_helpers.py`: helper functions. `pspnet`: CNN for semantic segmentation. `stylisation/`: directory containing all stylisation related functionalities. `style_transfer_model.py`: style transfer model. `helpers.py`: helper functions. `vggnet`: CNN for style transfer. `templates/`: directory for HTML web files. `static/`: directory for flow of input, intermediate and output files and images. `app.py`: main application to launch.

The system's software is inherently very modular, and profits from an Object Oriented Programming (OOP) structure, for both segmentation and stylisation functions. For instance, in the scenario that an extra method of segmentation is to be added, it would only be required to write an extra class for that new segmentation model in the `segmentation/models.py` file (see Figure 4.5), add the correct handlers in the main script `app.py`, and link to the correct inputs and outputs in the HTML web files from the `templates/` directory. The rest of the system should already be in place for it to function properly. Therefore, should the research be picked up from the current system, it would be relatively simple to modify or add to the current codebase.

```
class SemanticModel():
    ...
class ThresholdModel():
    ...
class ColourModel():
    ...
class xyz_Model():
    # Add the code for this new segmentation model
```

Figure 4.5: Structure of the `models.py` file containing the segmentors model

4.2 High-Level Overview

As discussed in the Analysis and Design section, the main goal is that the system offers greater control over the outcome of the style transfer task. After having chosen a content image and a style image, the user is invited to complete a form with a number of options. Depending on initial option choices, some other extra options might appear (see Figure 4.6). The first step

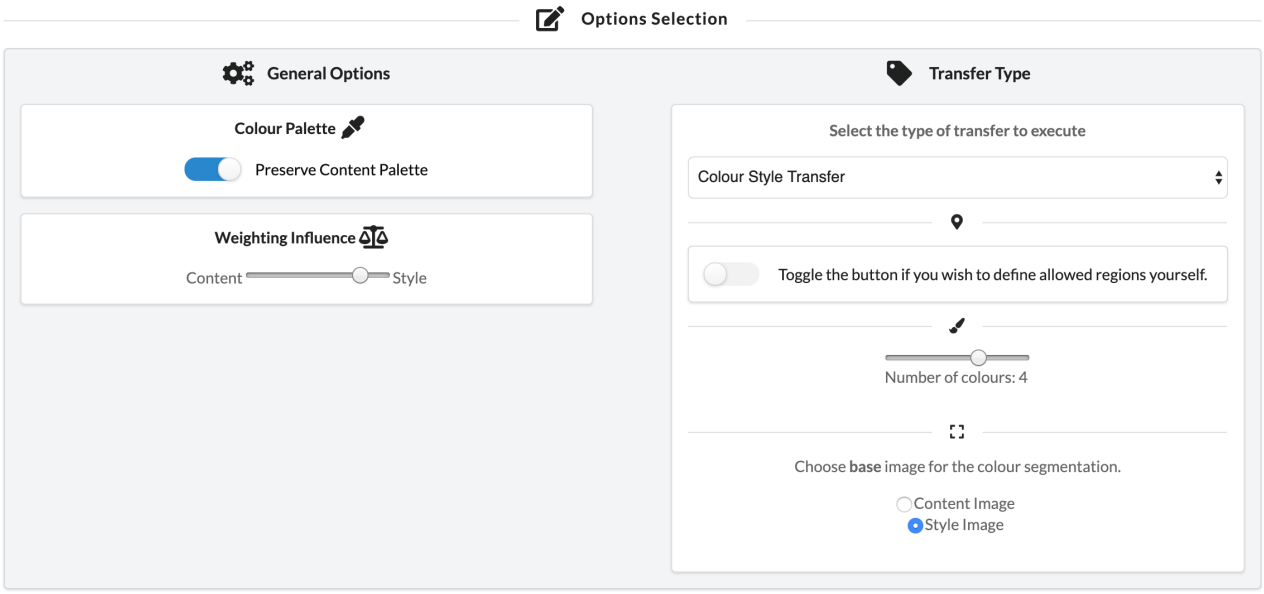


Figure 4.6: Application's options form

is to indicate whether the colour of the content image should be preserved during the transfer process, in which case histogram matching is performed using the content image as the target and the style image as the source, as already explained in section 3.2.1. The second option involves choosing through a slider the amount of relative influence the content and the style image can have on the output (see section 3.2.3). Then the user can select in a dropdown menu the type of transfer to be performed:

- *Full Style Transfer*: no segmentation.
- *Semantic Style Transfer*: with a semantic segmentation step.

- *Threshold Style Transfer*: with a threshold segmentation step.
- *Colour Style Tranfer*: with a colour segmentation step.

4.2.1 The Four Transfer Options

The subsequent sections explain the different type of possible workflows of the app depending on the transfer type chosen by the user.

Full Style Transfer

The system does not require any additional input and performs straight away the transfer task, using the entirety of the content image and the style image. This operation exactly matches the style transfer process of [2], and will mainly be used as a benchmark against the newly introduced features.

Semantic Style Transfer

Semantic segmentation, as described in section 3.2.2 is performed on the content image only. The user is then presented with an intermediate page where all the known and found labelled regions are displayed as selectable mask images (see Figure 4.7). After having selected the "allowed" masks, a combined binary mask is created from them such that the final style transfer step is performed on the content image exclusively in those selected regions.

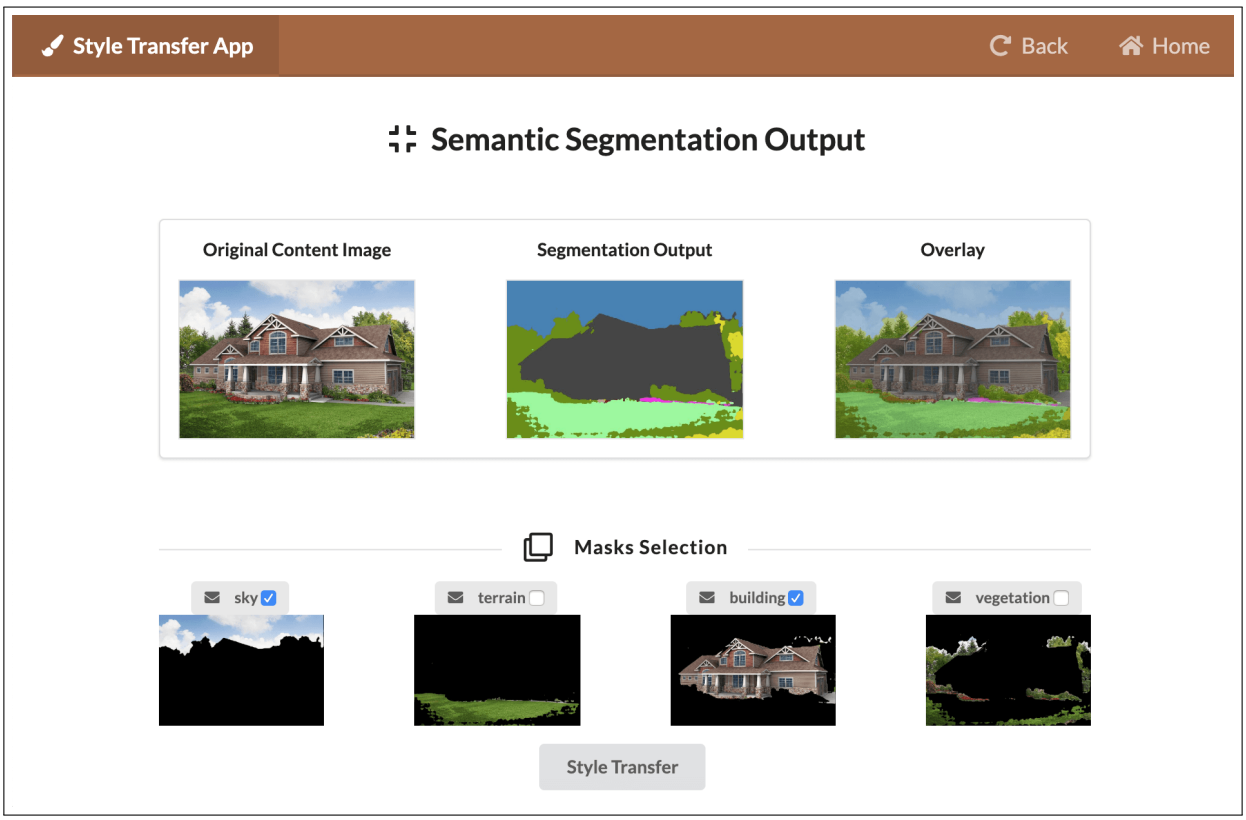


Figure 4.7: Application's intermediate page for semantic mask selection

Threshold Style Transfer

For both threshold and colour transfer options, the user is required to specify extra parameters to proceed further. In this case, n_t, the number of threholds is necessary to segment the images into the correct number of brightness levels.
An additional option *user-defined region* is also available, which, if selected, allows a custom selection of the individual segments from the segmentation output (similar to the semantic case in Figure 4.7 but with brightness level segments) for both the content and the style image. This would lead to the transfer occuring from selected regions in the style image to selected regions in the content image. If the the option is not selected, the style transfer is executed everywhere according to value-matched segments between the two masks, as previously explained in section 3.2.2 and illustrated in Figure 4.8

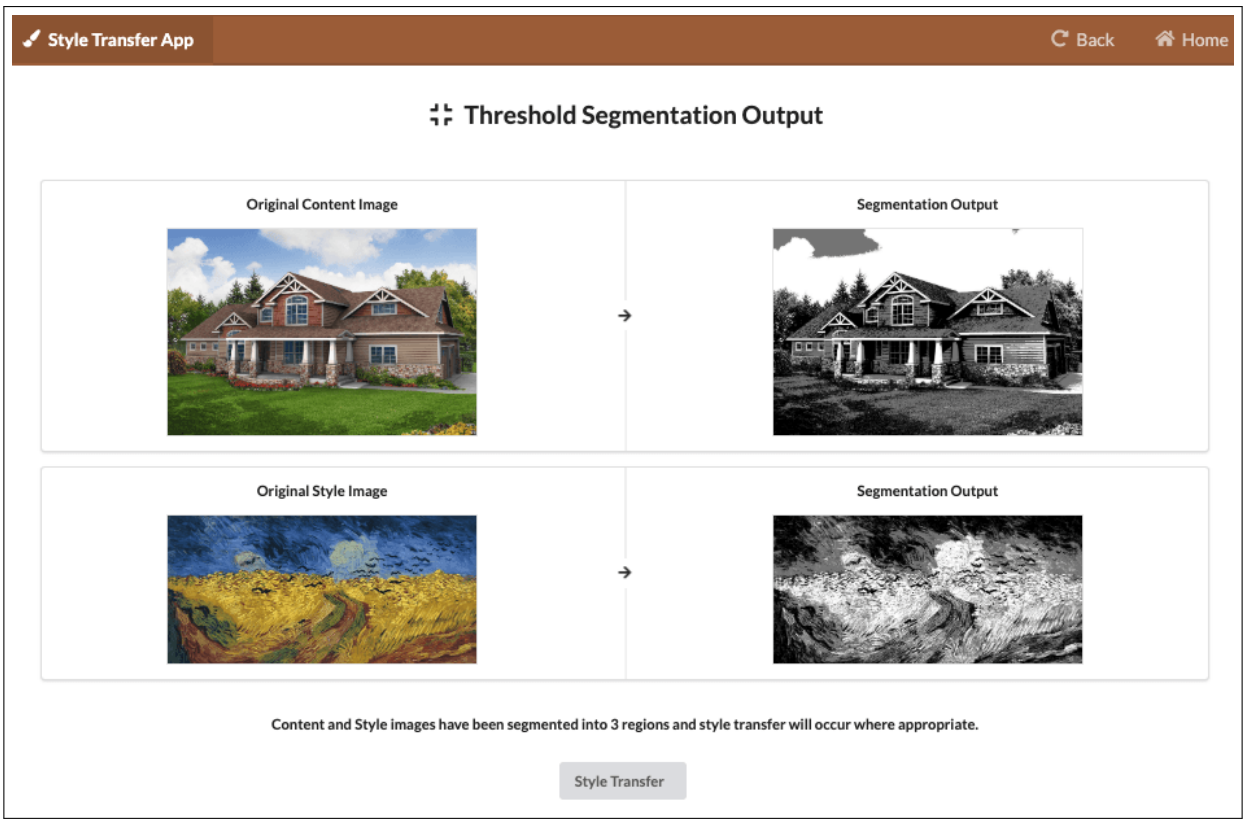


Figure 4.8: Application's intermediate threshold segmentation page (when *user-defined regions* is not selected)

Colour Style Transfer

The colour style transfer process is very similar to that of the threshold one. Here, n_c designates the number of dominant colours to be found. The additional *user-defined region* option works very much in the same spirit as in the threshold case. However, depending on whether it is selected, the program will look for dominant colours differently. If selected, two separate colour segmentor models will run on the content and style image, finding the respective n_c dominant colours of each image, and segmenting them accordingly. As before, the different segments of both images are presented for the user to select as allowed source and destination

regions (similar to Figure 4.7 but with colour-based segments). If not selected, i.e. the app should perform an automatic matching between the segments of the two masks, only one colour segmentor model is created for both images, since there needs to be a match in the values of the two masks' segments. In this case, the user is first required to specify which image should be chosen to be the *base image* of the colour segmentor model[2].

4.2.2 Output Page

Finally, the output page of the app shows the final resulting image of the style transfer task with all the options taken into consideration.

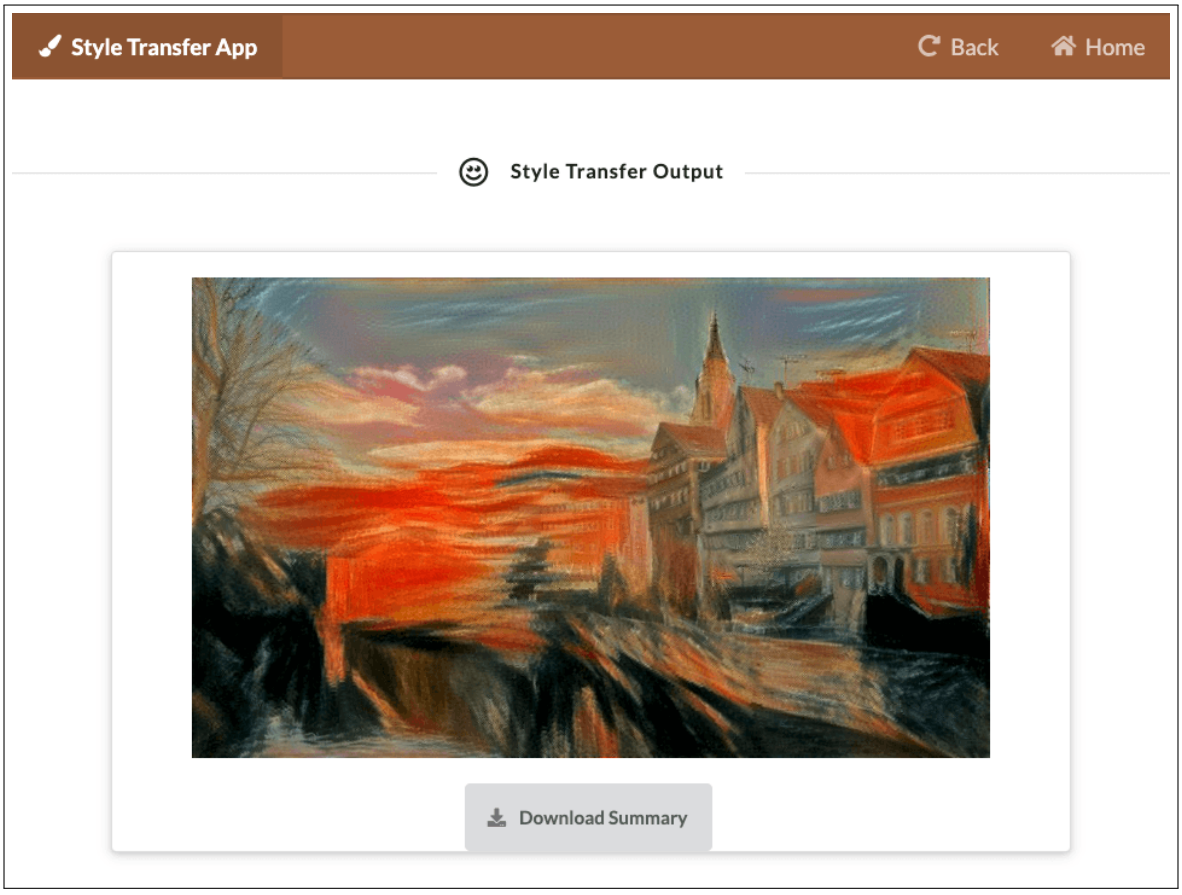


Figure 4.9: Output page

Additionally, a download button lets the user download a `summary.zip` file containing all the input images, the intermediate images (segmentation masks and their selection), and an `info.txt` text file with an auto-generated summary of the process and its selected options (see Figure 4.10).
An overview of the application workflow is depicted in the diagram in Figure 4.11.

[2]The image from which the n_c most dominant colours will be searched.

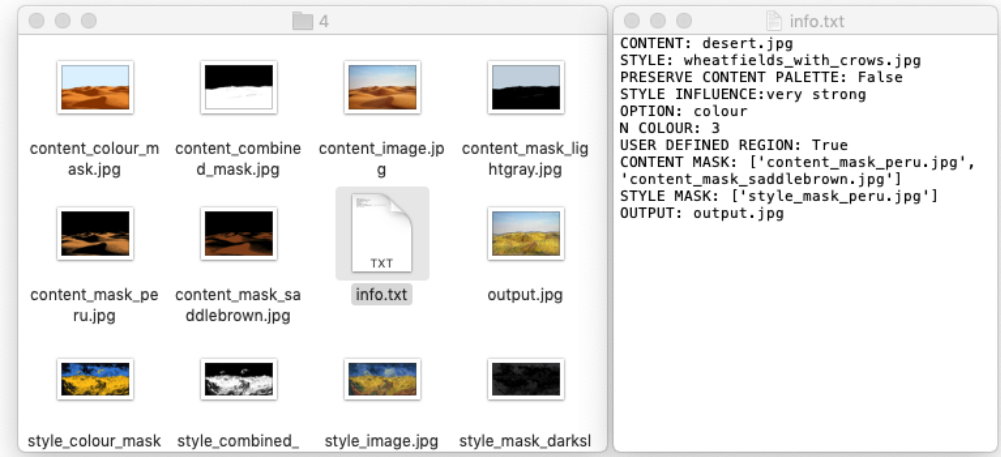


Figure 4.10: `summary.zip` file content.

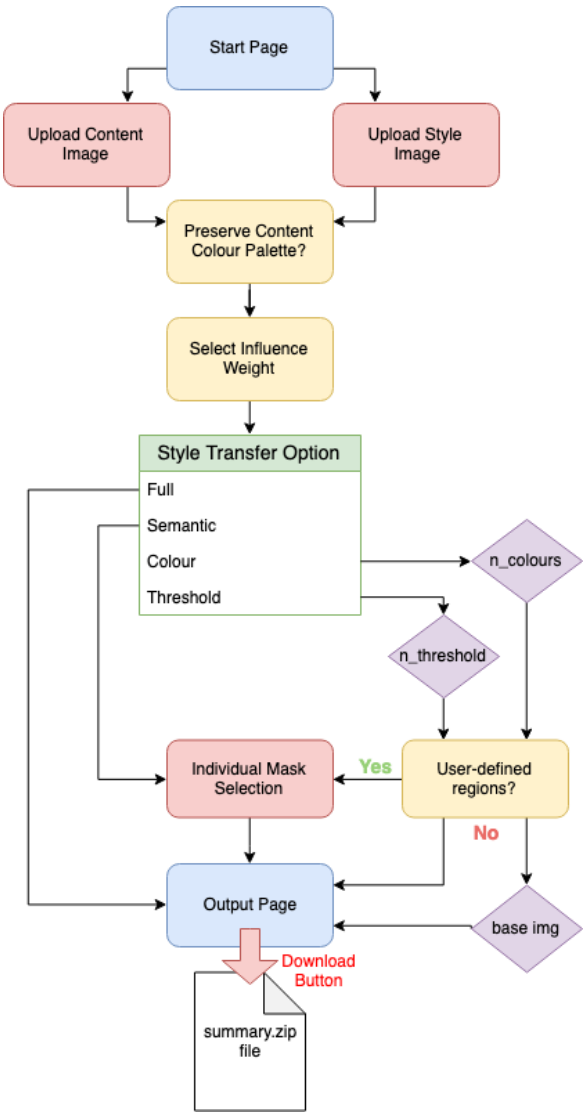


Figure 4.11: Flowchart diagram of the application workflow.

5. Testing

The task of style transfer can be ambiguous at different levels. Indeed, its definition is not strictly well-defined, and can be interpreted differently depending on diverse factors.
Therefore, it is formally quite challenging to determine whether a particular style transfer process is successful or not. It is generally accepted that a style transfer is successful when the resulting image "looks like" the content image with the design of the style image and with a high perceptual quality. Using a more scientific approach, there exist desirable criteria that can commonly define a successful style transfer such as the presence of hallucinative stylistic elements, the accuracy of content presence, or the control over the transfer outcome. Indeed it is exactly the absence of a formal way to assess a style transfer process that makes parameter control a successful factor.

Style Transfer Quality

In order to assess the visual pleasance of the resulting image from the new algorithm, it will be visually analysed with criteria to look for including hallucinative components, perceptual quality, accuracy of the content presence, accuracy with respect to the style image, and last but not least, the accuracy of the resulting image with respect to the control parameters. In this respect, the `zip` file download functionality summarising the operation will prove to be useful when analysing the ins and outs of a particular run process.

Granular Control over Outcome

As mentioned above, a fine control over the outcome is desirable since the task of style transfer is not well defined. This way, one would be able to perform style transfer exactly how one desires, which is bound to reflect a successful process. Additionally, it would also imply that the shape of the output image is likely to be more predictable, as we are in control of its characteristics.

Speed performance

In order to assess the speed performance of the system, duration for various operations was recorded for the server-deployed system, and compared with the duration for the local run equivalent, using different input content and style images.

6. Results and Evaluation

In the subsequent section, all the tests and outputs were realised using the described system in the Implementation section, unless stated otherwise. Additionally, all input test images used in this section can be found in the Appendix.

6.1 The Fundamental Test

It makes sense to start with the most basic functionality of the system which is a simple neural style transfer with no segmentation, or additional option. Note that this corresponds to the default configuration of the system:

- Content Image: input
- Style Image: input
- Content Colour Preservation: false
- Weight ratio for content and style influence: $\frac{\alpha}{\beta} = \frac{1}{0.2}$
- Segmentation: none (full style transfer)

In the following set of figures, the content image is fixed ("Tubingen" photograph) while the style image is varied to obtain different outputs.

(a) Content **(b)** Style **(c)** Output

Figure 6.1: Full Style Transfer 1. (a) "Tubingen" photograph. (b) "Starry Night" painting.

(a) Content **(b)** Style **(c)** Output

Figure 6.2: Full Style Transfer 2. (a) "Tubingen" photograph. (b) "The Shipwreck of the Minautor" painting.

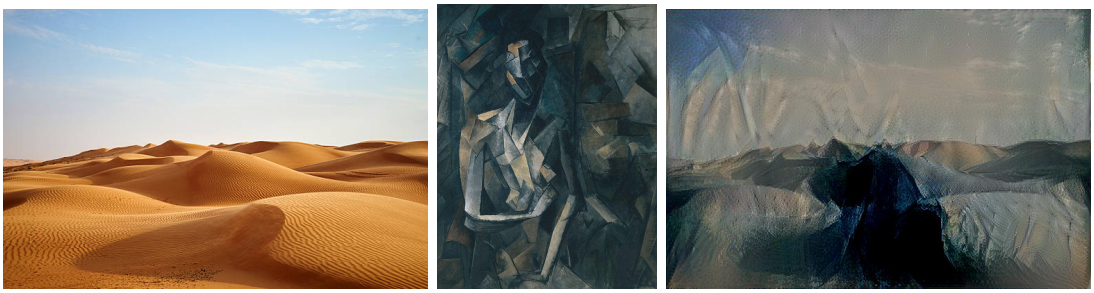
(a) Content **(b)** Style **(c)** Output

Figure 6.3: Full Style Transfer 3. (a) "Tubingen" photograph. (b) "Composition VII" painting.

As can be seen from the above images, the correct shapes of the content photograph are found in all the outputs and the style of each artwork is also correctly applied. In the next set of figures, the same system configuration is kept; however this time, the style image is fixed ("Femme Nue Assise" artwork by Pablo Picasso) and the content is varied.

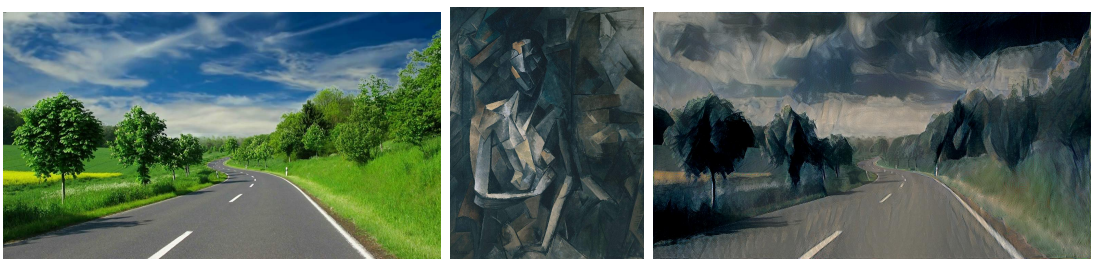
(a) Content **(b)** Style **(c)** Output

Figure 6.4: Full Style Transfer 4. (a) "House" photograph. (b) "Femme Nue Assise" painting.



(a) Content **(b)** Style **(c)** Output

Figure 6.5: Full Style Transfer 5. (a) "Desert" photograph. (b) "Femme Nue Assise" painting.



(a) Content **(b)** Style **(c)** Output

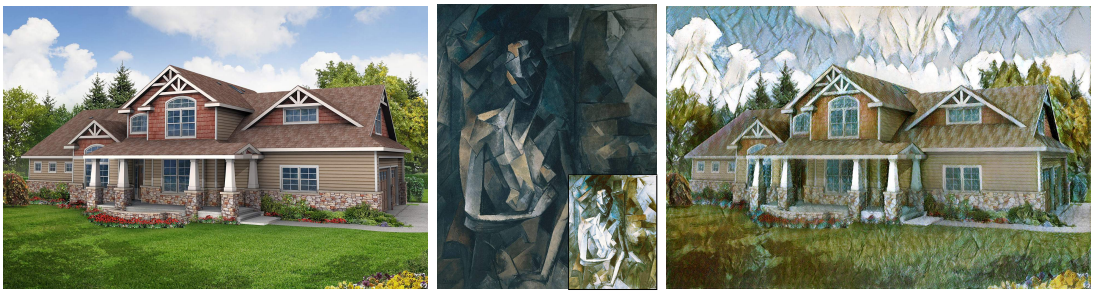
Figure 6.6: Full Style Transfer 6. (a) "Road" photograph. (b) "Femme Nue Assise" painting.

6.2 Colour Palette

In this second part, the colour preservation of the content image is put to the test. The rest of the system configuration remains otherwise unchanged:

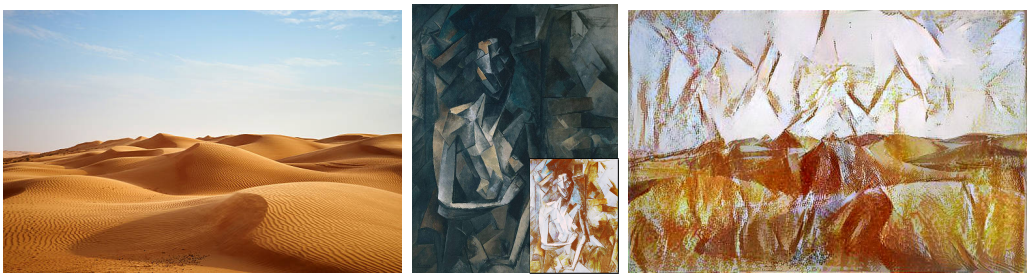
- Content Image: input
- Style Image: input
- Content Colour Preservation: true
- Weight ratio for content and style influence: $\frac{\alpha}{\beta} = \frac{1}{0.2}$
- Segmentation: none (full style transfer)

In order to do so, the same three pairs of content and style images in Figures 6.4, 6.5, 6.6 are used to make later comparison easier.



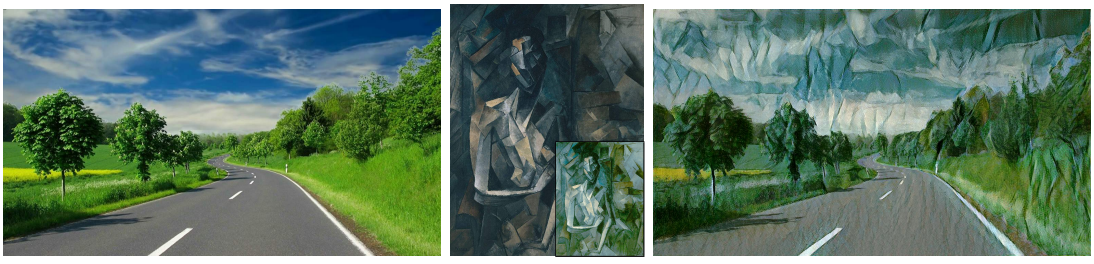
(a) Content **(b)** Style **(c)** Output

Figure 6.7: Colour Preservation 1. (a) "House" photograph. (b) "Femme Nue Assise" painting.



(a) Content **(b)** Style **(c)** Output

Figure 6.8: Colour Preservation 2. (a) "Desert" photograph. (b) "Femme Nue Assise" painting.



(a) Content **(b)** Style **(c)** Output

Figure 6.9: Colour Preservation 3. (a) "Road" photograph. (b) "Femmme Nue Assise" painting.

Interestingly the output images are now much closer to the original appearence of content images. Indeed, there is a good match in the meaningful areas of colours between the content and output images. Instead of being completely in the style of Picasso's painting, the resulting images seem to have inherited more of its painting technique, as if the artist had painted them using the same brushing movements, but with a different colour palette at hand. Note that although the input style images to the system were the original painting of Picasso, the actual input style image to the neural style transfer algorithm is that produced by the preprocessing operation of histogram matching. Those intermediate style images are displayed in the bottom right corner of the original painting in the above figures.
Thus, this use case might be desirable when the particular texture of a painting is to be applied to a photograph, without necessarily transfering the colours as well.

6.3 Balance of style and content

This section shows the result of controlling the relative weight of the content and style in the transfer process. Therefore, keeping the content weight constant: $\alpha = 1$, the style weight is varied: $\beta = \{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2\}$. The rest of the configuration is set to its default state and thus the content colour preservation is brought back to false.

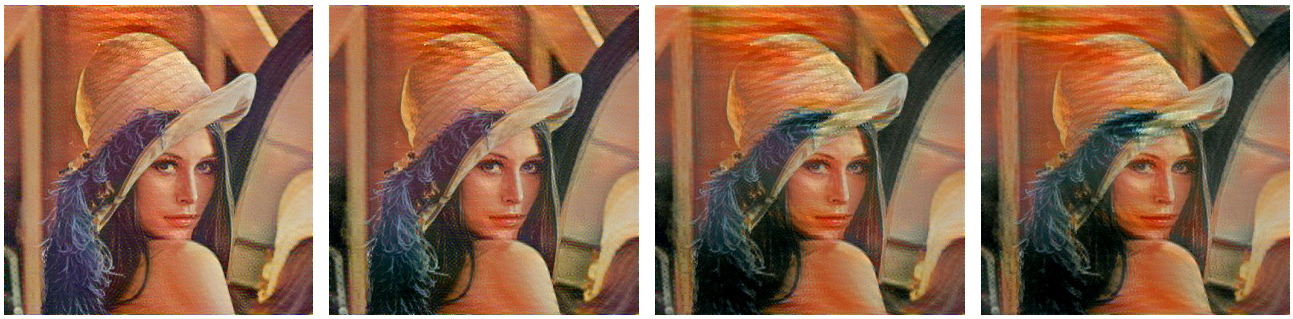
- Content Image: input
- Style Image: input
- Content Colour Preservation: false
- Weight ratio for content and style influence: $\frac{\alpha}{\beta}$
- Segmentation: none (full style transfer)

For the following set of tests, "Lena" photograph and "Der Schrei" painting were chosen to be the content and style image respectively, as depicted in Figure 6.10

(a) Content Image **(b)** Style Image

Figure 6.10: Input images to test weight ratio for content and style influence

The results are illustrated in Figure 6.11. It can be observed how as the ratio $\frac{\alpha}{\beta}$ decreases, the influence of the painting is becoming greater in the resulting image.



(a) $\frac{\alpha}{\beta} = 2000$ **(b)** $\frac{\alpha}{\beta} = 1000$ **(c)** $\frac{\alpha}{\beta} = 200$ **(d)** $\frac{\alpha}{\beta} = 100$

(e) $\frac{\alpha}{\beta} = 20$ **(f)** $\frac{\alpha}{\beta} = 10$ **(g)** $\frac{\alpha}{\beta} = 5$

Figure 6.11: Tests for content and style weight ratio control

The choice of ratio range was determined based upon empirical results. It was established that any value above 2000 was typically useless as artistic elements of the style image could barely be noticed. With regards to the lower bound, a ratio below 5 led to the same looking artistic creation, which suggested an asymptotic behaviour on the lower end. Nevertheless, the results show that controlling the balance of content and style is indeed possible and that it was correctly implemented.

We have seen how it is possible to control some parameters in the neural style transfer algorithm to obtain resulting outputs of different colours, shapes and styles. However, the implemented system also allows the use of spatial guidance masks which have the power to dictate both what regions of the style image are allowed to be used in the transer process, as well as the regions of the content image that will be stylised.
As discussed in the previous sections, there exist many different techniques and parameters for this, such as how many segments are included in one spatial guidance mask, or whether the pair of masks should be used to perform a matched transfer between corresponding segments. These shall all be investigated one at a time.
Thus, the following section presents results obtained from neural style transfer paired with the three of threshold segmentation, colour segmentation, and semantic segmentation.

6.4 Threshold-based Transfer

The motivation for prepending a style transfer with a threshold segmentation step was to segment (and create spatial masks) based on the images' brightness level, leading to a style transfer of dark stylistic elements to dark regions of the content image and similarly for bright regions.

6.4.1 Automatic matching of regions

The system's configuration for this section has the *user-defined regions* parameter set to false as we let the system decide the pairing of mask regions:

- Content Image: input
- Style Image: input
- Content Colour Preservation: false
- Weight ratio for content and style influence: $\frac{\alpha}{\beta} = 5$.
- Segmentation: threshold
  - Number of thresholds n_t: input
  - User-defined regions: false

In this first set of tests, a default case style transfer is compared with a threshold case transfer where n_c is varied from 1 to 4, with an expected refined process as the parameter is increased. For this purpose, the "Tubingen" photograph and "Starry Night" painting are selected as test images (see Figure 6.12).

(a) Content Image **(b)** Style Image

Figure 6.12: Input images to test threshold-based transfer (automatic)

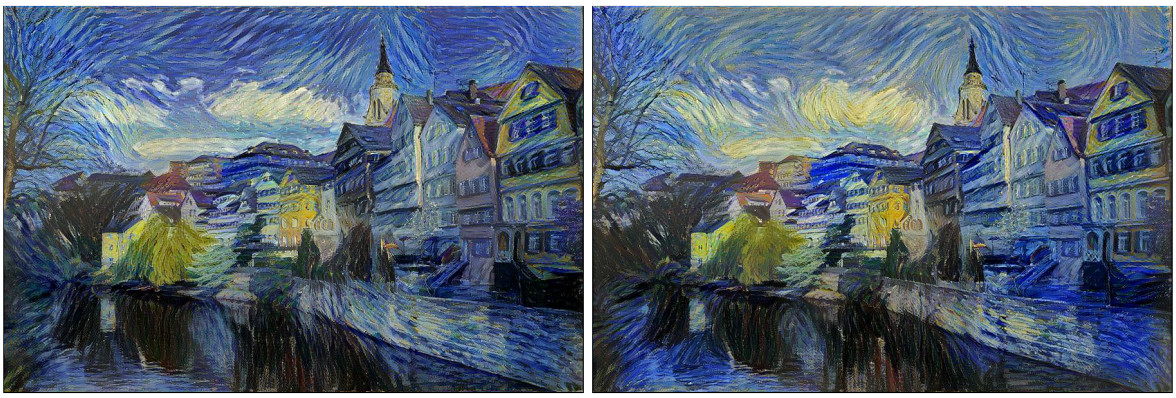
Threshold number = 1

With only one threshold value, the obtained masks are binary masks. Figure 6.13 shows the segmentation masks obtained from running the threshold segmentation algorithm with $n_t = 1$.

(a) Content Mask **(b)** Style Mask

Figure 6.13: Threshold Segmentation Output ($n_t = 1$)

These binary masks suggest that the yellow parts of the painting and light blue areas should be transfered to the photograph in regions such as the sky or some house facades, and the darker shades (black and dark blue) to the roofs and the river. The output of the style transfer with those given spatial guidance mask is illustrated in Figure 6.14, alongside with the default baseline case.



(a) Default Style Transfer **(b)** Threshold-based Transfer ($n_t = 1$)

Figure 6.14: Threshold-based Transfer Output Comparison ($n_t = 1$)

From the above results, it can be observed that threshold style transfer practically did not change the process, as the output is very similar to the default one. One can note the lighter shades (of yellow and light blue) in the sky, but the general appearence of the resulting image almost identical.

More Thresholds

The same operation is repeated for $n_t = \{2, 3, 4\}$ which segments the input images into more regions of brightness levels. The results are directly shown in Figure 6.15.

(a) Default Style Transfer

(b) Threshold-based Transfer ($n_t = 2$)

(c) Threshold-based Transfer ($n_t = 3$)

(d) Threshold-based Transfer ($n_t = 4$)

Figure 6.15: Threshold-based Transfer Output Comparison ($n_t = 2, 3, 4$)

The previous observation also applies to cases where the number of threshold is bigger; their effect on the outcome is barely noticeable. It could even be argued that the visual pleasance of images resulting from a threshold style transfer is sightly less than that of the default case, as too many constraints are forced upon the transfer; however, this remains a subjective matter. An interesting finding is that the use of threshold segmentation distorts the colour accuracy in the output. Indeed, the clouds of the photograph are becoming less visible as they turn yellow, and most of the rooftops turn to dark blue in the process. This suggests that the default neural style transfer algorithm already performs the job of transfering style intelligently between regions with similar properties.

6.4.2 User-defined region

Threshold segmentation can also be used as an initial step to obtain different region segments separately and feed the neural style algorithm a self-composed spatial guidance mask based on the selection of segments. Therefore, the configuration of the system will hold "true" in the *user-defined region* field.

- Content Image: input
- Style Image: input
- Content Colour Preservation: false

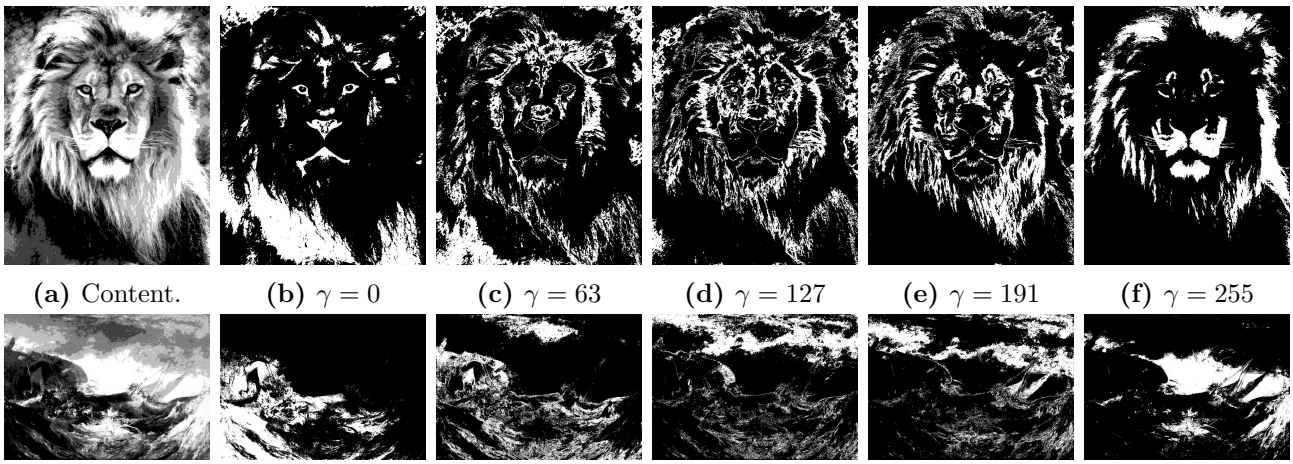
- Weight ratio for content and style influence: $\frac{\alpha}{\beta} = 5$.
- Segmentation: threshold
  - Number of thresholds n_t: input
  - User-defined regions: true

The content and style images ("Lion" photograph and "The Shipwreck of the Minautor" painting respectively) for this second set of tests are displayed in Figure 6.16.

(a) Content Image **(b)** Style Image

Figure 6.16: Input images to test threshold-based transfer (user-defined)

The number of threshold is chosen to be $n_t = 4$ in order to obtain more granularity in the choice of allowed segments. Figure 6.17 shows the output of the threshold segmentation on the content and style images, alongside their individual brightness level segments.

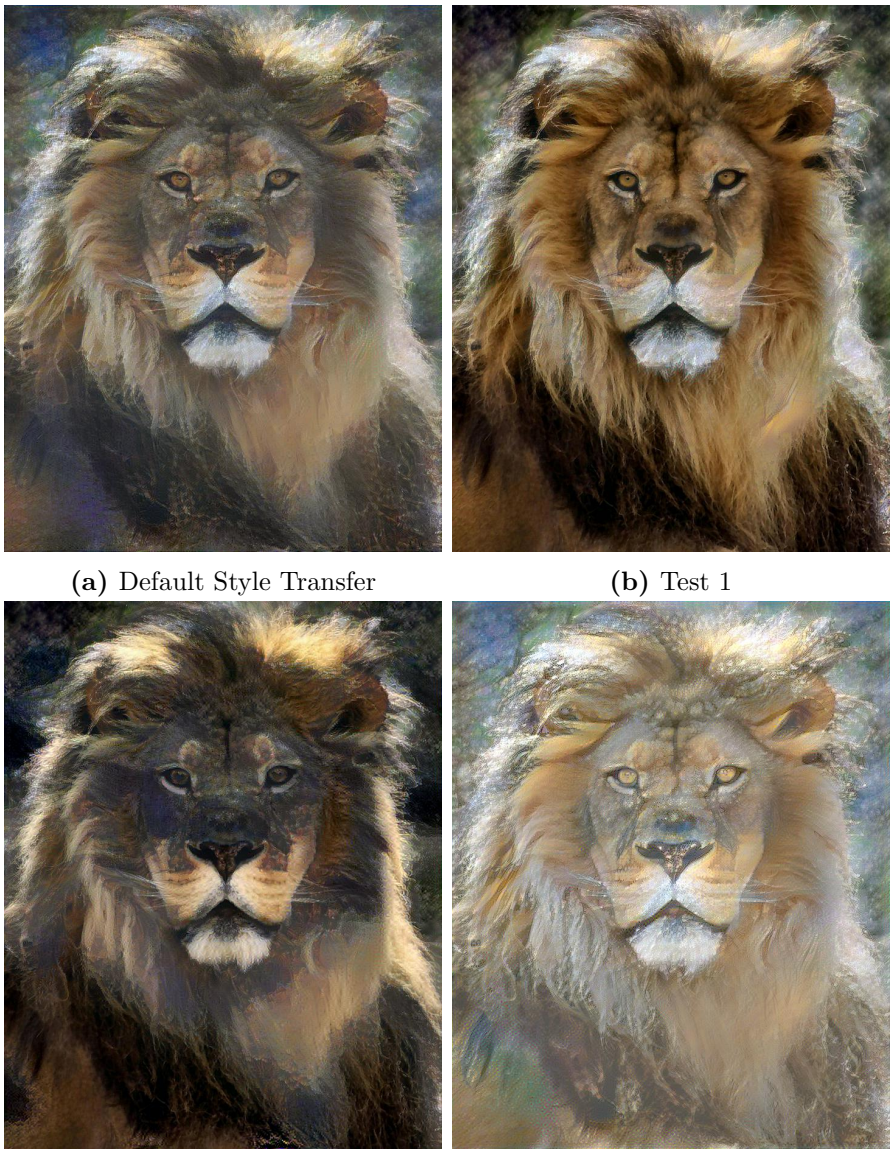


(a) Content. **(b)** $\gamma = 0$ **(c)** $\gamma = 63$ **(d)** $\gamma = 127$ **(e)** $\gamma = 191$ **(f)** $\gamma = 255$

(g) Style. **(h)** $\gamma = 0$ **(i)** $\gamma = 63$ **(j)** $\gamma = 127$ **(k)** $\gamma = 191$ **(l)** $\gamma = 255$

Figure 6.17: Threshold Segmentation Output ($n_t = 4$) with gray scale brightness level (γ).

Figure 6.18 are results obtained from performing style transfer using different combinations of segmented regions together.



(a) Default Style Transfer **(b)** Test 1

(c) Test 2 **(d)** Test 3

Figure 6.18: Style Transfer Output with custom combination of threshold msks. Test 1: Content mask ($\gamma : 255$) - style mask ($\gamma : 255$). Test 2: Content mask ($\gamma : 127 - 191$) - style mask ($\gamma : 0, 63$). Test 3: Content mask ($\gamma : 0 - 63 - 127 - 191 - 255$) - style mask ($\gamma : 255$).

In the above set of results, the outputs look very diverse depending on the segment selection. In test 1, only bright regions of both images are involved. This made the edges of the lion's fur brighter and sparklier. In test 2, the middle range regions of the lion inherited from the style of dark regions of the painting. In test 3, the whole content image was styled with only the brighter parts of the painting.

Whilst output images might not be all judged as visually pleasing, there is more room for artistic creativity, and producing aesthetic results should just be a matter of experience with the choice of input image pair and the combinations of segments in creating the guidance mask. We have thus demonstrated the extension of achievable results through threshold segmentation.

6.5 Colour-based Transfer

In a similar way as for threshold segmentation, the goal of colour segmentation is to separate and pair regions of the content image and style images which possess similar properties; only this time, it has to do with colour.

6.5.1 Automatic matching of regions

To start with, the *user-defined regions* is left de-activated and the extra *base* parameter is introduced to indicate whether to select the most dominant colours from the content image or from the style image.

- Content Image: input
- Style Image: input
- Content Colour Preservation: false
- Weight ratio for content and style influence: $\frac{\alpha}{\beta} = 5$
- Segmentation: colour
  - Number of colours n_c: input
  - User-defined regions: false
  - Base image: input

In the following experiments content and style images are carefully chosen to contain approximately the same range of colours such that the automatic pairing between regions is maximised. Here, "Mou Aysha" photograph will be used with "George Moore" painting (see Figure 6.19).

(a) Content Image **(b)** Style Image

Figure 6.19: Input images to test colour-based transfer (automatic)

Seeing that approximately three colours dominate both images, the number of colour segments parameter was set to three: $n_c = 3$ and the content image was set as base image[1], from which the segmentation masks in Figure 6.20 were obtained.

(a) Content Image **(b)** Style Image

Figure 6.20: Colour Segmentation Output ($n_c = 3$ - Base Image: Content)

Given the shape of the guidance masks, we thus expect a direct mapping between both faces, the style of the background to be applied to the girl's clothes, and the style of the black jacket to be transfered to the background of the content image. The result of the style transfer with those guidance masks is shown in Figure 6.21, alongside with the default transfer output.

(a) Default Style Transfer **(b)** Colour-based ($n_c = 3$)

Figure 6.21: Colour-based Transfer Output Comparison

The spatial guidance mask did not help creating a better output image than the baseline configuration. This is a similar situation as for the threshold segmentation case, where the default neural style transfer did a better job, as it creates an image that is already minimising

[1]In all test cases, the choice of base image was arbitrary and did not matter too much when the input images had similar colour palettes. The pair of images were segmented into rougly the same regions.

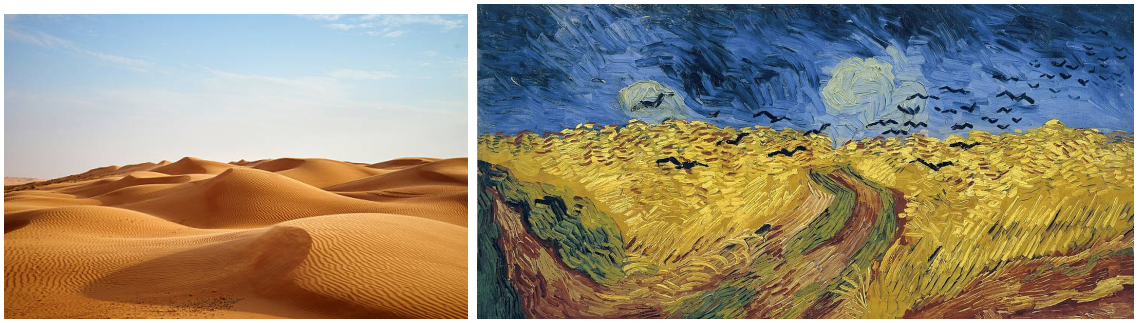
both content loss and style loss, a process which inevitably handles colour information as well. This indicates that there might actually be no need to create such colour guidance masks to enhance a style transfer outcome. A few additional subsequent tests were conducted in a similar fashion whilst varying n_c and a few other input images, and the standard neural algorithm almost always outperformed the colour-based transfer case. Taking this into consideration, the next objective was to test the system with custom segment selection.

6.5.2 User-defined region

For this section's configuration, there is no more need to specify which image to choose as base image for the colour segmentation model as content and style images can be segmented individually:

- Content Image: input
- Style Image: input
- Content Colour Preservation: false
- Weight ratio for content and style influence: $\frac{\alpha}{\beta} = 5$
- Segmentation: colour
  - Number of colours n_c: input
  - User-defined regions: true

Using the colour segmentation model as a region selective model does not necessarily require both images to match in terms of colour palette, though it does most often still make sense for it to be the case; it only depends on the sort of creation one wants to achieve.
The choice of images for the next set of tests are displayed in Figure 6.22 ("Desert" photograph as content and "Wheatfields with Crows" painting as style).

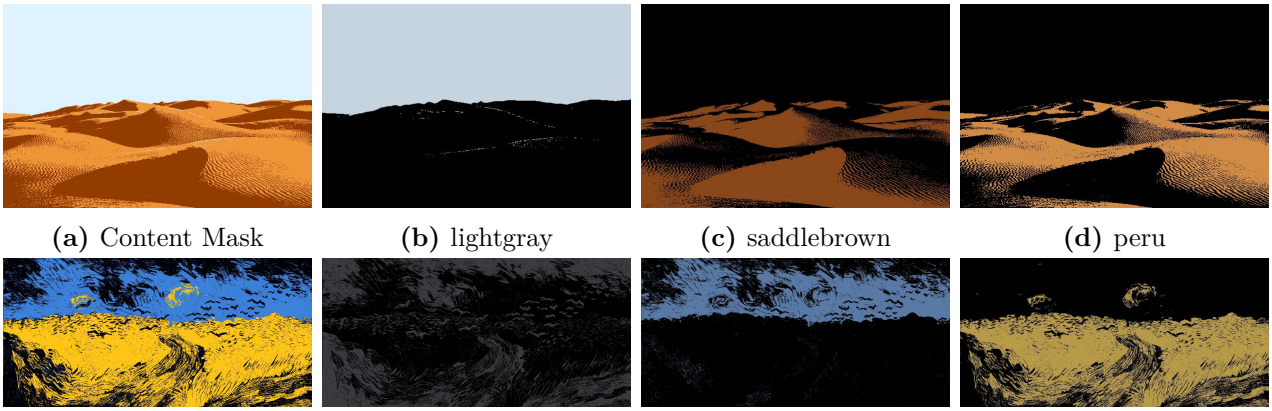


(a) Content Image **(b)** Style Image

Figure 6.22: Input images to test colour-based transfer (user-defined)

Performing individual colour segmentations on the input images with $n_c = 3$ yielded the six segmentation masks displayed in Figure 6.23. It is worth noting that the individual segments label names are directly derived from the RGB pixel value represented by the specific region,

using the `webcolors` package for Python[2]. This small design choice made it a lot easier to distinguish and categorise individual segments, as opposed to having to deal with triplets of integers between 0 and 255 (the RGB case).

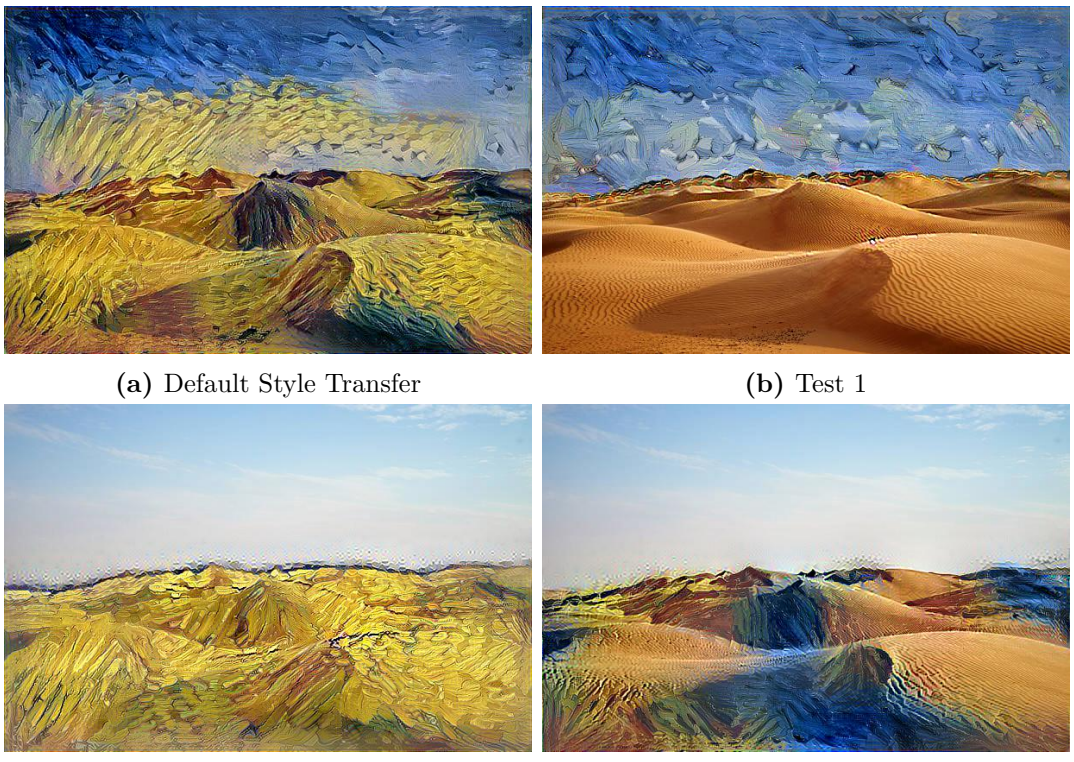


(a) Content Mask **(b)** lightgray **(c)** saddlebrown **(d)** peru

(e) Style Mask **(f)** darkslategray **(g)** steelblue **(h)** peru

Figure 6.23: Colour Segmentation Output $n_c = 3$

Depending on the choice of segments for the final guidance mask, different style transfer outputs were produced. Figure 6.24 shows another comparison of the default style transfer case with the region-defined ones.



(a) Default Style Transfer **(b)** Test 1

(c) Test 2 **(d)** Test 3

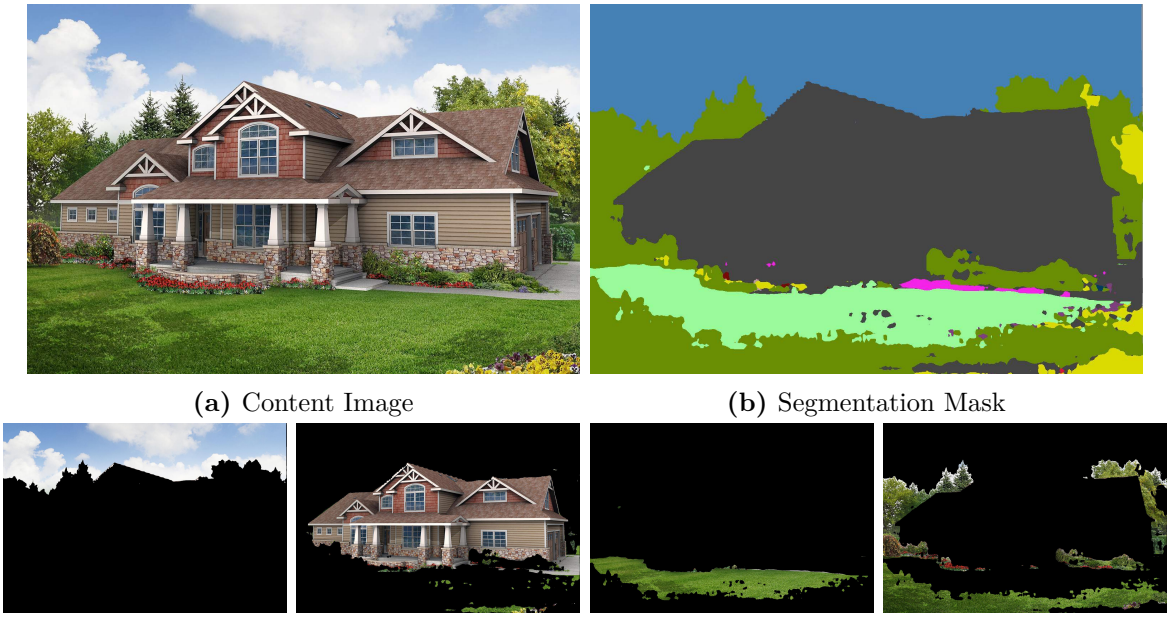
Figure 6.24: Style Transfer Output with custom combination of threshold masks. Test 1: content mask: "lightgray" - style mask: "steelblue" Test 2: content mask: "saddlebrown" + "peru" - style mask: "peru". Test 3: content mask: "saddlebrown" - style mask: "darkslategray".

[2]Where no matched colour label was found, the name of the closest colour was chosen instead.

As shown in Figure 6.24, region control of style and content elements is possible via a colour-based segmentation. In test 1, the sky of the painting was directly transfered to the sky of the photograph. Test 2 transformed the dunes area with the style of the wheatfield regions, and finally test 3 linked part of the sky and the trails across the field of the artwork to the shaded areas of dunes in the content image.
It is interesting to note that the sky portion of the resulting image of test 1 using colour segmentation does not include a large yellow region like the default case output does, which in many ways will be considered as more desirable. Inspecting the content image closer, it is found that the lower region of the sky is indeed further from a blue shade, which might explain the appearance of yellow stylistic elements in that zone using the default neural algorithm. By forcing the sky to be stylised with blue elements only from the style image, this type of issue can be avoided.
Therefore, while not all outputs might be considered as good artistical creations, there are countless possible of combinations, suggesting that some should most certainly produce visually pleasing results. Indeed, as is the case for test 1, it has been shown how colour-based segmentation might be more beneficial than the baseline approach.

6.6 Semantic-based Transfer

Lastly, this section studies and tests the performance and the use of semantic segmentation before the style transfer step. As stated in section 4.2.1, semantic segmentation is only performed on content images as they are usually real photographs. For the first series of test the "House" photograph is used again. Figure 6.25 shows the output of a semantic segmentation run on the content image, including all the individual segments.

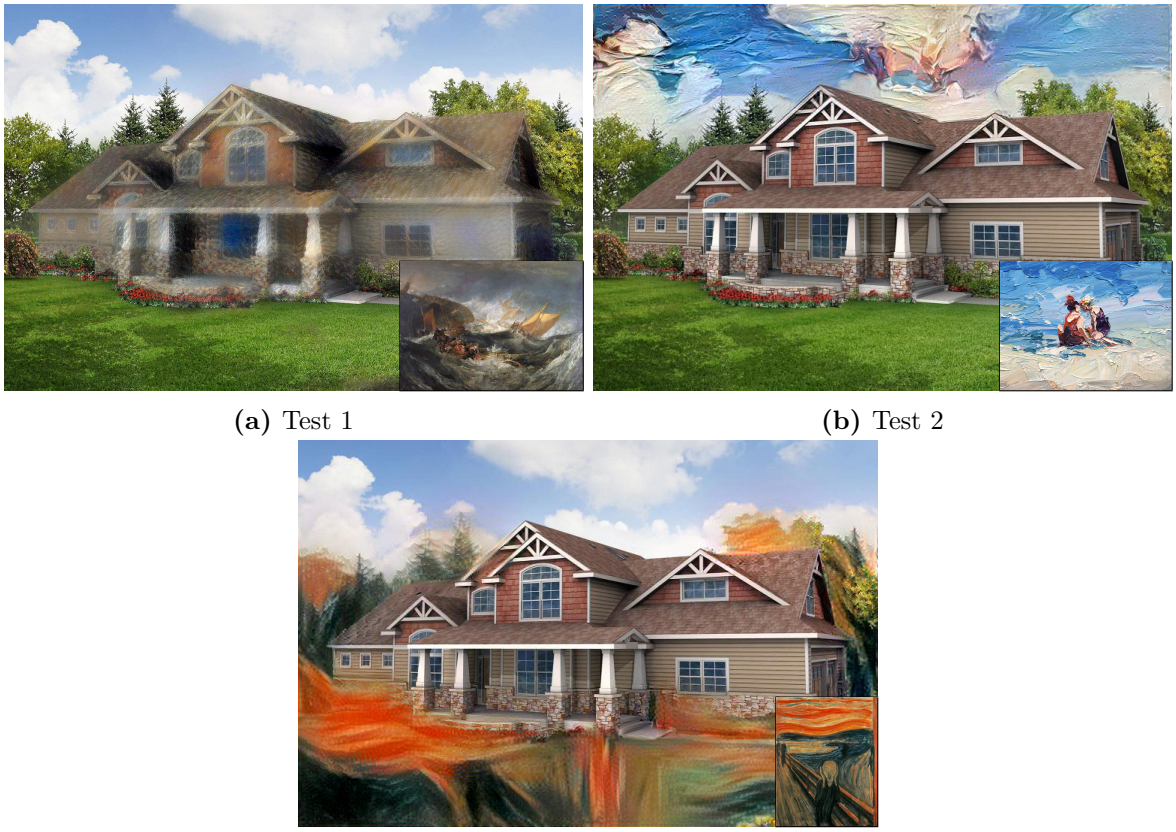


(a) Content Image **(b)** Segmentation Mask

(c) Sky **(d)** Building **(e)** Terrain **(f)** Vegetation

Figure 6.25: Semantic Segmentation Output on "House" photograph

From the above figure, it can be seen how the semantic segmentation model successfully identified the main regions of the image, including: "sky", "building", "terrain", "vegetation". The advantages of this segmentation method are clear as the concerned regions need not have similar colour or brightness properties; they only need to represent a known label to the semantic neural network. It therefore allows to create segmentation masks that were not possible to obtain using the threshold or colour-based approach.
In Figure 6.26 below, we present the style transfer results obtained from combining different segments of the previous segmentation, and using different input style images.



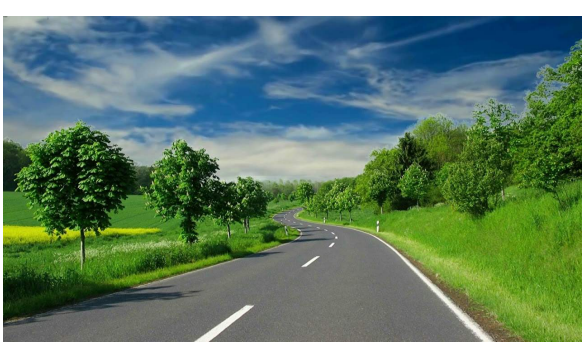
(a) Test 1 **(b)** Test 2

(c) Test 3

Figure 6.26: Style Transfer Output with custom combination of semantic masks Test 1: content mask: "house" - style image: "The Shipwreck of the Minautor". Test 2: content mask: "sky" - style image: "Children on the Beach". Test 3: content mask "terrain" + "vegetation" - style mask: "Der Schrei".

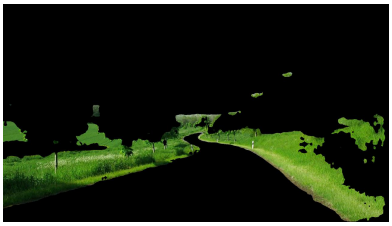
As an example for the above point, it would have been impossible to separately stylise the sky and house regions of this content image using previous segmentation methods. Indeed, the clouds would have almost always certainly been in the same segmented region as the white borders of the house, making these two regions unseparable.

As always, the content segments and the style image can be chosen to reflect better pairings in an attempt to obtain better visually pleasing results. In Figure 6.27, the style tranfer was produced from choosing the "vegetation" and "terrain" segments from the semantically segmented road photograph, and using "The Poet's Garden" as the style image.
Here, the outcome of the transfer can certainly be said to be visually pleasing, as the vegetation



(a) Content

(b) Style



(c) Terrain

(d) Vegetation

(e) Output

Figure 6.27: Example of a successful style transfer outcome using colour segmentation

elements of the painting replaces that of the photograph very naturally.

7. Conclusion

This project has been the subject of a few main accomplishments:

- Style transfer compute system.
- Web application.
- Server deployment.
- Image processing techniques merging.
- Provision of research facility tool.

In essence, the design of a style transfer compute system was realised, and tested at many levels. In order to use it more quickly and efficiently, a web application to serve as a user interface was developed, then deployed to a cloud server to take advantage of more powerful compute capabilities. In designing the system, several techniques of image processing (segmentation, histogram specification / equalisation) and ideas of different style transfer related papers were merged together to offer multiple more use cases. The deliverable as currently developed could be considered as commercial application as much as a research tool. Indeed, for researchers an extra research facility tool was provided in order to more efficiently analyse and keep records of all the tests accomplished.

Regarding the general deliverables of the project, mainly involving developing an optimised style transfer system, it can be safely stated that they were met, as conveyed by the resuts obtained. It was shown how combining and tweaking different parameters can lead to various aesthetic creations, which can be shaped as pleases the user's taste.

However, there are still existing issues in the accuracy of certain segmentation methods. This is particularly true for the semantic segmentation case, where not every image could be correctly segmented according to the known labels by the trained network. Additionally, the semantic segmentation dit not perform well at all on artistic images (i.e. paintings), even though the human perception of basic objects was very clear. While this flaw agrees with neural network theory and the provided trained network, it did prevent the option of a semantic segmentator to be run for style images as well, which incurred some limitations in the system functionalities. Moreover, it was discovered throughout a few tests that the initial idea of having an automated pairing of the masks' regions for the threhsold and colour segmentation was not necessarily advantageous (though debatable in some cases), as the standard neural style transfer algorithm already takes care of matching colours and brightness levels between content and style images where possible. Therefore, these added features might be considered as useless functionalities for most cases. On the other hand, threshold and colour transfer processes were found to be fruitful in the case where the *user-defined region* option was ticked, which allowed to manually select arbitrary segments of the mask.

With regards to further work, there are a few possible changes and additions that would enhance the system.

Starting with the semantic segmentation, one could train the PSPNet network further in order to handle more labels and possibly even labels in stylistic images. If this can be achieved, an automated pairing of found labels could conceivable in the style transfer process, where objects would inherit the style of their corresponding matched label from the style image.
The colour segmentor model could also be improved, by developing an algorithm which would automatically determine which of the content or style image to choose as base image (when *user-defined regions* is false) in order to maximise the matched coloured region pairings.
An additional option related to segmentation would be to allow the type of segmentation to be chosen separately for the content and for the style (*user-defined regions* would then naturally be forced to true). This could solve the issue where for instance the semantic model is unable to identify the sky in a painting. Instead, a semantic segmentor could be used for the photograph, and a colour segmentor could be used to artwork. The subsequent transfer from sky to sky regions would therefore be possible. Finally, other segmentation methods could be explored, as there exist many more within image processing techniques.

The general system's algorithm could be improved at certain levels. Firstly, more image preprocessing and post processing could be added to ameliorate results. This might for example include averaging the resulting image at boundaries of chosen segments to alleviate harsh transitions between stylised and non-stylised regions. Secondly, it is known that the neural network algorithm can be initialised with a content image with added random noise, in order to randomise the outcome of the style transfer process. Indeed, the current system will always produce the same result, given the same inputs and same parameters. Whilst it was sufficient and helpful just for the purposes of testing parameters, enabling this extra option would be a trivial addition and would expand the range of possible results by a substantial amount.

Finally, some further work can be undertaken for an enhanced user experience. For instance, instead of just providing a `zip` download functionality, one could imagine an extra summary page that would show all the results produced by the user, with insightful data next to them such as the date, the transfer type, the list of chosen options and essentially everything that is currently included in the `zip` file. Extra-information that this summary page could display might also include graphs of running time, CPU/GPU usage, etc... It is worth noting that this summary page feature would require the design and implementation of a database management system, and an authentication process.

8. User Guide

The code to run the tests and simulations of this project is available at the following address: `https://github.com/caoanle13/FYP/releases/tag/v1.0.0`.

8.1 Adding pre-trained CNNs

- Download the source code and uncompress it
- Download the semantic segmentation network `pspnet.zip`, uncompress it and place it under `root/code/segmentation/`
- Download the style transfer network `imagenet-vgg-verydeep-19.mat` and place it under `root/code/stylisation/`

8.2 Installing dependencies

In order to run the application, a few Python packages (listed in `requirements.txt`) need to be installed first[1]

- First open `requirements.txt` and uncomment one of the two lines for `tensorflow`. Uncomment line 35 if you do not have a GPU, and line 36 if you have one. Save the changes and close the file
- Create a new Python virtual environment called venv where the packages will be installed: `virtualenv venv`
- Activate the newly created virtual environment: `source venv/bin/activate`
- Run the command `python3 -m pip install -r requirements.txt`. This will install all the required packages that the app needs

8.3 Running the App

- To run the app move inside the `code/` directory and run `python3 app.py`
- If run locally, open up a web browser at the address `localhost:8000`
- If run from a server, use the server's external IP address

Enjoy style transfer!

[1]This section assumes Python 3 is already installed, along with `pip` and `virtualenv`.

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015.

[2] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, June 2016.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[4] Georgie Rastall. How are smartphones shaping the future of photography? `https://www.digitalrev.com/article/how-are-smartphones-shaping-the-future-of-photography`. Accessed: 24/01/2019.

[5] `https://prisma-ai.com/`.

[6] Alexei Efros and William T. Freeman. Image quilting for texture synthesis and transfer. *Computer Graphics (Proc. SIGGRAPH'01)*, 35, 07 2001.

[7] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. In *SIGGRAPH '05*, 2005.

[8] O. Frigo, N. Sabater, J. Delon, and P. Hellier. Split and match: Example-based adaptive patch sampling for unsupervised style transfer. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 553–561, June 2016.

[9] Michael Elad and Peyman Milanfar. Style transfer via texture synthesis. *IEEE Transactions on Image Processing*, 26:2338–2351, 2017.

[10] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.

[11] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. *CoRR*, abs/1604.04382, 2016.

[12] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. *CoRR*, abs/1603.03417, 2016.

[13] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. *CoRR*, abs/1611.07865, 2016.

[14] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[15] Dianne P. O'Leary. Robust regression computation using iteratively reweighted least squares. *SIAM Journal on Matrix Analysis and Applications*, 11(3):466–480, 1990.

[16] Eduardo S. L. Gastal and Manuel M. Oliveira. Domain transform for edge-aware image and video processing. *ACM Trans. Graph.*, 30(4):69:1–69:12, July 2011.

[17] Javier Portilla and Eero Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40, 10 2000.

[18] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, December 1997.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[20] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[21] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[22] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

Appendix

Please find below all the content and style images used in this study.

Figure 8.1: Photograph: "Tubingen". File name: `tubingen.jpg`. Dimensions: 900×599.

Figure 8.2: Photograph: "House". File name: `house.jpg`. Dimensions: 1545×1030.

Figure 8.3: Photograph: "Lena". File name: `lena.jpg`. Dimensions: 512×512.

Figure 8.4: Photograph: "Lion". File name: `lion.jpg`. Dimensions: 720×900.

Figure 8.5: Photograph: "Desert". File name: `desert.jpg`. Dimensions: 612×403.

Figure 8.6: Photograph: ”Road”. File name: `road.jpg`. Dimensions: 1280×720.

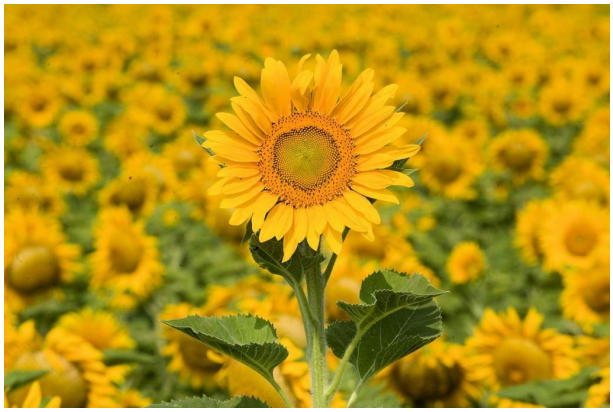


Figure 8.7: Photograph: ”Sunflower”. File name: `sunflower.jpg`. Dimensions: 1024×683

.

Figure 8.8: Photograph: ”Mou Aysha”. File name: `mou_aysha.jpg`. Dimensions: 1024×683.

Figure 8.9: Painting: "Starry Night". Artist: Vincen Van Gogh. File name: `starry_night.jpg`. Dimensions: 1280×1014.

Figure 8.10: Painting: "The Shipwreck of the Minautor". Artist: Joseph Mallord William Turner. File name: `the_shipwreck_of_the_minautor.jpg`. Dimensions: 1200×846.

Figure 8.11: Painting: "Der Schrei". Artist: Edvard Munch. File name: `der_schrei.jpg`. Dimensions: 940×1198.

Figure 8.12: Painting: "Children on the Beach" (unknown). Artist: Unknown. File name: `children_on_the_beach.jpg`. Dimensions: 1920×1440.

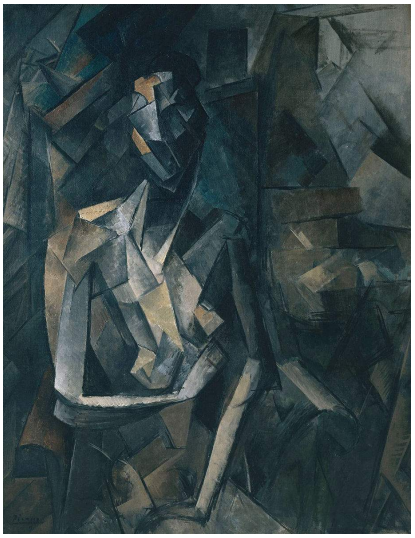


Figure 8.13: Painting: "Femme Nue Assise". Artist: Pablo Picasso. File name: `femme_nue_assise.jpg`. Dimensions: 919×1197.

Figure 8.14: Painting: "composition VII". Artist: Vassily Kandinsky. File name: `composition_VII.jpg`. Dimensions: 520×342.

Figure 8.15: Painting: "The Poet's garden". Artist: Vincent Van Gogh. File name: `the_poets_garden.jpg`. Dimensions: 1686×1339.

Figure 8.16: Painting: "Wheatfield with Crows". Artist: Vincent Van Gogh. File name: `wheatfields_with_crows.jpg`. Dimensions: 1279×614.

Figure 8.17: Painting: "George Moore". Artist: Walter Richart Sickert. File name: `george_moore.jpg`. Dimensions: 1268×1536.