Investigation of Style Transfer on Digital Images

by

Julia Patterson, B.A.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

August 2012

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Julia Patterson

August 29, 2012

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Julia Patterson

August 29, 2012

Acknowledgments

I would like to thank my supervisor, John Dingliana, for his support and advice in this project. I would also like to thank the various coffee shops around Dublin who allowed me to camp out for days at a time while writing this, and anyone who proofread my drafts for me.

Julia Patterson

University of Dublin, Trinity College
August 2012

Investigation of Style Transfer on Digital Images

Julia Patterson
University of Dublin, Trinity College, 2012

Supervisor: John Dingliana

Style transfer is the transfer of characteristics — such as hue, saturation, and brushstrokes — from one image to another. Applications of style transfer range from the educational to the whimsical, with potential uses including documentaries, simulations, and even mobile photo-editing applications. Research in the field is expanding, but works tend to focus on very specific aspects of style transfer: transferring one characteristic, or replicating one particular style of art. A unified framework which combines many disparate style transfer goals would be useful for many situations, but does not yet exist.

This project investigates the current state of style transfer in order to determine whether creating such a framework is possible at this time. In addition, some features have been implemented in an application to demonstrate what is achievable. Preliminary successful results were attained in the areas of colour transfer and manual brushstroke style transfer. Research indicates that automatic brushstroke detection and application, arguably essential for style transfer, are still unsolved problems. We

conclude that while it is conceivable to create a framework combining many diverse style transfer goals, a complete framework is still not possible at this time.

Contents

List of Figures

Chapter 1

Introduction

Computer graphics in its earliest form appeared in the late 1950's, soon after the invention of computers themselves. One of the earliest innovations was a piece of software called "Sketchpad," which used a light pen to create images on a screen, composed of basic lines and shapes. The following decade saw some progress in the field of computer graphics, but it was not until the formation of a graphics group at the University of Utah in the mid 1960's that computer graphics in its modern form began to take shape. In the sixties and seventies, this group was responsible for most of the important developments of graphics in use today, such as rendering, shading, lighting, and texture mapping.

Around this time, researchers began exploring the use of computers and algorithms in creating art. Several computer art shows were held in the late 1960's, which helped to bring attention to and legitimise the field [4]. In one very early example, from 1968 [5], researchers demonstrated the use of sine curves to create a human face, as well as images created from the random distribution of lines and figures. The authors pontificate on how computers will fundamentally alter the way artists think, and the "terrifying and exciting"[5] possibility that computers might eventually overtake artists in their work.

Much progress was made in both 2D and 3D graphics over the next few decades. Since the sixties, technology has rapidly improved. Freed from the mainframes of the mid-twentieth century, the field of computer graphics began to grow explosively. In the 1980's, computer graphics began appearing in advertisements and films, and software

started becoming much more available to the general public. Since the earliest days of computer graphics, thousands of papers have been published and special interest groups such as SIGGRAPH have been established. Today, computer graphics are widely used for educational, entertainment, and advertising purposes. The technology has become so ubiquitous that one can even alter photographs and create digital art using a mobile phone.

1.1 Motivation

In the world of computer graphics, it is often desirable to alter the overall appearance of an image, whether to imitate a well-known style of art, or to transform the image in a completely novel way. The alteration may involve changing the hue, saturation, contrast, type of brush stroke, or even the photo-realism of the image, in a process which is often termed "style transfer." While the term has no official definition, it is commonly accepted to refer to the application of characteristics from one image to another, or to the transformation of the image so it mimics a specific style of art. As the field of computer graphics has developed, much research has been done in the fields of non-photorealistic rendering and style transfer. However, this research generally only focuses on very specific areas: changing one characteristic, such as the image's colour space; imitating one particular style of art, for example, pointillism or cartoon shading; or trying to reproduce a physical medium, like watercolour.

Of course, images can be modified simply for aesthetic reasons; this is evidenced by the widespread use of photo-editing software such as Adobe Photoshop and the popular application Instagram. Images are incredibly important for telling stories, capturing moments, preserving history and culture, and educating the public. Alternately, they can be used to simulate fantasy worlds or the future. Style transfer could also be applied to a 3D rendering, image, or film to give it the style of some pre-existing image or film. In this way, images could be nearly seamlessly blended together in a coherent style to be used in a film or documentary.

Rather than having to use a series of separate applications, it would be very useful to have a cohesive system which combined many of these elements together, thus facilitating the process of style transfer. The goal of this project was to investigate the current state of style transfer and determine whether it would be possible to create such

a framework, and if so, discover which steps must be undertaken. Another goal was to build a prototype to demonstrate a few characteristics that are possible to include in such a system at this time, and to determine how the framework could be extended to encompass further style goals in the future.

1.2 This Report

Because this project is heavily dependent on concepts from the art world, colour theory and style theory will first be discussed, in an attempt to frame some of the main considerations of style transfer. This will be followed by a comprehensive overview of existing research in various subfields of style transfer, including colour transfer, texture transfer, and brushstroke transfer.

The next chapter will contain a description of the design for the project, which choices were made, and why. The report will then go into detail about the project's implementation itself and results achieved. Finally, we will discuss conclusions, techniques which were not possible to include, and any other suggestions for future work.

Chapter 2

State of the Art

2.1 Art Theory

2.1.1 Defining Style

The Oxford English Dictionary defines style as "the manner in which a work of art is executed, regarded as characteristic of the individual artist, or of his time and place" [7]. Often, it is difficult to precisely say what makes a style unique.

It is generally accepted that the formal analysis of art is split into several main categories such as line, colour, shape or form, and texture. Line is defined as "an identifiable path created by a point moving in space"[16]. It describes both the direction (horizontal, vertical, diagonal, curved) and width of lines in the painting. Colour includes hue, the actual "colour" of an object, such as red or blue; value, how light or dark a colour is; intensity, which is the "dullness" of a colour; and warmth, e.g. a colour tending towards red or yellow is considered "warm," whereas a blue-tinged colour is considered "cool." Shape and form describe the actual geometry and depth of an artwork. Of course, there are many other features which can be used to categorise art, but these are several of the most common.

In terms of analysing art with a computer, style can be broken down into aspects of the aforementioned qualities: overall hue, saturation, contrast, edge definition, and type and size of brushstrokes, to name a few. Brushstrokes themselves can be qualified by their width, length, direction, and smoothness or shape.

Even when all of these features have been extracted from a style, it may not be a

simple task to imitate the style itself, or to quantitatively say what makes works done in the style so appealing. Some studies have been attempted to determine what qualities of an image make it seem beautiful or aesthetically pleasing [6]. These features include overall qualities such as proper exposure, saturation, and "colourfulness", as well as generally accepted "rules of thumb" for composition, such as the Rule of Thirds, and even the proportions of an image [6]. The combination of all of this information is bringing researchers ever closer to the artificial generation of believable art.

2.1.2 Colour Theory

Colour is an entire study unto itself. Colour theory is a way of framing ideas about colour, and determining relationships between colours and how they can be combined to create various effects. There are different ways of mixing colour: additive mixing, where colours add together to form white, as light does, and subtractive mixing, where colours darken when mixed, as with paints and inks. There are many different theories about the ideal ways colour should be mixed, combining scientific observations and aesthetic considerations.

There are also numerous colour spaces with which to represent colour. In a colour space, all colours can be represented as fitting on a series of axes, or components. The most well-known is RGB, which is an additive colour space with three axes, red, green, and blue. Any visible colour can be represented as a number between 0-255 in each dimension. For example, a colour with an RGB value of [255,100,0] translates to orange. An example of a subtractive colour space is CMYK (cyan, magenta, yellow, black), which is used in printing.

Several other colour spaces are based on human perception. Research done on human vision has shown that there are cones in the eye which process different bands of colour. In 1802, a researcher named Thomas Young proposed the trichromatic theory, stating that there were three receptors in the eye to process colour [11]. Further research supported this theory, but could not explain all aspects of human perception. In 1872, Ewald Hering proposed the new opponent colour theory, which extends the theory and explains how the brain actually processes colour. In this theory, there are three channels: red-green, yellow-blue, and white-black. Each side of an axis is at the opposite end of a spectrum from the other, and both colours cannot be perceived at

the same time [11]. For example, a colour could be described as reddish-yellow, but never greenish-red.

Examples of this type of colour space are l$\alpha\beta$ and YC_BC_R, both of which will be discussed more in Section 2.2.

2.2 Style Transfer

2.2.1 Colour Transfer

Colour transfer involves replicating a colour scheme from one image onto another. This process can be applied to correct undesirable lighting, evoke certain emotions, or simply to make an image more visually interesting. There are many solutions to the colour transfer problem, falling into one of two categories: global transfer and local transfer. Global colour transfer is a process in which the colour scheme from a new image is applied over the entire target image, whereas an image using a local colour transfer algorithm can have colour schemes from several different images applied to different regions.

One of the earliest, most seminal papers published on style transfer, by Reinhard et al., describes a method for colour transfer. The research presented in this paper was groundbreaking, and is referenced in nearly every other colour transfer paper, as well as appearing in many other places.

Reinhard's method [23] uses a series of matrices to convert a source image and target image from RGB colour space to the l$\alpha\beta$ colour space. As mentioned earlier, this colour space was chosen because it more accurately describes colour in the manner that the human vision system processes it. This allows for more accurate transfers. The successful use of l$\alpha\beta$ colour space here led to its inclusion in later colour transfer research.

In Reinhard's work, once the images are converted to l$\alpha\beta$ colour space, statistical calculations – the mean and standard deviation – are performed, and the source image's pixels' colours are globally shifted by the appropriate amount. Finally, the images are converted back to the RGB colour space.

Many others have since built on Reinhard's work, in order to create a more accurate colour transferring process.

Zhao et al. [34] assert that the l$\alpha\beta$ colour space is inadequate for colour transfer, because it "lacks independent manipulation of [each of] the channels" [34]. Instead, the authors chose the similar oRGB colour space, developed by Bratkova et al [2]. oRGB colour space is similar to l$\alpha\beta$ in that it uses the same opponent colour axes, but allows for adjustments [2, 34]. The process itself is similar to Reinhard's, but also uses histogram matching to maintain the luminance of the original image.

Another paper which disagrees with the use of l$\alpha\beta$ colour space was published by Xiao, et al. in 2006. The authors demonstrated a method for "directly manipulating" the RGB colour space [31]. The method involves some slightly more complicated mathematics, where covariance matrices, eigenvalues, and eigenvectors must be calculated, but it is essentially similar to the Reinhard process (i.e. multiplying by matrices to shift the colour space). The main benefit provided is that the image always remains in RGB space, rather than being transformed to LMS, logLMS, and finally l$\alpha\beta$ colour space. As well, the resulting images are extremely similar to Reinhard's.

Pitié et al. devised a method to be used for correcting an image's lighting [21], for example in film post-processing. The authors critique Reinhard's method because there, the statistical calculations are applied separately to each channel. Here, rather than using straight statistical calculations, the authors apply a probability density function across the image to analyse the colours. Next, a colour mapping is created, which maps each colour in the source image to a target image, e.g. every blue pixel becomes grey [21]. This new colour scheme is combined with the target image's colour gradients, in order to preserve it more accurately.

Further issues were discovered in the Reinhard method by Xiao et al., including colours being transferred to the wrong part of an image, and low fidelity "in terms of scene details and colours" [32]. The authors mention Pitié's method, acknowledging that their method offers more fidelity than prior methods, but bringing attention to the extra edges the method creates[32]. Xiao et al. offer a solution to the fidelity issue in which they combine the colour gradient of the original image with results of a histogram of the style image.

A further colour space which has been demonstrated to be useful in colour transfer is the YC_BC_R colour space. Like Xiao et al above, Li et al. use an extremely similar technique to Reinhard's [14]. The RGB colours are multiplied by a different matrix to the one used for l$\alpha\beta$, which immediately transfers them to YC_BC_R colour space, rather

than going through several intermediates. Like Reinhard, the authors use statistical calculations to skew the existing colours, and those results are then converted back to RGB.

Neumann et al. use a different approach [17], choosing to focus on hue, lightness, and chroma (saturation), rather than a specific colour space. Histogram matching is done on these three properties of a given image, with the goal of preserving hues, keeping them the same before and after the colour transformation.

Colour transfer has also been used to add colour to greyscale image, as in a method by Welsh [29], where a coloured image is converted to l$\alpha\beta$ colour space and sampled, and then its colours are added to a greyscale image based on luminance and texture information. Conversely, colour transfer can be used to remove colour from coloured images, as seen in the work of Wu et al. [30]. This may seem unnecessary, as the technology is quite common nowadays, but Wu's method strives to improve existing technology [30]. Their algorithm analyses the available colour information from the original image to find visual areas of interest, and applies this information in order to increase the contrast in those interesting areas, thus preserving the image's visual appeal.

An example of a local colour transfer algorithm is a scattered point interpolation method, developed by Wang et al. A user draws a few strokes in one or more areas of a source and target image, defining corresponding sections [28], e.g. a pink flower in a source image is highlighted, as well as a red rose in the target image. In addition, the two images' differently coloured leaves are highlighted in a corresponding pair. Once all pairs in the images are defined, the algorithm scans neighbouring pixels of the highlights until it reaches the boundary of the selected area. The difference between the colours in the two images is calculated to decide how to alter the pixels in the target image, and finally, the new colours are interpolated through the full selected area.

2.2.2 Texture Transfer

Colour is not the only characteristic of an image which may be transferred; often it is desirable to copy the texture of the image, in order to mimic a type of brushstroke or medium, or simply to add visual interest to a new image. In addition, texture transfer

can be used to create a large, believable texture from a small sample [8]. Like colour transfer, there are countless proposed methods for texture transfer; these can be split mainly into the categories of pixel-based and patch-based transfer [33].

Often, these texture transfer techniques provide a reasonable approximation to the desired texture, but are not believable enough to the human eye. Some degree of randomness added to the algorithm can increase believability [33]. Xie et al. took the process a step further by creating "Feature-Generated Texture Synthesis", in which so-called "feature fields" marking the important edges of an image are explicitly defined. The edges are determined by analysing the differences between pixels in an image; adjacent areas which differ greatly in colour value are an edge. The process then creates an image pixel by pixel, deciding whether the pixel should be more similar to the source or target image[33]. Images with fine lines are recreated more sharply when using the FGTS method, both when the fine lines were pre-existing in the original image, or when the resulting image contains a sketched or pen-and-ink texture [33].

Some researchers, such as Eisenacher et al., have experimented with creating large textures from any source [8], whether it is irregularly-shaped or distorted, or somewhat low-quality. The completed procedure may also be used to remove artefacts or objects from photographs, and cover the spaces with a continuation of the bordering texture. The technique can even be used to create convincing textures on angled planes. A user employs a tool in order to approximately define the basic geometry of the photograph. Once the geometry is defined, the user selects another area to which the texture should be applied. Based on the curvature of the geometry and neighbouring pixels around the area of interest, the program then determines how to fill in the selected area.

Similarly, Nguyen et al. automatically extract materials from a given image and place them onto a 3D environment [18]. Using a series of algorithms, the original image is split into diffuse, specular, and texture components. The resulting data is used to create materials, which are then assigned to similar objects in the 3D environment, based on statistical cost-based functions.

Colour and texture can be transferred not just to a 2D image, but also to 3D volumetric renderings. Bruckner et al. [3] created a process in which a user selects styles that have been extracted from existing images — including matte, shiny, and sketched samples — and applies them to a 3D image, for example, results of a medical scan. The selected styles are transferred by way of transfer functions, which lookup

what values in a given image correspond to which bits of a texture to be transferred. The method preserves the contours of an image and includes transparency information, thus allowing for more realistic and useful graphics or diagrams.

2.2.3 Transfer of Specific Styles

In addition to colour and texture transfer, many researches have attempted style transfer, which attempts to imitate a specific medium or style of an artist. In this case, both colour and texture are transferred.

Rather than focus on a small, specific type of art, Hertzmann et al. set out to create a general process for automatically extracting and "learning" a style, given an unadulterated image and the same image having been passed through a filter. The two images are compared to learn what has changed, and therefore what would need to be changed in an entirely different image to achieve the same result. This process is called "image analogies." compare the process to the definition of an analogy, A : A' :: B : B' [9]. The image analogies process improves upon existing filters, as it is not limited to a selection of predefined styles. Instead, it is able to replicate a myriad of styles such as "oil, watercolour, and line art rendering," [9] and transfer textures, as well as mimic effects which are perhaps not considered a "style," such as increased sharpness or blurring.

Of course, many attempts have been made to synthesise certain styles of art or famous artists. A large amount of research done on texture-transfer is various attempts to imitate impressionism and pointillism, both defined for the most part by their brushstrokes [12, 19, 27]. The goal can be either to generate brushstrokes in a pattern which is believable and does not repeat, or to address the direction of brushstrokes, as mentioned earlier. The imitation of many specific styles has nearly been solved.

Many other styles have only been researched a small bit. This includes the work by Rosin et al., who introduce a "perceptually minimal rendering model"[24] to create images in a style similar to Andy Warhol's. First, the algorithm finds and smooths the important edges of a given image, and then removes extraneous lines by disconnecting edges. For a more interesting image, the final lines can be represented as black and white on a grey background, and dark blocks and patterns can be added for more

abstraction.

Curtis et al. were able to create a model simulating watercolour painting which is both accurate and very interesting, as it uses physical processes to model the flow of the paint. Physical properties of the simulated paint, paper and water are set, and the program iterates over a cellular-automata-based algorithm which calculates the movement of water and pigment in each timestep. Last, the various layers of paint are combined together in a realistic manner. What is most impressive is that the simulation can mimic a wide variety of brushstrokes and watercolour styles, rather than just a thin even layer of colour. Using the interface, a user can create a virtual watercolour painting manually, or an image can be automatically generated, by first extracting colour data, and then placing brush strokes accordingly.

Some widely-used styles, such as pen and ink, do not require colour at all, and instead are entirely based on texture, in this case, strokes. In the early days of computer graphics, Salisbury et al. created a basic working model of pen and ink [26]. In their model, a user can create a facsimile of a pen and ink drawing by specifying a style of pen stroke, several of which are predefined and stored in the system, and then "brushing over" the area where the texture is required. The system fills in the necessary lines, complete with some randomness in the spacing, length and "scribblyness," to increase realism.

As opposed to the previous example, sometimes style can only involve the automatic transfer of colour. A publication by Qu, et al. describes a method for colourisation of Japanese manga art[22]. Manga, when colourless, is essentially pen and ink illustration, meaning pages are crowded with an infinite number of tiny dots and lines. The new system described here is able to recognize patterns in the details, in order to identify the boundaries of a region. A user can draw a rough line over an area of the manga they wish to colourise, and the pattern matching will fill the area to the edges.

2.2.4 Brushstroke Detection and Transfer

At the moment, a few pieces of research exist on the subject of automatic detection and application of brushstrokes to images. However, the field is in the "early stages of development"[10], and the brushstroke problem is far from solved. Even the papers that do exist are more specific than a general brushstroke detection algorithm, and

are not always successful. Additionally, all of the detection algorithms discussed below have only been used on greyscale image examples.

In terms of detection, Berezhnoy et al. published work which can determine the overall orientation of brushstrokes in an image[1]. Their experiment looks at brushstrokes in oil paintings, and uses a novel algorithm called POET, or Prevailing Orientation Extraction Technique. The authors specifically state that the focus of the work is on extracting texture orientation of brushstrokes, not entire brushstrokes, due to "the difficulty of segmenting individual brushstrokes". The algorithm has two steps: first, filtering, where the image is convolved with a circular filter to enhance contours, and second, orientation-extraction, where the convolved image is processed to produce "binary oriented objects" representing brushstrokes. The largest objects, or brushstrokes, in an image are measured to determine the overall orientation. POET is as successful as humans in determining brushstroke orientation.

Another work done concerning brushstroke detection, published by Johnson et al. [10], focuses on analysing a large amount of brushstroke samples to determine whether a painting is a Van Gogh or a forgery. Like the previous method, paintings are split into many small patches which are analysed individually. Brushstrokes are extracted from the patches using wavelet transforms. The process extracts wavelet-based texture features, as well as coefficients which represent different orientations and abruptness of strokes. Some edge detection is used, to capture the geometric characteristics of strokes (i.e. length, orientation, and average curvature). The authors note that it is "extremely challenging to locate strokes accurately from greyscale images in a fully automated manner" and the process is additionally challenging "due to the intermingling nature of brushstrokes". Once all of the information has been extracted, features are clustered to find the patches that are most similar to one another. Dozens of images were needed to train the program, and results were not entirely accurate. While the program was able to accurately identify many real and forged Van Goghs, it also incorrectly identified a few paintings. The authors describe the results as "encouraging but not perfect"[10]; more work must be done before brushstroke information can be easily extracted.

Lettner et al. [13] performed a successful study to differentiate between fluid and dry brushstrokes in images. They tested three methods: statistical features of first order, which calculates the mean and standard deviation of grey values in an image, but ignores the spatial arrangement of them; a co-occurrence matrix, which is a ma-

trix describing the spatial alignment and density of different levels of grey; and, like Johnson et al, discrete wavelet transforms of the brushstrokes. Of these three methods, wavelet transforms were most successful at identifying whether a stroke was fluid or dry. Overall, results varied, from 65.3% up to 90% accuracy. A major limitation with this study is that wet and dry strokes can be differentiated, but it is not yet possible to differentiate the type of media used in the stroke. Although the process is fairly successful, the actual information conveyed is not widely useful.

Some studies have also been done regarding automatic application of brushstrokes to an image. Lu et al. created a method for applying painterly brushstrokes to an image, but the parameters (length, width, brush type) must still be selected by the user. Previously, brushstrokes had been simulated in a variety of ways. Spline curves have been used to represent long, continuous strokes, but perform badly. 3D modeled environments can use 3D particles based on the scene's geometry to represent strokes, but this does not work for 2D environments. Several other works have based brushstrokes on local features of an image and so are computationally expensive [15]. Lu's method places each stroke individually, regardless of other strokes, which the authors assert is a great improvement over the existing methods.

In Lu's work, each stroke is represented by a texel, with its information (angle, center point, and short and long radius) stored in a multichannel 2D texture. First, the image is passed through a low-pass filter and the image gradient is calculated. The gradient determines which direction strokes will be placed in. Next, a stochastic stroke placement process takes place. The image has several layers of strokes applied to it, each with a different probability of strokes being placed on each texel. For the coarse layer, fewer, larger strokes are placed, and as the layers become finer, strokes have a higher chance of being drawn. This information is passed to a geometry shader, which draws the strokes as rectangular point sprites. Brush texture and an alpha mask are applied at this time. Finally, lighting is applied, based on a height map. This method is able to draw strokes in a somewhat believable manner, but it is only capable of determining the hue and direction of a brushstroke. In order to accurately imitate a style, more information is required, such as length, width, transparency, and even texture. These parameters are not automatically detected, and must be defined by the user.

According to this sampling of research, it is very difficult to analyse brushstrokes.

Any research which has been done has required vast sets of data, or has only been able to return minimal information. The most successful stroke detection can only return information regarding the orientation or wetness of a stroke. Automatic application of strokes is somewhat possible, but most properties still need to be selected manually, regardless. Due to such incomplete state of research in the subject, it was not possible to automatically extract brushstrokes in this framework. Instead, a few examples of manual style transfer were implemented.

Chapter 3

Design

The goal of this project was to determine whether a universal shader-like framework could be developed to combine many different aspects of style transfer onto one image, in opposition to the scattered research which currently exists.

3.1 Selecting a Platform

Many computer languages can be used for image processing, from very low-level C to high-level languages such as Python. A plethora of libraries, such as OpenCV, also exist to assist in image manipulation.

C# is a high-level, object-oriented language developed by Microsoft since 2001. It is used in a variety of settings, including developing .NET applications and Windows graphical user interface forms, and, in conjunction with the Microsoft XNA library, creating games. It was chosen for this project for several reasons: the ease with which a graphical user interface (GUI) can be created, its abundant selection of built-in libraries, including vector and matrix operations, the fact that it is well-supported, and because speed and optimisation were not a factor in this project.

Originally, the project consisted of a single form application. As more features were added, the application became somewhat unwieldy. Eventually, a command line version of the program was also created, in order to be used in batch processing. At this point, most functionality was split out of the main program and placed in a library of its own, which is called by both the GUI and command-line applications.

3.2 Choices Made

After considering all the elements vital to style transfer, it was decided that colour was the most fundamental and necessary characteristic. Therefore, this was the first part of the process that was undertaken. Many, many different methods exist for altering the colour of an image. The majority of these research methods are based on groundbreaking work by Reinhard [2001], with the basic formula altered to suit more specific needs. It was unknown what problems might be encountered when performing colour transfer on a given set of images, so implementing Reinhard's transfer in $l\alpha\beta$ colour space seemed to be an appropriate place to begin.

The Reinhard process presented problems, returning invalid results. After an unsuccessful period of time making adjustments to the code, a new search was undertaken for a different colour transfer method. Li et al.'s proposition of using the YC_BC_R colour space was then selected. This colour transfer process is quite similar to that of Reinhard, but requires fewer operations, so it is simpler and there is less room for error. YC_BC_R colour transfer was successful across numerous types of image.

As a small adjustment, brightness and contrast sliders were added to give more control over the output image.

The addition of automatic brushstroke detection and transfer would ideally be the next feature involved. Though highly desirable, it was shown to be infeasible for this framework, due to a lack of research in the subject, as discussed in Chapter 2. Instead, it was thought that images could be compared to a sort of database of model style images, which would then return the style which a given image was most likely to be. To do this, several image comparison metrics were investigated. In the end, the chi-squared distance algorithm was chosen because it is not overly complex, and yet still performs fairly accurately.

Finally, a full set of style transfer algorithms could not be implemented, but one in particular was chosen as a demonstration of what could eventually be implemented in the future. The style chosen was Impressionist or Pointillist — a very familiar style in which many large dots or strokes are used to partially abstract an image but convey the feelings of the scene.

Chapter 4

Implementation

4.1 Colour Transfer

As discussed earlier, colour is probably the most essential characteristic of an image, immediately perceived by humans and able to convey an entire "feeling" to an image on its own. Of the many methods invented to transfer colour, Reinhard's process involving $l\alpha\beta$ colour space was one of the first, and most-widely referenced, so it was chosen to be used in the framework.

The reason for converting to $l\alpha\beta$ colour space rather than just altering the RGB colour space of an image is that in a RGB image, "there will be correlations between the different channels' values. [...] If we want to change the appearance of a pixel's colour in a coherent way, we must modify all colour channels in tandem" [23]. Instead, Reinhard et al. wanted "an orthogonal colour space without correlations between the axes." The colour space they selected was the $l\alpha\beta$ space, discussed in Chapter 2. The three channels in $l\alpha\beta$ are l, or luminance/lightness (white vs. black), and the colour axes α and β, which represent red-green and yellow-blue, respectively.

Returning to the algorithm at hand, RGB is converted to $l\alpha\beta$ by multiplying by a series of matrices. First, the RGB pixels must be passed through two intermediate colour spaces: XYZ colour space and LMS colour space. The matrices used to convert from RGB to XYZ and then to LMS are combined into one simple matrix

multiplication:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.5783 & 0.0402 \\ 0.1967 & 0.7244 & 0.0782 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The log of L, M, and S are then taken to eliminate skew. Next, the coordinates of the matrix are rotated using another matrix multiplication to enter l$\alpha\beta$ colour space.

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} log(L) \\ log(M) \\ log(S) \end{bmatrix}$$

Reinhard refers to the original image as the "source image," and the image with the desired style the "target image." These names are not entirely intuitive, so henceforth these shall be referred to as the "original" and "style" images. The first step in the transfer is to convert both of the images to l$\alpha\beta$ space. Once this been achieved, it is necessary to calculate some statistics — the mean and standard deviation for each axis of each image. To produce the resulting image, the first step is to subtract each of the style image's means from each of the original image's means (e.g. subtract the mean L of the style image from each pixel's L-value in the original image). The pixels are then scaled by multiplying the newly-calculated l, α, and β values (now known as l*, α*, and β*) by the standard deviation of the original axes over the standard deviation of the style axes.

To explain more easily:

$$l' = \frac{\sigma^l_{style}}{\sigma^l_{orig}} l^*$$

$$\alpha' = \frac{\sigma^\alpha_{style}}{\sigma^\alpha_{orig}} \alpha^*$$

$$\beta' = \frac{\sigma^\beta_{style}}{\sigma^\beta_{orig}} \beta^*$$

Finally, each mean of the style image is added to the pixels of the original image.

This ensures that the pixels have been shifted into the colour space of the style image, while maintaining the deviations from the mean they had prior to alteration. After this shifting has occurred, the images are converted back to RGB colour space with the following matrix multiplication:

$$\begin{bmatrix} log(L) \\ log(M) \\ log(S) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & -2 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{3}}{3} & 0 & 0 \\ 0 & \frac{\sqrt{6}}{6} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix}$$

Next, the log(L), log(M), and log(S) values are raised to the power of ten "to go back to linear space" [23, 4]. The final multiplication required is:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4.4679 & -3.5873 & 0.1193 \\ -1.2186 & 2.3809 & -0.1624 \\ 0.0497 & -0.2439 & 1.2045 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

When implemented in the application, this algorithm consistently returned values far out of the RGB spectrum range of 0-255. Clamping was used in order to avoid this issue. While the algorithm did transfer colour from image to image, more research was done to determine if there was a "better" algorithm that could be used. As seen in Chapter 2, many alternatives to the Reinhard l$\alpha\beta$ method exist. Colour transfer in YC_BC_R colour space, as suggested by Li et al., was chosen as a replacement. Like l$\alpha\beta$ colour space, the three channels represent luminance, and approximately a red-green axis and a blue-yellow axis. Officially, C_B represents blue minus luminance (B-Y) and C_R represents red minus luminance (R-Y) [14].

The YC_BC_R method is based on Reinhard's work, but was designed to be simpler and faster. Rather than the series of matrix multiplications Reinhard's method requires, Li's method only requires one, and additionally does not require any logarithmic operations. Because our application in its current state is still a prototype, speed is not of the utmost importance. However, the simplicity of the YC_BC_R algorithm made it a wise choice, since fewer intermediate conversions means less chance for errors to occur.

The RGB to YC_BC_R matrix conversion is as follows:

$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1140 \\ -0.1687 & -0.3313 & 0.500 \\ 0.5000 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Other than the conversion to the new colour space, the rest of the algorithm is identical to that of Reinhard: calculating the mean and standard deviation for each image, and applying the same subtractions and multiplications as above.

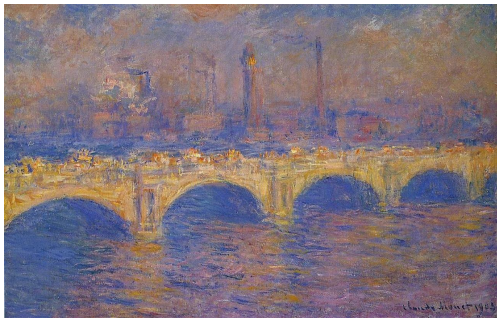
4.1.1 Results of Colour Transfer

The YC_BC_R method can be used to transfer colour to and from paintings, photographs, and 3D rendered images. Like Reinhard and Li explain in their respective works, not every image works well with every other image. Reinhard explains that "the result's quality depends on the images' similarity in composition" [23]. This was discovered to be true, though images need not be of similar scenes to achieve successful results.

Monochromatic and greyscale images behave differently to colour images when passed through the transfer filter. Using another monochromatic or greyscale image for the style image transfers well. Since the transferred colour is based on the mean of the style image, the tone of that image is simply passed onto the original image, as expected. Transferring a colour image onto a monochromatic image does not perform as well. The very same principle which lends itself so well to monochromatic transfer creates problems for colour images.

Because the mean is used to shift the colours, the monochromatic image receives whatever that mean is, and applies it across the whole image. Any details, like small accents of bright red or yellow, are lost as they blend into the mean. However, luminosity (Y axis) is transferred very accurately. A high standard deviation in luminance will lend more contrast to the original image; a low standard deviation will dull the image. The mean luminance simply makes the overall original image lighter or darker. On occasion, a brightly coloured image, in particular one with a high standard deviation in the C_B or C_R axis, will create an image which appears to have two colours (see Figure 4.2). In these situations, one colour will tint the highlights of an image, and its opposite will tint the shadows. That is, an image with a high standard deviation in its blue-yellow (C_B) axis may appear to have bluish shadows and yellow highlights.

(a) Monet, *Banks of the Seine, Vétheuil* [2]



(b) Monet, *Waterloo Bridge* [1]

(c) Colour transferred from (b) to (a)

Figure 4.1: Example of colour transfer between two Monet paintings.

Transferring from one colour image to another also provides different results. As Reinhard and Li stated, using radically different images may provide undesirable results. One example is that images which appear dark all-over (have a lower mean luminance), such as Van Gogh's "Starry Night," or a photo taken of Dublin at night with a dark sky, do not accept transfers very well, especially from images with a higher mean luminance. Highlights in the images become more evidently altered by the style image, but generally, the transfer can be considered unsuccessful.

A hypothesis was proposed that perhaps the standard deviation affects the quality of the results, specifically that images with standard deviations closer to each other will transfer better. Statistics were calculated for one arbitrarily chosen image that seemed to produce "good" transfers against several others. In this small set of results, there was minimal evidence that smaller differences in standard deviations produced

better results. This evidence only appeared when ignoring the standard deviations of the luminance axis. Even then, a few pieces of conflicting evidence appeared. The Charing Cross image, which has a larger difference in standard deviation than the other "good" images in the set, provides a pleasing result. Conversely, the Leenane image has a relatively low difference, yet performs poorly.

It is possible that when performed over a very large set of results with more significant statistical analysis, some patterns would emerge, but it is very difficult to analyse manually. Patterns could be in the standard deviations, means, specific axes of these, or something completely overlooked. In addition to some hypotheses proposed above, such as differences in content, like a dark sky, it is most likely that what constitutes a "good" or "bad" image is entirely subjective.

Because images all have such different characteristics, it is difficult to determine whether a "successful" transfer is due to statistical information alone or to human perception. A transfer may have been successful in terms of being faithful to the colours in the style image, but a human may decide that the same image looks unrealistic or "strange" in some way. Paintings with unrealistic colours are, of course, more acceptable than photographs, since they need not conform to reality. Conversely, photographs are likely to appear strange, since a human will tend to compare the photograph to a real-world setting.

Some preliminary examples of calculated statistics appear below; note their inconclusive results. The columns represent the style image in question, the difference in standard deviation between the original and style images for Y, C_B, and C_R, the total difference, and the total difference, ignoring Y.

Image	$\triangle\sigma$ Y	$\triangle\sigma C_B$	$\triangle\sigma C_R$	Total $\triangle$	$\triangle C_B C_R$
Original	48.22	16.56	9.22	–	–
Waterloo	27.62	20.22	7.11	26.34	5.77
Soleil	22.61	2.17	3.77	28.56	5.95
Waterlilies	29.84	11.85	7.23	25.10	6.70
Charing Cross	11.85	6.54	4.58	51.06	14.66

Table 4.1: "Good" matches

Image	$\triangle\sigma$ Y	$\triangle\sigma C_B$	$\triangle\sigma C_R$	Total $\triangle$	$\triangle C_B C_R$
Original	48.22	16.56	9.22	–	–
Leenane	25.52	3.82	3.17	32.51	6.99
Stephen's Green	79.32	7.51	7.47	41.91	10.80
Picasso-blue	33.48	8.88	12.37	25.57	10.81
Galway	67.14	31.90	17.81	42.85	23.97

Table 4.2: "Bad" matches

4.1.2 Batch processing

In order to automate the process of transferring approximately seventy images to every other image, a command-line version of the application was created, which accepts filenames for two input images and the output to be saved to. Two batch files were written to call the application with given parameters. Two separate batch files were created because batch was not designed to easily interpret double for-loops.

Transfer2.bat gets every file in a given folder and passes the filename as a parameter to transfer.bat, one by one.

```
for /f "delims=" %%f in ('dir /b [input filepath]') do transfer.bat %%f
```

Transfer.bat takes the file that was passed as a parameter from transfer2.bat and uses it in place of variable %1. Like transfer2.bat, it goes through the given input folder and for each file in that folder (%%f), applies a colour transfer using the parameter (%1) passed to it. It saves the stylised image to an output folder with a filename which concatenates the two images' names, for easy identification later.

```
for /f "delims=" %%f in ('dir /b [input filepath]') do
colortransfercmd [input filepath]\%%f [input filepath]\%1 [output path]\%%f-%1
```

Running this code over sixty-nine images resulted in 4,761 colour-transferred output images. The process took just under three hours to complete, or approximately twenty-eight images created per minute.

4.1.3 Video

The colour transfer process was also successfully applied to a video. The video in question was filmed in 1976 while driving through Dublin, beginning near Grafton Street. The file was split into successive frames using a command line program called FFmpeg, which is used to decode, encode, and otherwise alter media files. Once split, the same batch transfer was used on every frame. The frames were then recombined into a new video, again using FFmpeg. Results of this transfer were successful; the new video has cohesive colour transfer throughout the frames with no flickering. For the low-resolution 640x480 video we were processing, transferring colour to 213 frames took approximately six minutes to complete.

As an example, an Impressionist style filter was applied to the same video, post-colour-transfer. This process took twenty-one minutes, three and a half times as long colour transfer alone. There does not seem to be any "shimmering" in the video, but it is hard to tell, as the quality of the film itself is somewhat grainy.

Figure 4.5: Still frames of video, unaltered vs. with style transfer [7]

4.2 Style Identification

Since research suggested automatic detection was most likely not possible, it was decided that a first step would be to compare an image to a group of "model" images, each which would represent a style of art. Ideally, it would be possible for the program to look at an image and see its overall hue, contrast, and such characteristics, and

decide that the image is most similar to a Mondrian, or a Van Gogh, or Picasso in his blue period, and act accordingly.

Research was then done to see how best to compare the similarity of images. It originally appeared that an algorithm known as "Earth mover's distance" would be most appropriate. As investigated in Rubner's work as a "metric for image retrieval," it was the most successful of five algorithms — along with L1 distance, Jeffrey divergence, chi-squared statistics, and quadratic-form distance — at returning images which were most similar to each other [25]. Earth mover's distance was nearly selected, but the algorithm involves many steps, including solving a transport equation, and was deemed too complex and time-consuming for the precision required for this preliminary undertaking.

Because ultimate precision was deemed unnecessary, the simpler, less-precise chi-squared method was chosen instead. The chi-squared method has been "successfully used for texture and object categories classification, near duplicate image identification, local descriptors matching, shape classification, and boundary detection"[20]. Simple as it may be, this algorithm can be quite powerful.

In the chi-squared method of the application, the user selects a starting image, as well as a group of any number of images to compare with. A histogram is then created, using a specified number of "bins." Without bins, there would be 255 values for each pixel to fit into, making it difficult to compare several images. Bins allow a smaller number of categories to fit each pixel into, i.e. if there are ten bins, each bin will keep track of the number of pixels in the image that fall into its specific range (0-25, 26-50, and so on). Separate statistics are kept for each R, G, B value. After final tallying, each bin's count is divided by the total number of pixels in the image, to normalise the results.

After bins are calculated, the chi-squared distance is calculated with the formula:

$$\chi^2(P,Q) = \frac{1}{2}\sum_i \frac{(P_i - Q_i)^2}{(P_i + Q_i)}$$

In this case, since there are three variables to consider, all three chi-squared distances are added together. Of all images being compared, whichever one returns the lowest chi-squared value (shortest distance) is the one the algorithm chooses as most similar.

4.2.1 Results

The chi-squared algorithm is often accurate, but does not always return the image that a human would consider to be the most similar. This may be due to human vision being unable to detect the true average colours throughout an image. And, as noted above, the chi-squared algorithm is not the most accurate algorithm possible. The images presented below are an example of a successful chi-square matching. The original Van Gogh at the top was given as the input image, and the images in the row underneath were selected as comparisons. The numbers returned are the chi-squared distances from the input to each of the other images. More similar images return smaller values. These results range from another Van Gogh painting that uses very similar colours with a quite low distance, to a high contrast black and white image, not similar to the original at all, with a high distance.

Figure 4.6: Van Gogh, *Wheat Fields with Reaper*[8]

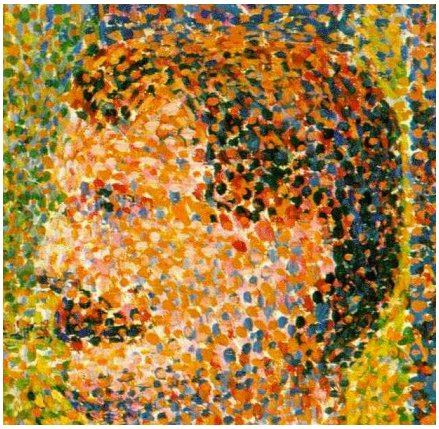
Figure 4.7: χ^2 distances: 0.002743203, 0.002840308, 0.004657236, 0.4484885 [9,10,11,12]

4.3 Style Matching

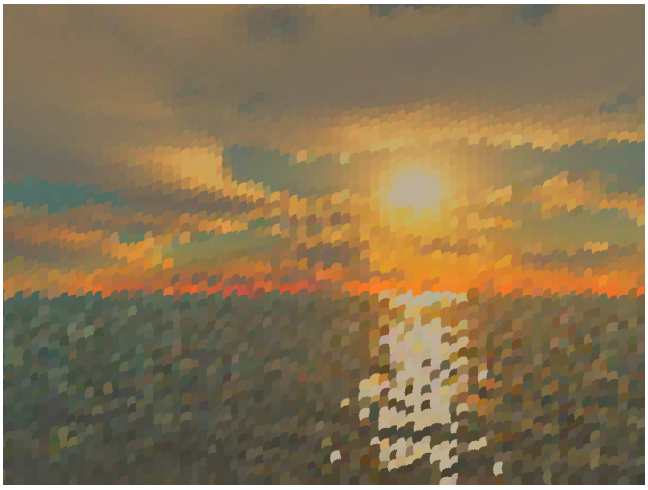
Assuming the image returned by the chi-squared calculation is correct, the program will now be able to apply a style to the original image. If, for example, the program has determined that the given style image is a pencil sketch, it will then be able to apply a sketch-like brushstroke style across the whole image; for a Pointillist painting such as a Seurat, a Pointillist filter would be applied, and so on.

It was not possible to implement every possible style filter, so a Van Gogh-like impressionist brushstroke style was chosen as an example. First, a blank image is created. Next, a random number r is generated, which acts as the increment counter for the pixels. Every r pixels, the current location is saved to a list of points. These pixels are too evenly spaced, so two more random numbers are generated, and the location is shifted in the x and y directions accordingly. This makes the image more naturalistic, since humans tend towards committing slight errors. At each new shifted location, the pixel colour is looked up at the corresponding old location. At the new location, the pixel, as well as every pixel within a certain user-specified brushstroke width, is transformed to that colour. There will be some gaps between the brushstrokes, so it is necessary to perform this operation a number of times to cover the "canvas".

(a) Rendered Sunset [13]



(b) Close-up Seurat[14]



(c) Result of style transfer

Figure 4.8: Close-up of Seurat's *Parade de Cirque* transferred to rendered sunset; Pointillist/Impressionist style

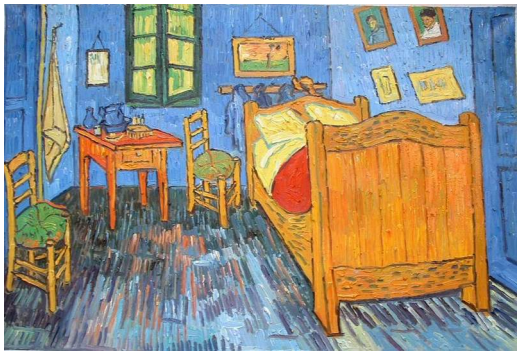
Additionally, we created a modified version of this algorithm which displays images as though they were sketched with pencils. The algorithm is identical until the point where every pixel within a certain distance would be filled in with the location's colour. Instead, every pixel from *(x-n, y-n)* to *(x+n, y+n)* is filled in with the colour, creating a sketchy effect.

Figure 4.9: Image with pencil sketch style

(a) Original image [3]

(b) Aungier Street, Dublin [4]

(c) Colour transferred from (b) to (a)



(d) Van Gogh, *The Bedroom* [5]

(e) Colour transferred from (d) to (a)

Figure 4.2: Colour transfer onto a greyscale image

Figure 4.3: Original: Cottage, Connemara, Ireland [6]

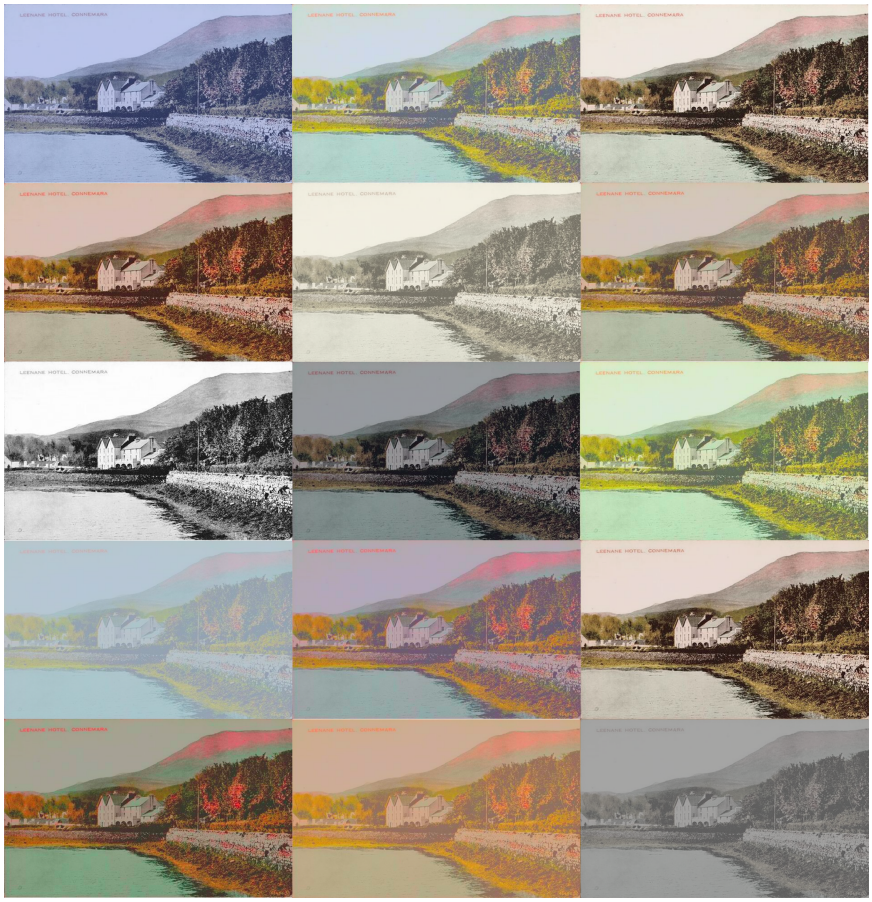


Figure 4.4: Fifteen variations created with colour transfer

Chapter 5

Conclusion

5.1 Results

5.1.1 Is such a framework possible?

We set out to investigate the state of the art in the field of style transfer, in order to determine whether a cohesive framework could be created which brought together many varied facets of the transfer process. In our research, we determined that it is certainly possible to include some aspects of style transfer in a framework, but a fully-automated transfer process does not seem to be possible at this time.

We did have success implementing colour transfer; the algorithm selected was successfully applied to a variety of still images and was also used to convert an entire video. While the process is not perfect, in general, the overall colour tones from a style image were successfully transferred to the original image. Manual adjustment of brightness and contrast allowed for further alterations of the new image.

The automatic detection and application of brushstrokes, arguably the second most important aspect of style transfer, was not possible, due to being an unsolved problem at the time of writing. Our idea, for style identification against a database, could work, though it proved very difficult, due to two factors. First, though it is quite accurate, the chi-squared distance algorithm does not return the optimal result 100 percent of the time. Even when the result returned is most similar, as judged by human standards, there may not be enough information in the returned result. Second, the chi-squared test is based solely on the colour palette used in an image, and not on brushstrokes or

content. Even an artist who is known to consistently use the same hues of paint across their body of work will vary somewhat, or produce some works which are completely different. Other artists may use very similar palettes, leading to misidentification. Even when two images are created by the same artist, using the exact same colour palette, finding accurately coloured reproductions of the images is extremely difficult, due to skew in different pieces of equipment and computer monitors. A much larger, more accurate database could aid the identification of images, but it is unlikely to solve the problem completely.

Implementing style transfer functions individually and combining these with colour transfer process has been shown provide some interesting and pleasing results. However, adding a large number of functions would be a very time-consuming process, and would essentially defeat the purpose of creating a unified application. Based on this information, we have determined that it is not possible to create a fully comprehensive style transfer framework at this time, but a similar, less cohesive, more manual framework, could be built.

5.2 Future Work

5.2.1 Automated Brushstrokes

In the future, as more research is done in the areas of automatic brush detection and application, we hope that an algorithm could be included in the framework. The addition of automated brushstrokes in a framework of this type would be greatly useful. It would allow for a far wider range of styles to be transferred to an image. Even if the time existed to incorporate a large number of style filters, many styles would not be included, and of course it would still be simpler and less time-consuming to implement a single algorithm.

5.2.2 Optimisation

The existing framework was coded naïvely; performance was not taken into account when coding. Using the YC_BC_R colour transfer reduces computational time somewhat, as well as some of the more subtle coding choices, such as the use of lambdas to eliminate

a few large for-loops. However, the application is single-threaded, and the current style transfer application does not act in real-time. At the most, only a few images can be processed every second, depending on the size of the selected images. This is acceptable when creating a video from still frames, but it would not be appropriate for transforming real-time video, or a real-time animated 3D scene. Parallelisation could be implemented in the future to speed up the transfer process.

5.2.3 Improved User Experience

The Windows Form used for the graphical user interface does not run in a separate UI thread. As a result the application can become unresponsive when processing a large amount of data, and it is not possible to press another button on the form at this time. The addition of threads to delegate various tasks to would speed up the program and improve the user experience greatly. In addition, the form was designed practically, containing no superfluous decorations or visually interesting items. Improving the overall look of the form could make its use more enjoyable.

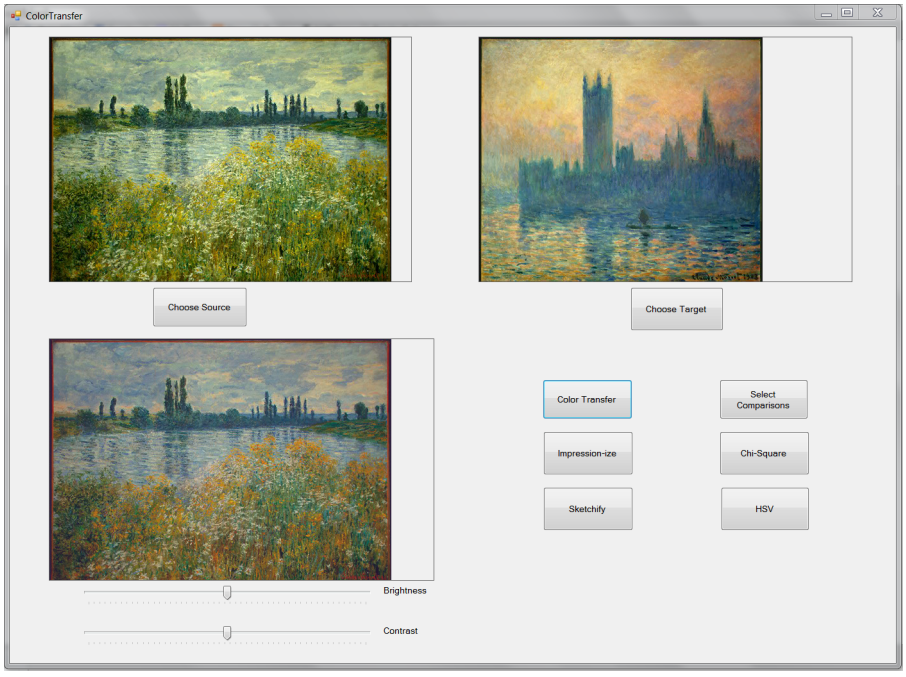


Figure 5.1: Screenshot of the existing application

The application developed demonstrates the advantages of a cohesive framework for style transfer. In addition, its modular nature allows for future extension, allowing for other algorithms and styles to be seamlessly added.

Bibliography

[1] Igor E. Berezhnoy, Eric O. Postma, and H. Jaap van den Herik. Automatic extraction of brushstroke orientation from paintings: Poet: prevailing orientation extraction technique. *Mach. Vision Appl.*, 20(1):1–9, October 2008.

[2] Margarita Bratkova, Solomon Boulos, and Peter Shirley. orgb: a practical opponent color space for computer graphics. *IEEE Comput. Graph. Appl.*, 29(1):42–55, January 2009.

[3] Stefan Bruckner and Meister Eduard Gröller. Style transfer functions for illustrative volume rendering, September 2007. Eurographics 2007 3rd Best Paper Award.

[4] W. Carlson. A critical history of computer graphics and animation.

[5] Charles Csuri and James Shaffer. Art, computers and mathematics. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part II*, AFIPS '68 (Fall, part II), pages 1293–1298, New York, NY, USA, 1968. ACM.

[6] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Studying aesthetics in photographic images using a computational approach. In *Proceedings of the 9th European conference on Computer Vision - Volume Part III*, ECCV'06, pages 288–301, Berlin, Heidelberg, 2006. Springer-Verlag.

[7] Oxford English Dictionary. Style, 2012.

[8] Christian Eisenacher, Sylvain Lefebvre, and Marc Stamminger. Texture synthesis from photographs. In *Proceedings of the Eurographics conference*, 2008.

[9] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 327–340, New York, NY, USA, 2001. ACM.

[10] C.R. Johnson, E. Hendriks, I.J. Berezhnoy, E. Brevdo, S.M. Hughes, I. Daubechies, Jia Li, E. Postma, and J.Z. Wang. Image processing for artist identification. *Signal Processing Magazine, IEEE*, 25(4):37 –48, july 2008.

[11] Michael Kalloniatis and Charles Luu. Color perception, 2007.

[12] Hochang Lee, Sanghyun Seo, Seungtaek Ryoo, and Kyunghyun Yoon. Directional texture transfer. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '10, pages 43–48, New York, NY, USA, 2010. ACM.

[13] Martin Lettner, Paul Kammerer, and Robert Sablatnig. Texture analysis of painted strokes. *28th Workshop of the Austrian Association for Pattern Recognition*, 2004.

[14] Guangxin Li. Image fusion based on color transfer technique. 2011.

[15] Jingwan Lu, Pedro V. Sander, and Adam Finkelstein. Interactive painterly stylization of images, videos and 3d animations. In *Proceedings of I3D 2010*, feb 2010.

[16] The J. Paul Getty Museum. Understanding formal analysis, November 2008.

[17] Laszlo Neumann and Attila Neumann. Color style transfer techniques using hue, lightness and saturation histogram matching, 5 2005.

[18] Chuong H. Nguyen, Tobias Ritschel, Karol Myszkowski, Elmar Eisemann, and Hans-Peter Seidel. 3d material style transfer. *Comp. Graph. Forum*, 31(2pt2):431–438, May 2012.

[19] Takashi Nishiyama, Hayato Ichino, Tsuyoshi Nakamura, Masayoshi Kanoh, and Koji Yamada. Impressionist painterly style transfer based on texture localization.

In *SIGGRAPH Asia 2011 Posters*, SA '11, pages 46:1–46:1, New York, NY, USA, 2011. ACM.

[20] Ofir Pele and Michael Werman. The quadratic-chi histogram distance family. In *Proceedings of the 11th European conference on Computer vision: Part II*, ECCV'10, pages 749–762, Berlin, Heidelberg, 2010. Springer-Verlag.

[21] François Pitié, Anil C. Kokaram, and Rozenn Dahyot. Automated colour grading using colour distribution transfer. *Comput. Vis. Image Underst.*, 107(1-2):123–137, July 2007.

[22] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. Manga colorization. *ACM Trans. Graph.*, 25(3):1214–1220, July 2006.

[23] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Comput. Graph. Appl.*, 21(5):34–41, September 2001.

[24] Paul L. Rosin and Yu-Kun Lai. Towards artistic minimal rendering. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '10, pages 119–127, New York, NY, USA, 2010. ACM.

[25] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth movers distance as a metric for image retrieval. *International Journal of Computer Vision*, 40, 2000.

[26] Anderson Sean E. Barzel Ronen Salisbury, Michael and David H. Salesin. Interactive pen-and-ink illustration. In *ACM SIGGRAPH*.

[27] Amelia C. Sparavigna and Roberto Marazzato. Non-photorealistic image processing: an impressionist rendering. *CoRR*, abs/0911.4874, 2009.

[28] Dong Wang, Weijia Jia, and Guiqing Li. Color transfer using scattered point interpolation. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '11, pages 163–170, New York, NY, USA, 2011. ACM.

[29] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. Transferring color to greyscale images. *ACM Trans. Graph.*, 21(3):277–280, July 2002.

[30] Jinliang Wu, Xiaoyong Shen, and Ligang Liu. Interactive two-scale color-to-gray. *The Visual Computer*, 28:723–731, 2012.

[31] Xuezhong Xiao and Lizhuang Ma. Color transfer in correlated color space. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, VRCIA '06, pages 305–309, New York, NY, USA, 2006. ACM.

[32] Xuezhong Xiao and Lizhuang Ma. Gradient-preserving color transfer. *Computer Graphics Forum*, 28(7):1879–1886, 2009.

[33] Xuexiang Xie, Feng Tian, and Hock Soon Seah. Feature guided texture synthesis (fgts) for artistic style transfer. In *Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts*, DIMEA '07, pages 44–49, New York, NY, USA, 2007. ACM.

[34] Hanli Zhao, Xiaogang Jin, Jianbing Shen, and Feifei Wei. Real-time photo style transfer. In *Computer-Aided Design and Computer Graphics, 2009. CAD/Graphics '09. 11th IEEE International Conference on*, pages 140–145, aug. 2009.

Image References

[1] Monet, Claude. Waterloo Bridge. 1903. Monetalia.
$< http://www.monetalia.com/paintings/monet-waterloo-bridge-london.aspx >$
[2] Monet, Claude. Banks of the Seine, Vétheuil. 1880. National Gallery of Art, Washington, DC.
$< http://www.nga.gov/collection/gallery/gg85/gg85-46652.html >$
[3] Playboy Magazine. Lena. Nov. 1972.
$< http://www.ece.rice.edu/\ wakin/images/lena512.bmp >$
[4] Postcard of Aungier Street, Dublin. Old Irish Postcards: Dublin. Belfast Telegraph.
$< http://www.belfasttelegraph.co.uk/news/nostalgia/old-irish-postcards-dublin-15115648.html >$
[5] Van Gogh, Vincent. Bedroom at Arles. 1888. Van Gogh Gallery
$< http://www.vangoghgallery.com/catalog/Painting/715/Vincent_s-Bedroom-in-Arles.html >$
[6] Postcard of cottage, Connemara, Ireland. Old Irish Postcards: County Galway. Belfast Telegraph.
$< http://www.belfasttelegraph.co.uk/news/nostalgia/old-irish-postcards-county-galway-15121310.html >$
[7] Driving in Dublin in 1976. User:willmcq. Video. YouTube.
$< http://www.youtube.com/watch?v=pWUferyKCww >$
[8] Van Gogh, Vincent. Wheat Fields With Reaper at Sunrise. 1889. Van Gogh Gallery
$< http://www.vangoghgallery.com/catalog/Painting/757/Wheat-Fields-with-Reaper-at-Sunrise >$
[9] Van Gogh, Vincent. The Siesta. 1889. Art of Europe.
$< http://www.artofeurope.com/van_gogh/van20.htm >$
[10] Postcard of The Silver Strand, Barna, Co. Galway, Ireland. Old Irish Postcards:

County Galway. Belfast Telegraph.
$< http : //www.belfasttelegraph.co.uk/news/nostalgia/old - irish - postcards - county - galway - 15121310.html >$
[11] Van Gogh, Vincent. Starry Night Over the Rhone. 1888. Van Gogh Gallery.
$< http : //www.vangoghgallery.com/painting/starryindex.html >$
[12] Grafton Street, Dublin. Old Images of Dublin in Black and White. Irish History Links.
$< http : //www.irishhistorylinks.net/pages/Old_Dublin_Black_White.html >$
[13] Premoze, Simon. Rendering of sunset. 2000. Rendering of Natural Waters.
$< http : //www.cs.utah.edu/ michael/water/ocean1.jpg >$
[14] Seurat, Georges. Parade de Cirque (close-up). Renoir Fine Art.
$< http : //www.renoirinc.com/biography/artists/seurat/detail.htm >$

All images used without permission.