Development and implementation of a Natural Media Filter for the post-processing of animated sequences

Luke Goddard[1]

March 2008

[1]email: goddardl@ntlworld.com

Abstract

Figure 1: Illustrations by Dave McKean, taken from the graphics novel "Black Orchid". (©Gaiman & McKean [1989])

This paper documents the research and development undertaken in the creation of a digital image processing technique that uses a custom image filter to simulate the appearance of traditional media. In particular, it focuses on the process of generating a filtered image in a chosen medium from raw footage only. The intention of the project was to provide a useful tool for creating a desired aesthetic without the need for any other form of input source other than the original footage.

The concept of this project was inspired by the illustrative works of Dave McKean and in particular his expressive and lavish use of multiple media within the graphic novel series "Black Orchid" (Figure 1).

Replicating a style similar in complexity and aesthetic to the works of McKean within a single image filter was originally the projects goal, however, translating the static imagery to a dynamic animation derived from footage only, whilst retaining the visual cohesion and continuity of motion proved to be a substantial challenge. As a result, research into several avenues of image analysis (most notably, computer vision [4.1]) was prompted and the project digressed away from the production of a singular image filter tool towards a workflow solution, combining existing and custom written software applications to achieve the desired result.

Despite the format of the media filter moving form a singular piece of software to a workflow, the intention of the project remained the same: to develop a means of replicating the visual style and characteristics of natural media.

Throughout this report, the methods developed, and research undertaken in response to the challenges posed during the creation of the media filter are described and critically analysed. The report is structured to reflect the projects development, describing the research and implementation of a basic media filter [3], followed by further development into integrating computer vision [4.1] and concluding with the production of an animated short showcasing the developed method and tools.

Contents

Chapter 1

Introduction

> Image processing is formally defined as "subjecting numerical representations of objects to a series of operations in order to obtain a desired result" (Rao [2004]).

Image processing is essentially the filtering of an input stimulus produced by a physical device such as video camera, microscope or radar to produce an output information signal. Its purpose is to extract useful information from the scene being imaged as the raw data is often unsuitable and therefore needs to be processed. This is called Image enhancement and is most commonly associated with digital image processing which uses a two dimensional signal as input; a digital image composed of rows and columns of pixels (Silver [2000]).

Digital image processing was developed in the 1960s and proliferated as computers and hardware became cheaper, faster and more readily available. It is now widely used in many applications including facial recognition, digital photography and medical imagery (Wikipedia [2007]). As a corner stone of the film industry, it is also frequently used during the editing and correction of footage and makes such techniques as digital compositing and other forms of post processing possible.

A single or combined series of digital image processes that are used to produce an image of a particular aesthetic are called Filters and are widely used to treat raw film footage during post-production to correct and modify it.

1.1 Observtions and Objectives

When approaching the problem of producing a process to mimic the effect of a natural media, several existing options were also considered. These

solutions are all split into one of two groups; those that render the final image from a description of a scene such as a modelling package and, post processing solutions that change an existing image to produce the desired result. Digital image filters belong to the second category. Unlike other methods of image construction such as Non-Photo realistic rendering (NPR) that take a mathematical description of a scene to 'render' an image, digital image filters derive the resulting outcome only from pre-existing frames which are supplied as the input (Lum & Ma [2002]). In this respect, an image filter is more versatile as it can be applied to both filmed footage and that rendered from a 3D modelling application unlike a rendering solution that cannot. However, there are limits to the range of problems that an image filter can be used to solve; as a post-production process it is inherently restricted to only a single frame or sequence as input. Therefore, image filters are best suited to the correction or manipulation of an existing source by applying a procedure or set of rules to change its appearance rather than to produce a new, original image.

With this in mind, a series of objectives were derived to structure the development of a natural media image filter:

1. Analyse the technique and style of Dave McKeans illustrations, breaking them down into a series of rules to be applied by the media filter [1].

2. Identify and produce a foundation of basic but critical image filters for image enhancement [3].

3. Build upon the base image operations (2) to construct filters to achieve the rules outlined in step 1.

1.2 An Overview of Image Filters

The most basic way of representing an image digitally inside a computer is in the form of a bitmap. A bitmap stores a representation of an image by decomposing its colours into the three primaries red, green and blue. Each colour component is then usually stored within a byte in the computers memory, allowing it to be quantified by a number between 0 and 255 (which is the maximum range of a byte). This technique is called RGB encoding and each set of colour components that constitute a point on the image is called a pixel. Each pixel is then held within a 2D array that makes up the area of the bitmap (Patin [1999-2008]).

1.2.1 Image Operations

Digital image filters work by performing a function using the input pixels to produce a new, output pixel. In mathematics, this process of producing an output signal from an "impulse response" (function) and an input signal is called convolution (Appendix [1]) and is the fundamental concept in image processing. By iteratively performing the function (also known as an image operation) across a bitmaps rows and columns of pixels an output image is produced.

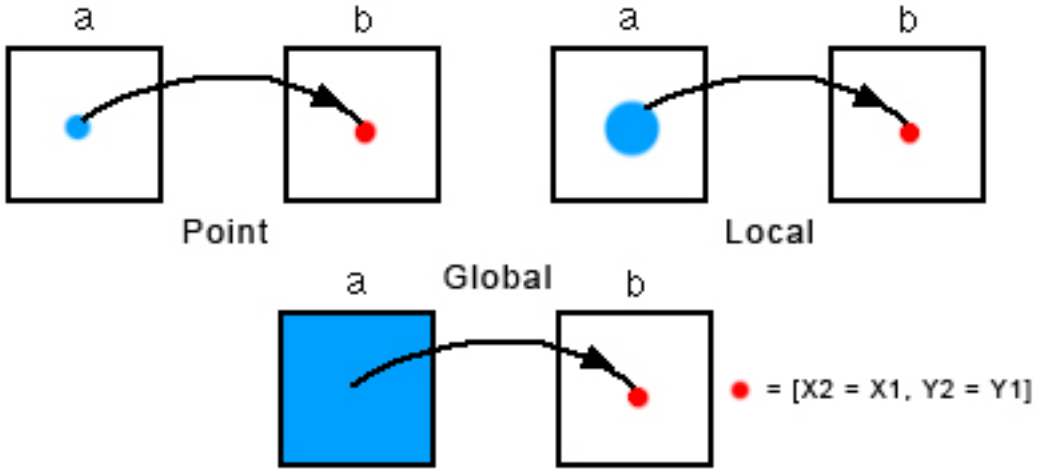


Figure 1.1: An overview of point, local and global operations. (©Young [2003])

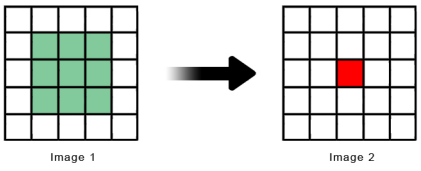


Figure 1.2: Illustrating the neighbourhood concept, the target pixel within image 2 (shown in red), calculates its colour by using the values of the pixels local to its coordinate in image 1 (show in green).

Image operations are characterized by the input data that they require to transform an input pixel a[x,y] to an output pixel b[x,y]. These can usually be classified into three groups: point, local and global (Figure 1.1). The output value of a specific coordinate as the result of a point operation is dependent

only on the input value at the same coordinate. Whereas, the output value of a specific coordinate using a local or global operation is dependant on the input values in the neighbourhood or entire image of the same coordinate (Figure 1.2). Simply put, each operation can be categorized by the number and locality of the pixels that it has to sample in order to produce an output pixel (Young [2003]).

Point image operations are mostly used in the more simple filters that deal with image adjustment such as desaturation or colour replacement whereas; local and global operations are used when more information is required about the pixels locality or neighbourhood on either a local or global scale. An example of such a filter would be a Gaussian blur which requires the colour values of the pixels within the source image that are in proximity to the output pixels coordinate.

1.2.2 Matrix Convolution

Most image filters that use local operations to sample input pixel values use a mathematical domain called 2D convolution or Matrix Convolution to derive a result (GIMP [2002 - 2008]). A simple explanation of matrix convolution is the treatment of one matrix by another, which is called a 'kernel'. The first matrix holds the two-dimensional image that we wish to treat and the kernel matrix contains a series of values that are used to manipulate it. Other names for a convolution matrix are PSF (pulse-spread function) and filter response.

To calculate the result of the filter, each pixel coordinate in the output image is looped over and, the input pixels within the neighbourhood are sampled. The sampled pixels are then each multiplied by the corresponding element of the kernel and summed.

The use of Matrix Convolution is best understood using an example (Figure 1.3). The image matrix is shown on the left with each of the pixels values marked, the pixel to filter shown in red, and the sample neighbourhood (area of the kernel matrix), outlined with a green box. The middle matrix is the kernel with its values shown and the right matrix is the convolution matrix (result).

The output pixel on the convolution matrix is calculated by looping through each of the values in the input image within the neighbourhood area by their corresponding values in the kernel matrix. The results of these calculations are then summed to create the value of the new pixel (GIMP [2002 - 2008]).

Convolution matrix operations are extremely easy to implement and allow a wide range of filters to be easily constructed by just changing the values of

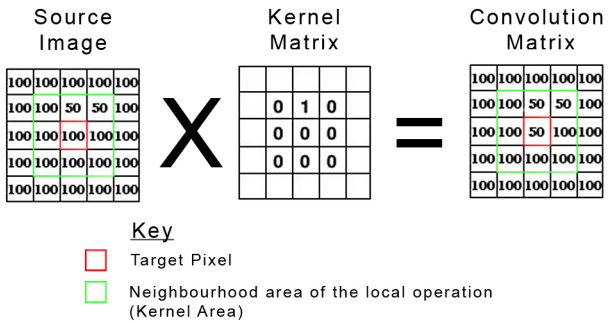


Figure 1.3: An illustration of Matrix Convolution. (©GIMP [2002 - 2008])

the kernel matrix (Appendix [2]).

Chapter 2

Characteristics of Natural Media

Theoretically, natural media such as paint, pastel and pencil are good candidates for image filtering. Disregarding style, technique and abstract representation, the application of any media can be broken down into a series of rules for constructing a likeness to a scene. Leaving only the characteristics of the medium to be simulated or replicated. As a result, by analysing the difference between an image rendered using a natural media such as paint and a photograph of the same scene, a series of rules can be established and a program (image filter) constructed to procedurally translate one aesthetic to the other.

However, the limitations of such a process are obvious and the omission of a stylistic influence would be a large oversight to any attempt at replicating the appearance of a handcrafted image. With this in mind, it was decided that rather than approaching the problem from a broad perspective, a singular artists style would be chosen (Dave McKean) with the intention of reducing his painting style down to its base elements, integrating and implementing them with the medias characteristics, as the result is fabricated.

2.1 An Analysis of McKean's Illustrative Style

Before the design and implementation of a media filter could begin, it was necessary to analyse the principles of how pastel, pencil and paint are used, and in particular, how they are applied within a mixed media illustration created by Dave McKean.

The main defining characteristics of McKeans works is the way he uses expressive, lavish paint strokes and splatters to layer up colours and texture

within an image, augmenting areas with pastels that are worked over the top, often to imply motion. Details are built up out of the loose application of paint resulting in a rich image that has been carefully constructed but, almost appears accidental as brush strokes are also rarely seen. With the exception of backgrounds, the majority of his illustrations are outlined with the subtle use of pastels and paint, giving better coherency to the details and contrasting well with his broad painting style.

McKean's use of colour is another defining feature within his work and is used to great effect when creating atmosphere and mood. Pencil and monochrome imagery is often used in contrast with bright or rich colours as a device to highlight important focal points or to express a character or scene's mood and atmosphere. His use of pencil is always complimented with black and white paint, defining and highlighting outlines and form whilst juxtaposing the fine, sharp features of the pencil. This helps to maintain a consistency of style with his more painterly techniques.

2.2 Derived rules for constructing an image in the style of Dave McKean

Through the analysis of Dave McKean's illustrative style, the key components of his technique were identified and used to define a set of rules that are required of the media filter in order to create a convincing image. These rules were used as goals for the functionality of the media filter:

1. An accurate application of paint splatters and strokes to define form, areas of contrast and volume (Figure 2.1, Panels 1, 2 and 3).

2. Feathered outlines created along the boundaries of painterly forms where the paint has sprayed, dripped and ran off edges. This effect is intentional but is also a property of the paint due to the method in which it is applied. This is most notably observed within Figure 2.1, Panel 1, along the wing of the parrot.

3. A combined use of monochromatic pencil and paint to create contrasting textures within the same image (Figure 2.1, Panel 2).

4. Well-defined edges and details using pastel and chalk, worked over the top of other media (Figure 2.1, Panel 3).

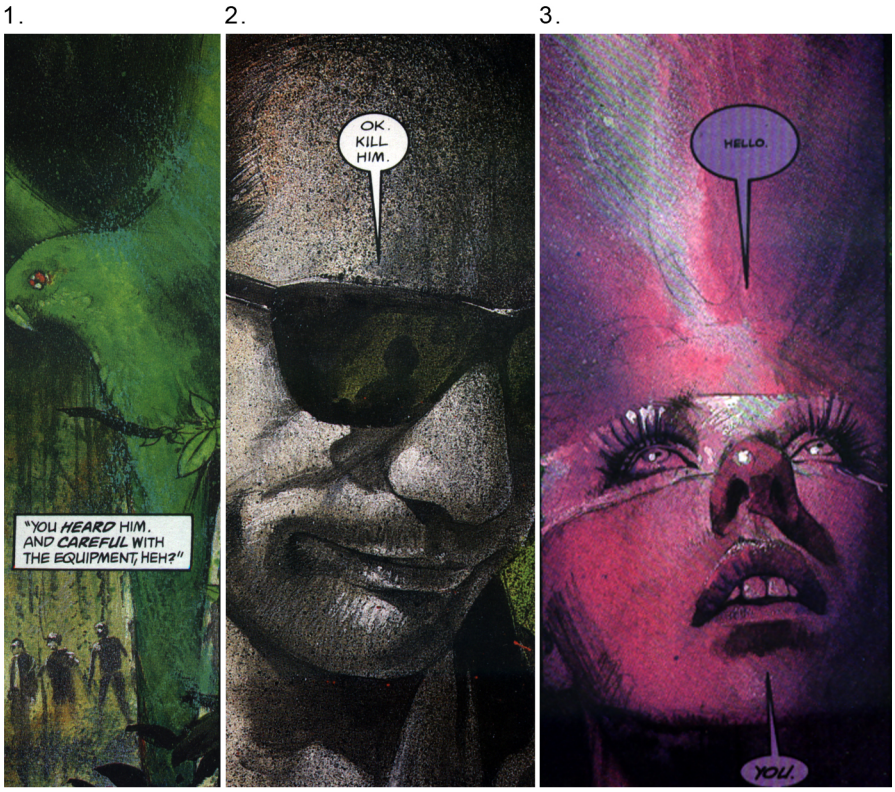


Figure 2.1: A Cross section of Dave McKean's different illustrative techniques within "Black Orchid". (©Gaiman & McKean [1989])

Chapter 3

Implementation

Due to the limited range of input provided to an image filter (a single frame), when approaching an initial method for implementing the characteristics of pencil, pastel and paint, more ways of extracting information from the input had to be considered. It was hoped that by expanding the number of inputs available to the media filter, more complex procedures could be written and a better likeness to the mediums achieved. Used partly as an exercise in the production of image filters, several standard image operations were first constructed and used to enhance a sequence of test images. The goal was to generate a range of results that could also be used later to broaden the input stimulus to the media filter.

This process developed from little more than trial and error as simple image operations were written in the C programming language, the results observed, changed and developed until useful information was extracted from the source image. This period of experimentation was critical to the projects development, expanding the restricted range of input data that was available to work with, ultimately speeding progress later.

3.1 An overview of the base image operations

The first processes written were some of the most common and simple but served as a solid starting point and baseline upon which more complex filters were later written. Mostly using point and global operations, these filters were desaturate, level correction and edge extraction. Despite being reasonably primitive, these operations were found to be integral to larger, more complex filters and provided a foundation upon which they could be developed. The most basic but fundamental image operations are illustrated within the following section as they each play a pivotal part in the construc-

tion of the final image filter. Incidentally, they also represent the main three groups of image operations; Point, Local and Global.

3.1.1 Desaturate

Figure 3.1: An example of the Desaturate Filter applied to a colour image.

The desaturate filter is the most simple filter that was implemented within the project and is an example of a basic point operation. However, despite its simplicity, it is used as a corner stone within other filters and is therefore extremely useful. Its purpose is to simply reduce an image to its grey scale components and is illustrated in Figure 3.1

As the colour of a pixel is RGB encoded, the values of each of the components therefore represent the luminosity of the three colour channels. As a result, by averaging the sum of the components, a single scalar value is produced that is the overall grey scale value of the pixel.

The algorithm used to implement this filter is shown below:

1. Iterate over the source image and get each pixel.

2. Sum the pixels components and divide by 3.

3. Put the result into the red, green and blue channels of the target images pixel.

3.1.2 Find Edges

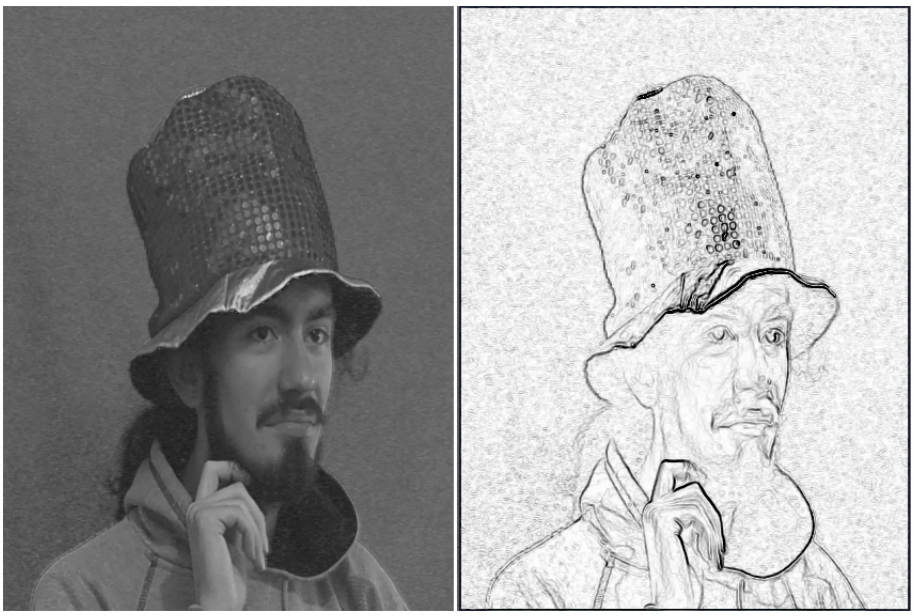


Figure 3.2: An example of the Find Edges Filter applied to a colour image.

$$\begin{pmatrix} -1/8 & -1/8 & -1/8 \\ -1/8 & 1 & -1/8 \\ -1/8 & -1/8 & -1/8 \end{pmatrix}$$

Figure 3.3: The Kernel Matrix used to create the Find Edges filter. (©Patin [1999-2008])

The find edges filter is also very common and is an example of a local operation. Its purpose is to emphasize the differences in contrast levels within an image, producing a bitmap that highlights edges as lines. This particular operation was exceptionally useful when combined with contrast adjustment, leading to a method for creating pastel and pencil outlines. This filter was

very easy to implement once the matrix convolution framework had been written. An example of the result of the find edges filter along can be seen in Figure 3.2, and the kernel used to create it shown in Figure 3.3.

3.1.3 Brightness and Contrast Adjustment

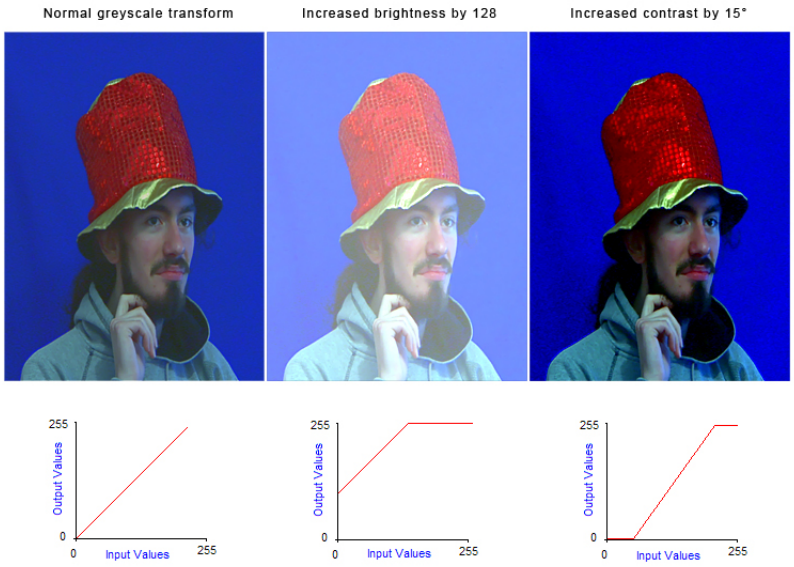


Figure 3.4: An example of the effects different grey transforms have on an image using an array of gamma curves. (©Patin [1999-2008])

Brightness and Contrast adjustment is achieved by mapping each of the grey scale values of the colour components against a "Gamma Curve" or look up table. This curve or table expresses the new grey values for any of the old. As each of the colour components are encoded as an integer between 0 and 255, they can be used to retrieve an output from the table or curve containing the new grey value (Illustrated in Figure 3.4). This filter is an example of a point operation but becomes a global operation if the levels of contrast within the image are analysed beforehand to influence the filter. For example, when creating an 'auto levels' filter that adjusts the images contrast levels to be evenly distributed.

The algorithm for the brightness and contrast adjustment filter is shown below:

1. Use the new gamma curve to create a look up table of new values.
2. Sample each of the pixels within the input image and pass them into the lookup table.
3. Write the new grey value to the output image.

3.2 Developing upon the base image operations to simulate natural media

Once a baseline of image operations had been constructed, it became possible to start extending their functionality to mimic the effects of pastel, pencil and paint to acomplish the goals previously outlined (page 9). With a now broadened range of input that could be fed to the filter, an approach to the construction of the natural media filter was chosen. It was decided that rather than augmenting a source sequence to mimic a combination of natural media, a better result could be achieved by reconstructing the image out of the product of a series of small filters, each of which adding a different characteristic of a particular media to the result. This lead to a more modular work flow, allowing filters producing separate elements and characteristics of the natural media to be developed independently and the results composited together afterwards, improving the users control over the images construction.

All of these more complex filter's were implemented using a Photoshop script. Combined with an intuitive GUI (graphics user interface), all of the filter's are integrated within the same environment, speeding the ease of use and development time. Details on the Photoshop plug-in can be found within the appendix (C.1).

3.2.1 Creating the filters for Boarders and Outlines

Pastel Outline

The first filter developed was the 'Pastel Outline' which reduces an input image down to its edges and shades them with a pastel-like texture (Rule 4, page 9). The filter was constructed using a combination of the find edges filter to extract the areas of contrast within the image, contrast correction

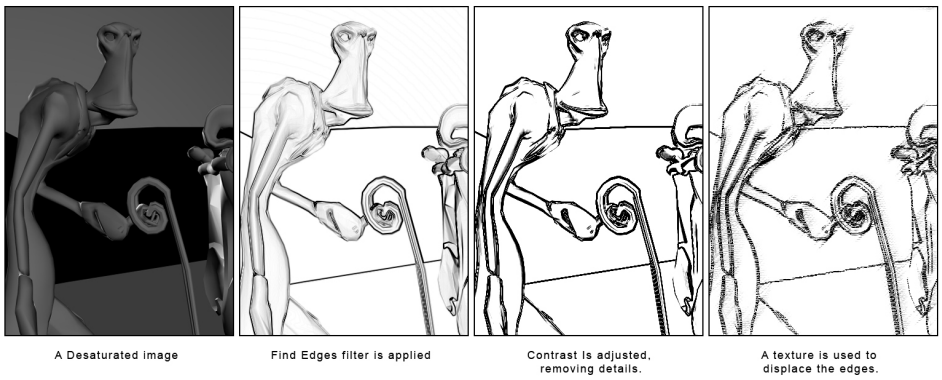


Figure 3.5:

to reduce the smaller lines to leave only the most prominent borders and a grey scale texture displacement to shade the outlines with a pastel texture.

The results of the filter are quite effective, giving the look of a rough pastel, pencil or chalk outline drawn around an image (Figure 3.5). An image processed with this filter is compared to the pastel and pencil outlines of McKean's illustrations in Figure 3.6. It is clear from the comparison that the filter successfully produces an output very close in aesthetic style to that of the pastel outline's in McKean's illustrations.

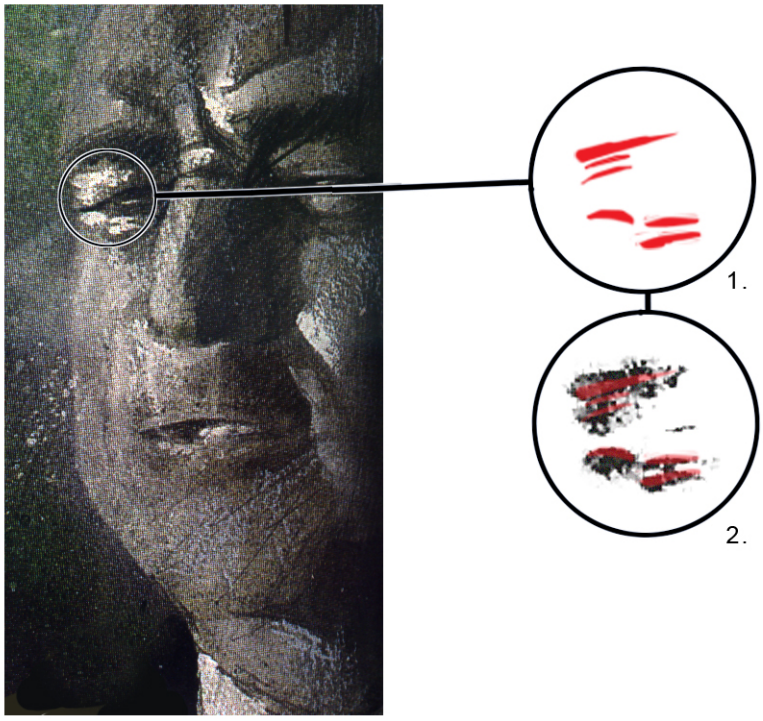
Painterly Outline

Replicating an effect to resemble the expressive, painterly edges found in McKean's illustrations (Rule 2, page 9) was a much less trivial problem than producing pastel outlines. Due to the manor in which paint is applied across a surface, splatters, drips and bleeds needed to be simulated along the extracted edges found within the source image. Rather than producing an image filter to outline the objects with paint, the goal was to create a series of masks with painterly boarders that could later be used to layer textures upon each other in a similar way to the traditional application of paint and having the effect of reducing the crisp and precise appearance of the edges to a rougher, more natural look (Figure 3.7).

To produce this effect, the filter was split into two sections; the generation of a series of black and white 'Contrast masks' representing the differing levels of brightness and a process to expand or erode the edges of these masks into a supplied texture to produce a feathered, painted aesthetic. The result of

A comparison between the result of the filtered edges and those found within an illustration by Dave McKean.

Figure 3.6:



1. The basic lines that define the form of the eye can clearly be seen.

2. However, due to the characteristics of the paint and the style through which it was applied, the edges become more obscured.

Figure 3.7:

the combined process is a series of 'media masks' of the varying levels of brightness in the image which can be used as a compositing aid to layer textures of paint on top of each other, mirroring the traditional technique of applying darker shades of paint and shadows over the lighter highlights that are applied first.

The process for generating these media masks was substantially more complex than any previously constructed filter and as a result, it was implemented within a Photoshop script to speed its development and to provide a comprehensive GUI for interaction with it [C.1]. The contrast mask filter was developed upon the brightness contrast adjustment operation and used it to extract the various levels of brightness, producing a series of user parametrised black and white images. Fed to the next stage as input, the media mask filter proceeds to overlay a sequence of user provided textures (and later the product of a texture generating filter) to the masks. By selecting the black areas of the brightness masks, and expanding them into the overlaid texture by a specified tolerance, the selections edges inherit parts of the texture and the output media mask created.

This process is illustrated within Figure 3.8.

Texture Generation

The generation of the pencil/pastel and paint textures was one of the last obstacles that needed to be overcome (rule 1, page 9) before a basic media filtered sequence could be created by compositing the textures together using the media masks and overlaying the objects outlines over the top.

Initially, the texture generation filters for both the pencil/pastel and paint were created using the same principle; the output image is constructed out of a large number of particles, textured using various scans of paint and pencil and driven by the input sequence. Together, these particles use the source frame (after image enhancement) to reconstruct it with a new textural aesthetic. The only difference between the application of this process to create the effect of pencil or pastel and paint, is that the pastel or pencil's texture can be created in one step, without the need to layer different textures upon each other using the media masks. The use of the media masks is only recommended when replicating the look of layered paint.

Each of the particles within the filter uses a local operation to obtain information from the pixels in proximity to their coordinates, dictating their texture selection. As a result, a desaturated input sequence was used to drive the particle's texture to resemble different variations in shading density or style, producing an image similar to that of a pencil, pastel or paint rendering. Using this method of driving the particles textures with the source frames

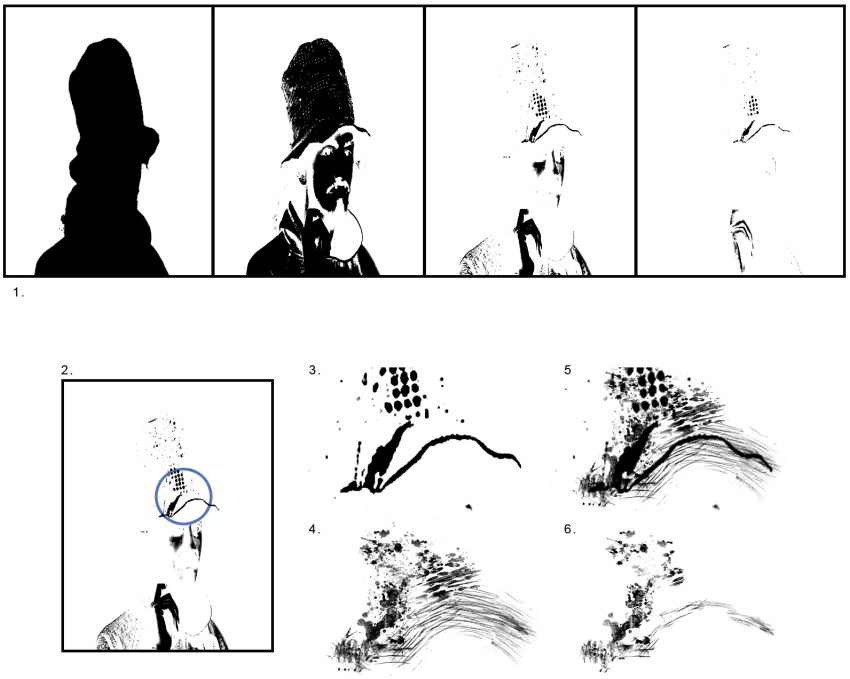
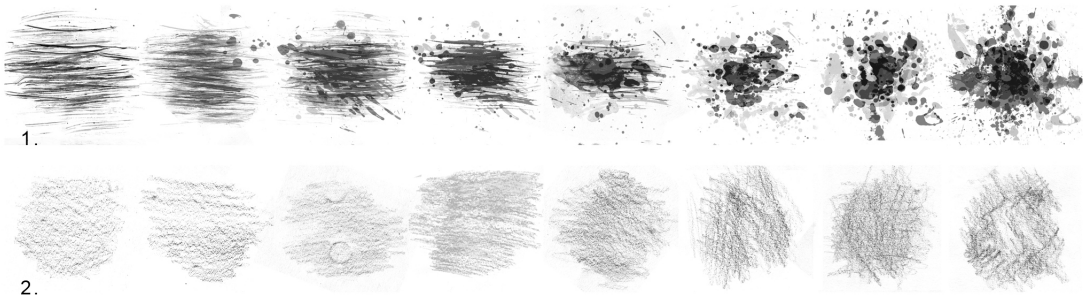


Figure 3.8:



The particle textures used for pencil (2) and paint (1), ranging in brightness and density from left to right. Textures were selected depending on the brightness of the pixels the particle was placed upon.

Figure 3.9:

brightness values, form and tone is easily and accurately portrayed within the output image which, can be seen in Figure 3.9, achieving one of the original goals (Rule 1, page 9). This approach to texture generation gives very pleasing results and is very flexible as any additional information to be used as stimuli for the particle texture selection can be extracted from the input sequence.

However, there are limitations to the likeness that this texturing algorithm can produce compared to an image traditionally rendered by an artist. The reason for this is because the information being extracted from the source frame only considers the forms within it in two diemensions. As volume within the image is not considered, implementng traditional shading techniques that use the angle of the stroke to suggest volume, are impossible to implement. As a result, the textures produced by this filter can appear 'flat and lifeless' when depth and volume needs to be expressed.

Regardless of this drawback, the texturing functionality of the work flow was the last in the series operations required to complete the media filter which, once added, was capble of producing very pleasing and natural media-like images.

Chapter 4

Media in Motion

With the development and implementation of the basic media filter already defined within the previous sections, this chapter details builds on the progress already made and furthers it with the digression of research into computer vision. Until this point in the projects development, all of the filters written and techniques considered were all based within a two dimensional spatial domain the bitmap. Identified after developing the first texture filter, it soon also became evident that to achieve a higher order of quality, more information needed to be extracted from the input animation.

The two main problems that had been identified were:

1. Inherently 'flat' textures due to no information on the depth or volume of the source images content being extracted.

2. Disjointed and distracting textures that destroy the visual cues within an animation's composition because they do not respond to the motion within the sequence.

It is clear from the definition of these problems that without considering a temporal domain within an animation, the filter cannot react to the motion of the image over time, resulting in inherently 'flat' and disjointed textures when applied to an animation. This problem origionaly arose by only considering an animated sequence as a series of separate still images that could be filtered independently of each other. However, within traditional animation this is not the case and like the suggestion of volume, stylistic and shading techniques are used to imply motion. A solution was needed that could provide information about the sequence over time allowing the filter to use this data as a stimulus and compensate. One referenced animation in particular was very inspirational when developing a better understanding of how natural media can be used to imply motion and can be found within the appendix (B.1, Inspirational and Referenced Animations).

4.1 Computer Vision

Computer vision is a relatively new area of research that is concerned with the theory of producing systems that obtain information from images. Although in its infancy, computer vision is a fast growing area of research with applications in 3D scene reconstruction, medical imagery and object detection. Almost all computer vision research was sparked by a seminal paper on optical flow published in 1981 and written by Horn and Schunck (University of Otago [2003]).

4.1.1 Optical Flow Analysis

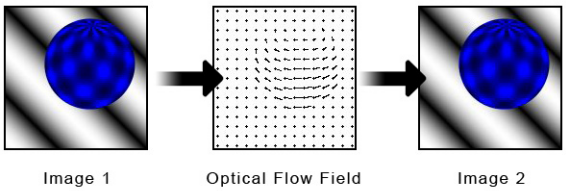


Figure 4.1: The sphere within image 1 is rotating from left to right, moving to the position shown in image 2 and generating the optical flow field shown in the centre. (©University of Otago [2003])

Most computer vision applications are concerned with the extraction of an optical flow field from an image stream. Optical flow fields are a visual representation of the motion within an image, expressed as a grid of vectors. In a similar way, the stereo disparity of two consecutive frames can also be derived using a variation of the same algorithm used for optical flow and a depth field generated. These vectors are often encoded as pixels within a bitmap, using each of the three colour component channels to represent a different axis of motion. An example of an optical flow velocity field is illustrated in Figure 3.9.

Optical flow algorithms work by comparing two frames and trying to map every pixel from one frame to the next. There are three main types of optical flow algorithms: Gradient based, Frequency domain based correlation and Block matching (Seymour [2006]). Each of these methods have their own advantages and limitations but generally use a similar approach to optical flow analysis; by assuming that all objects have a single motion, relative

to each other or the camera, they can each be separated to form layers of depth and used to produced an optic flow field. A four-stage process is used to extract the layers from the image. The motion within the image is first segmented into regions within each frame and then compared across the sequence to identify corresponding regions. These regions are then linked with the starting or 'reference' frame and are used to generate layers with intensity and alpha maps (Ogale et al. [2005]).

Figure 4.2: The extraction of an image into layers through the analysis of its Optical Flow Field. (©Ogale et al. [2005])

The result of such a process can be seen in Figure 4.2 which illustrates the analysis of an input frame and its separation into layers through the identification and extraction of its objects.

The optical flow process has many limitations that are often related to more complex or ambiguous shapes within the image. These include translucent objects, shifting lighting conditions or objects with multiple vectors of movement. An example of a problematic sequence is a spinning shiny sphere as its motion would be impossible to track the pixel values that describe its surfave between frames would not change at all. However, despite the limitations of the technique, even at the lowest quality or poorest result, the data retrieved from a processed image can provide both optical flow and depth fields holding substantial information about the content and motion within an image, perfect for image processing applications.

4.2 Integrating Optical Flow and Stereo Disparity into the Texture Generator

Based upon source code developed by Ogale et al. [2005], a program was written that analyses the optical flow and stereo disparity of an input sequence. The program uses several computer vision algorithms to produce a series of grey scale bitmaps that express the motion vectors of the animation in a visual form. Each of the vectors is encoded within the bitmap as a grey

scale pixel representing the motion or depth of the point between one frame and the next.

Figure 4.3: An example of a still from the animation "The Organ Grinder" and the ('X' and 'Y') optical flow fields derived from it.

Entering the output of the optical flow algorithm into the texture-generating filter as a sequence of frames representing the 'x' and 'y' components of the motion vectors, the bitmaps are automatically levelled (contrast adjusted) and blurred to reduced noise and imperfections. They are then used in conjunction with the existing particle texture system to construct the new media textures. Allowing each of the particles to use the vector data to select appropriate textures, rotations, opacity and scales depending on the motion, depth and brightness of the pixels within the neighbourhood it is positioned upon, much more complex textures can be generated which respond to the motion or depth and the source animation.

Given this extra dimension of input stimuli the number of possible particle textures was increased from a single row of eight images to a grid of 8x8 (Figure 4.4). The particle images held within this grid are specifically designed and ordered to reflect all of the possible combinations of motion vector over time. Organised from left to right by the magnitude of motion and from bottom to top by the degree of fluctuation of the vectors direction, a particles image can easily be selected from its history of motion. For example, a particle that has consistently had a constant direction would select a texture from the bottom row, relative to its magnitude as all of the images are very unidirectional. However, if its average vector over the past χ frames has been turbulent and its direction changes regularly, an image would be chosen from the top row according to its magnitude as all of the images are much more random and less linear.

In addition to intelligent texture selection based on a points optical flow value, the same information can be used to drive the particles rotation and scale, aiming, growing and shrinking it along its vector of motion. Using the optical flow field to drive textured particles, an image is constructed that reacts to the motion of the animation (Figure 4.5).

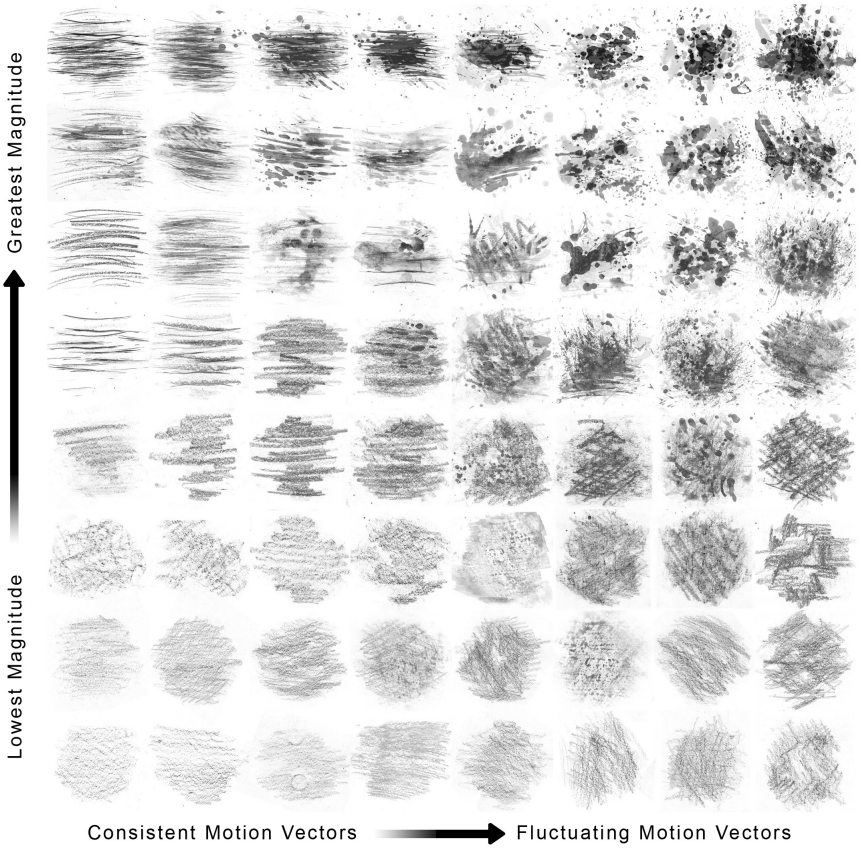


Figure 4.4: The sprite grid used to texture the particles that compose the filtered image.

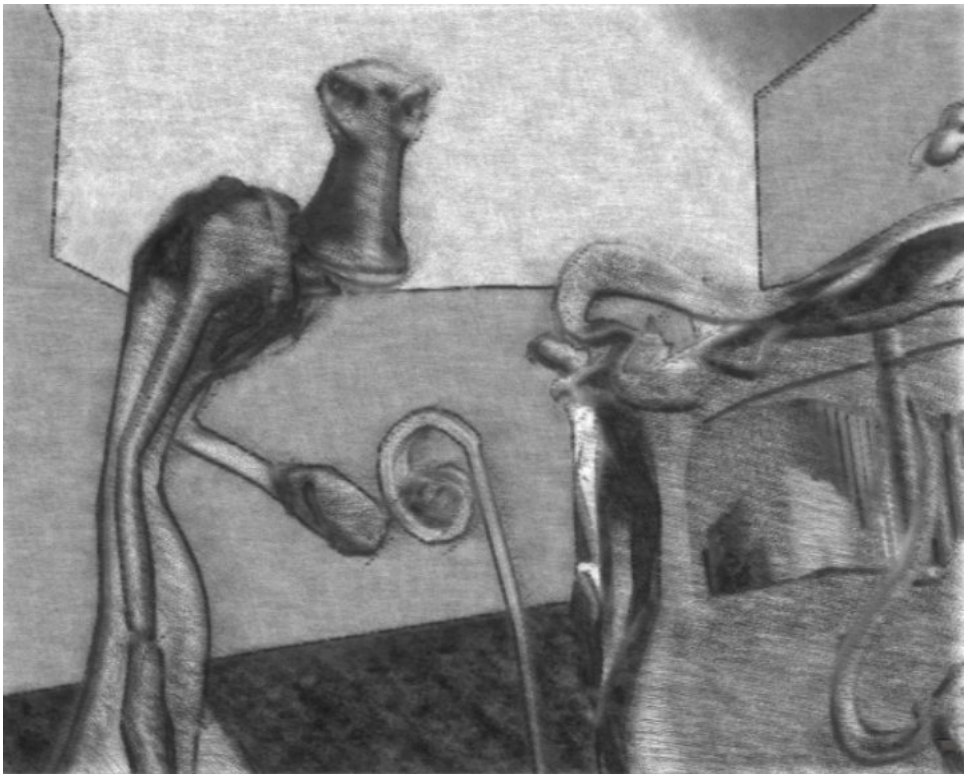


Figure 4.5: An image rendered from "The Organ Grinder" and filtered using the Optical Flow driven texture generator.

Shortcomings of the static distribution of particles

Although having a greatly improved result when compared to the early texture generating filter, this optical flow based design also has its drawbacks. Due to the limitations of the quality of the optical fields extracted from the source sequence coupled with the static nature of the particles distributed across the output frame, the transitions on the particles between textures, scales or rotations can appear jarring when noise in the motion vectors is introduced. As a result, the painterly textures that require larger particles to ensure that the splats, drips and strokes are visible, are more susceptible to these inconsistencies due to each particle on screen being more noticeable. This is compared to the pastel and pencil textures which are hardly affected as the particles are much smaller and therefore less individually noticeable. The solution to this problem would be to remove the particles direct level of interaction with the optical flow field by using it to drive a particle or fluid system instead. With more time, such a solution could be implemented but is currently left for further research.

Other methods of Optical Flow Integration

Due to the sheer quantity of information available as the result of computer vision analysis on the sequence, allot of time was spent experimenting with different techniques to combine its use with the textured particles. The first method for applying this newly available data was to physically drive the motion of the particles across the image in the hope that a more fluid texture could be produced. However, regulating the global distribution of the particles was a severe problem and it was common for them to form groups of a high density in parts of the image, leaving other areas exposed and bare. Without individually tweaking the motion parameters of each scene, it became clear that the due to the problem's scope, a solution would not be found easily without considering a more complex particle behavioural system. This was judged as impractical and costly to performance and eventually, the particles were just statically distributed over the image. The result of using this technique was still visually impressive, generating textures driven by not only the frames brightness value's but also its motion, maintaining the visual continuity and coherency of the animation.

Chapter 5

Conclusions

5.1 The Finished Pipeline

The final method developed for the filtering of an animation was the use of several small filters that together, produce a versatile work flow. The work flow integrates a comprehensive Photoshop script that encompasses three of the most complex and critical filters within an integrated environment and easy to use GUI (C.1). The second half of the filtering software was written in C++ and incorporates the source code developed by Ogale & Aloimonos [2005*b*] to create a sequence of images that describe the optical flow field of the input animation. Although created as a seperate application to allow motion vector fields to be extracted separately from the texturing process, the derived sequence is then passed into the texture generation filter which produces images resembling paint, chalk and pencil. Laslty, the products of all of these filters can be combined using compositing software to create the final, media filtered sequence. This work flow is illustrated in figure 5.1.

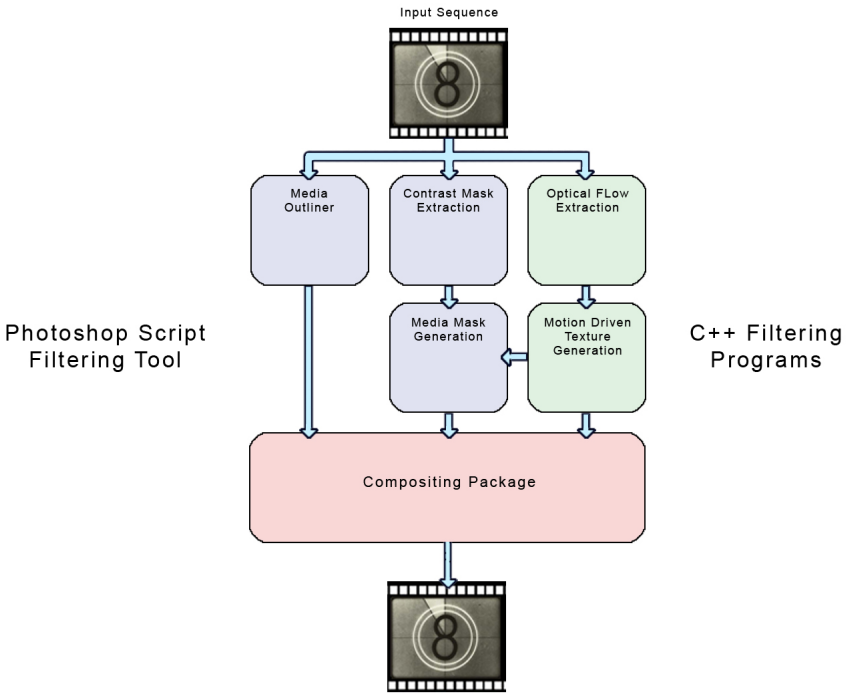


Figure 5.1: An illustration of the final work flow. The filtering components incorporated within the Photoshop filter can be seen in blue, the custom C++ filters in green and an external proprietary compositing application such as Apple's 'Shake' (©Apple Inc.) in red.

5.2 Conclusion and Evaluation

5.2.1 An evaluation of the projects success

The initial goal of this project was to research and develop a single application using image processing techniques to filter a sequence of images to mimic the aesthetic of natural media, and in particular, the artwork of Dave McKean. During the projects development, a range of image operations were written using C++, which due to their complexities and time constraints were eventually integrated into a Photoshop script that sped development and increased the filters usability by providing a simple and intuitive interface. However, in doing so the filters structure became fractured and the orientation of the project turned towards the production of a work flow of small tools rather than a singular filtering application.

Several problems were encountered during production of the media filter, such as the extraction of volume and motion using only a simple sequence of images as input. However I feel that the problems were met, researched and solved furthering the quality of the final filtering results. The digression of research that was the result of addressing these problems also helped to develop my understanding of the representation of motion within animations that use natural media. This led to several improvements of the filters design and implementation.

Without an existing media filter to compare the results of the project to, defining its successfulness is subjective. It was my original intention to develop a singular media filter application for the post-processing of a sequence of images without the need for additional scene data and, in this respect I have both failed and succeeded. Due to unforeseen texture-related complexities and time limitations in the development process, the production of a singular application became impossible. Instead, a series of tools created were designed to compliment existing packages and produce a work flow that was more flexible, time consuming to use but ultimately producing the desired outcome. As a result, I personally consider the project a success as the product of the filter looks reasonably convincing, and with the knowledge and experience gained whilst developing it, I feel that with time, it would be possible to consolidate the work flow into a single, effective tool.

Despite producing visually interesting images, in my opinion, the one large area that was a failure due to time constraints and the diversion of research, was the usability of the texture generator and optical flow filters. Neither of these programs have sufficient user interfaces as they were produced in parallel to the research into optical flow and as a result, were written ‘on-the-fly’ as development tools and were not designed with ease of use in

mind. Therefore, in its current state, I do not consider the work flow (with the exception of the Photoshop scripts) complete, and judge the best qualities of the projects outcome to be the depth of its research and the proposed method of image processing integration with computer vision.

5.2.2 Completion of Objectives

One of the main goals of the project was to drive the result of the media filter with only a singular sequence of images and to use image-processing techniques to extract any additional information required creating the desired aesthetic. This objective was the most successfully completed and yet still underdeveloped. Through research into areas of computer vision, more information was derived from the source frames than was effectively used, leaving many possible avenues of research and experimentation left to be exploited. Disregarding the effective results gained from research into this area, it is clear that I barely scratched the surface of what is possible through the integration of computer vision algorithms. However, further exploration of the applications of this innovative technology in more depth extends far beyond the scope of this project but, given time, is definitely a subject I would like to pursue in the future.

In defining the requirements of the media filter, several rules were defined (1, page 9). Each of these rules were addressed and successfully completed as either an integrated component within the Photoshop script or as a separate C++ application. Proof of this can be seen within figures 5.2, 5.3 and the animated short which all demonstrates the functionality of the media filter.

5.2.3 Future developments and research

If the project was to be extended or repeated and as I would have the benefit of hindsight, more attention could be paid to replicating the techniques concerned with chalk, pencil and paint renderings. Although not poorly achieved in its current state, I would consider the result of the filter as a good representation of natural media rather than a simulation. With the knowledge that has been gained about image filters and optical analysis, further designs could be better structured, utilizing computer vision algorithms as a fundamental baseline rather than as a response to the problem of the flat appearance of other texturing methods. This would in turn allow more of the work flow to be consolidated into a single, structured filter reducing the time to process a sequence and its complexity. Furthermore, once an all-encompassing environment for the filter had been developed, the integration

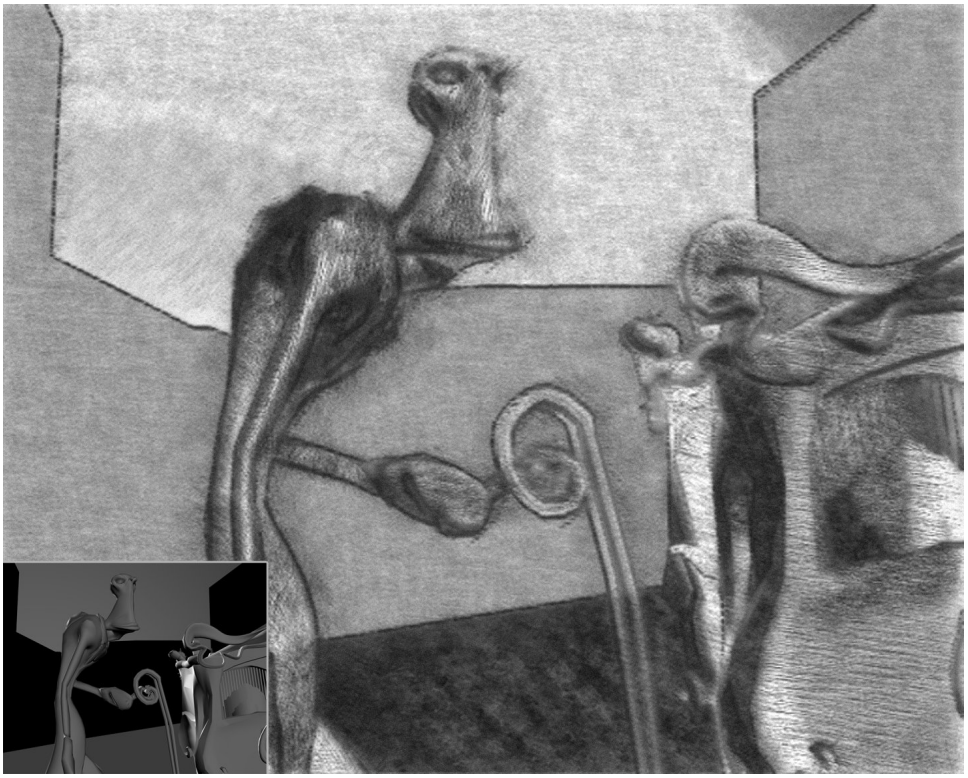


Figure 5.2: This image is a frame taken from an animation filtered to look like pencil.

Figure 5.3: This image is a frame taken from a short film clip and is filtered to look like chalk.

of a user interface would be trivial which, is currently the biggest drawback of the current design.

An interesting and final digression upon the research undertaken into computer vision and its applications within post processing would be its integration as stimuli for more advanced particle systems (proposed in 4.2). Through the combination of optical flow data with either fluid or particle simulation, much more fluid, smooth and complex textural motion could be achieved, solving problems associated with the static distribution of texture particles across and image (4.2). For example, the slightly jarring motion of the texture particles within the current implementation due to artefacts in the motion vector data could be solved by driving the particles using a fluid simulation, moving in response to the motion vectors of the optical flow field. This would create textures that would flow like water across the image, behaving much more like wet paint. The parameters of the fluid simulation could even be configured to simulate the viscosity of the medium being used.

Bibliography

ECE (1998), *An efficient algorithm for Gaussian blur using finite-state machines*, SPIE Conf. on Machine Vision Systems for Inspection and Metrology VI.

Gaiman, N. & McKean, D. (1989), *Black Orchid*, DC Comics.

GIMP, T. D. T. (2002 - 2008), *Gimp GNU Image Manipulation Program Documentation*, Gimp GNU Image Manipulation Program. http://docs.gimp.org/en/plug-in-convmatrix.html.

Lum, E. B. & Ma, K.-L. (2002), *Non-Photorealistic Rendering using Watercolor Inspired Textures and Illumination*, Department of Computer Science University of California.

Ogale, A. S. & Aloimonos, Y. (2005*a*), 'Shape and the stereo correspondence problem', *International Journal of Computer Vision* **3**, 147 – 162. http://www.cs.umd.edu/users/ogale/papers/ogaleIJCV05shape.pdf.

Ogale, A. S. & Aloimonos, Y. (2005*b*), 'Stereo and optical flow implementations in c++'. http://www.cs.umd.edu/users/ogale/download/code.html.

Ogale, A. S., Fermller, C. & Aloimonos, Y. (2005), 'Motion segmentation using occlusions', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**, 988 – 992. http://www.cs.umd.edu/users/ogale/papers/ogalePAMI05motion.pdf.

Patin, F. (1999-2008), 'An introduction to digital image processing'. http://www.gamedev.net/reference/programming/features/imageproc/.

Rao, K. (2004), Overview of image processing, Technical report, National Remote Sensing Agency, National Remote Sensing Agency, Hyderabad, India.

Seymour, M. (2006), 'Art of optical flow'. http://www.fxguide.com/article333.html.

Silver, B. (2000), An introduction to digital image processing, Technical report, Cognex Corporation, One Vision Drive, Natick, MA.

University of Otago, D. (2003), 'Optical flow algorithm evaluation'. http://www.cs.otago.ac.nz/gpxpriv/vision_downloads.html#optflow.

Wikipedia (2007), 'Digital image processing'. http://en.wikipedia.org/wiki/Digital_image_processing.

Young, T. (2003), 'Image processing fundamentals'. http://www.qi.tnw.tudelft.nl/Courses/FIP/noframes/fip-Contents.html.

Appendix A

Formulas

A.1 Convolution

A.1.1 Discrete and Continuous Convolution

Convolution using Continuous time

$$y(t) = h(t) \times x(t) = \int_{-\infty}^{\infty} h(\alpha)\ \mathbf{g}x(t-\alpha)\ \partial\alpha = \int_{-\infty}^{\infty} h(t-\alpha)\ \mathbf{g}x(\alpha)\ \partial\alpha$$

Convolution using Discrete time

$$y[k] = h[k] \times x[k] = \sum_{i=-\infty}^{\infty} h[i]\,\mathbf{g}x[k-i] = \sum_{i=-\infty}^{\infty} h[k-i]\,\mathbf{g}x[i]$$

Figure A.1: The formulas for convolution using continuous and discrete time where x(t) is the entering signal and h(t) is the response of the filter to a delta impulse (Patin [1999-2008]).

A.1.2 Two Dimensional (Matrix) Convolution

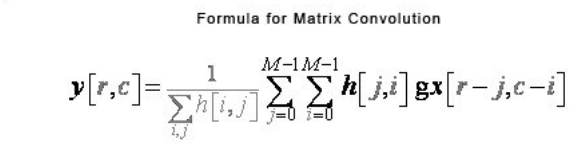


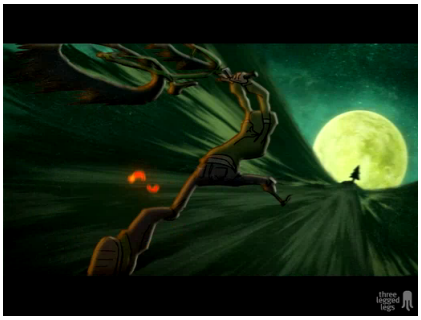
Figure A.2: The output image y of an entering bitmap x, through a filter system of bitmap impulse response h (width and height M) (Patin [1999-2008]).

Appendix B

Inspirational Animations

B.1 Four Legged Legs

When considering animations that use natural media whilst demonstrating extreme movement, the most inspirational that were found are a series of adverts produced by 'Three Legged Legs'. Each of the three sequences portrays a character participating in an extreme sport, such as BMX bike riding, skateboarding and stunt biking and all have extreme motion and movement.



The media within these animations is used to great effect, implying motion and blur by using 'scribbles' and elongation and exaggeration of limbs and movement (Figure B.1).

Figure B.1: These still frames from an animated series by 'Three Legged Legs' uses distortion of form and 'scribbles' to imply motion.

Appendix C

Photoshop Filter

C.1 User Guide

To speed development of the natural media filter and to aid usability by combining a comprehensive graphical user interface (GUI), some of the more complex image filters were implemented as a Photoshop script. This script presents the user with a broad range of controls for filtering a target sequence with an array of filters. The GUI of this Photoshop script is explained here, along with an explanation of the filters that it includes.

C.1.1 The Target Sequence Selection Menu

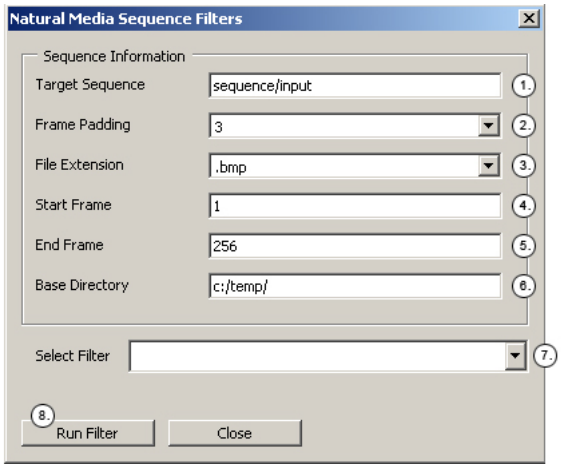


Figure C.1: The Target Sequence Selection Menu is the first GUI that the user is presented with before a filter is selected.

The fields that it displays all relate to the target sequences information:

1. Target Sequence: The name and location of the target sequence, relative to the base directory is entered here.

2. Frame Padding: This drop down box allows the user to select the number of leading ‘0s that are appended to the input sequences name.

3. File Extension: This option allows the user to specify the input sequences file format.

4. Start Frame: The starting frame number.

5. End Frame: The frame number of the last frame within the target sequence.

6. Base Directory: The field allows the user to specify the root directory that all of the other directory structures are relative to.

7. Select Filter: This drop down box presents the user with all of the available filters that can be used. Each of the filters uses the target sequence information presented within this GUI and, upon selection of a filter, the widow extends downwards-revealing options relevant to the chosen operation.

8. Run Filter: This button will execute the select filter if all of the required information is valid. If not, an error will be returned. Once a filter operation is complete, the user is notified.

C.1.2 Available Filters

Media Outliner

The ‘Media Outliner filter is an implementation of the ‘Pastel Outline filter (3.2.1, Page 15) but also expands upon its functionality, providing a greater range of media that can be used as the outlines texture.

1. Brightness Adjustment: This field allows the user to adjust the brightness of the input sequence before the find edges operation is applied, providing control over the removal of unwanted lines or darkening of others.

2. Contrast Adjustment: This Field should be used in combination with the brightness adjustment to remove or emphasize lines.

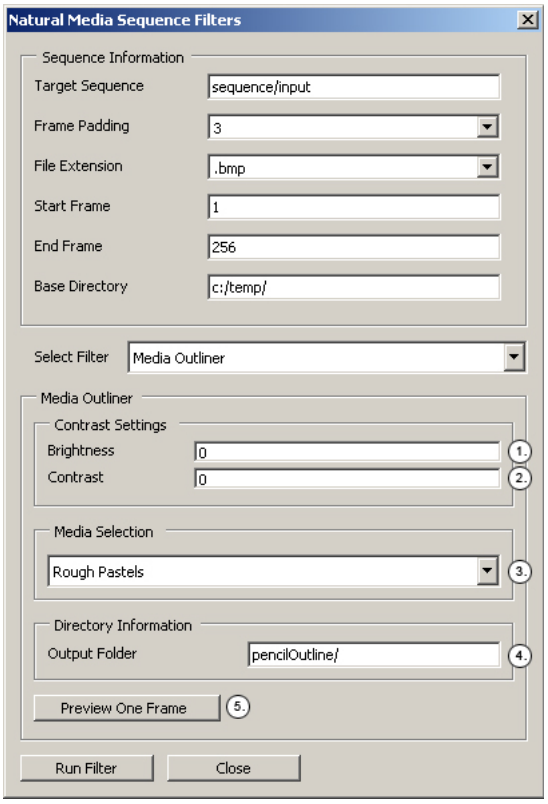


Figure C.2: The Media Outline Filter's GUI.

3. Media Selection: This drop down menu allows the user to select between the type of effect that is applied to the outline. The options are ‘Rough Pastel, ‘Pencil and ‘Chalk.

4. Output Folder: This field allows the user to specify the output directory that the resulting filtered sequence will be written to. This folder is relative to the base directory.

5. Preview One Frame: This button allows the user to preview the results of a single frame rather than processing an entire sequence.

Contrast Mask Generator

The contrast mask filter takes a sequence of images and extracts a specified number of contrast levels from it, saving the result as separate files. This filter is particularly useful as its output is used as input to several other filters.

1. Number of Masks: This option allows the user to specify how many levels of contrast are extracted from the input sequence. Usually between three and six masks are adequate.

2. Mask Mode: This drop down box presents the user with a choice between ‘Combined Levels and ‘Separate Levels. These options refer to whether the output contrast masks are multiplied upon each other, concatenating all previous masks together or, to only save the difference between the levels of contrast.

3. Edge Threshold: The edge threshold refers to the sharpness of the mask produced. When set at 9, the edges are very sharp, creating a black and white image. However, the more the number decreases, the more tapered the edges become, introducing grayscale values.

4. Contrast Curve: The contrast curve defines how the levels of contrast are extracted. The choice is between linear, exponential and inverse exponential. A linear extraction will sample the contrast levels evenly whereas an exponential curve will sample more mid-range levels of contrast.

5. Brightness Clamp Min: This slider allows the user to shift the minimum brightness value of the contrast levels extracted. This has the effect of stopping the lower brightness values from being sampled.

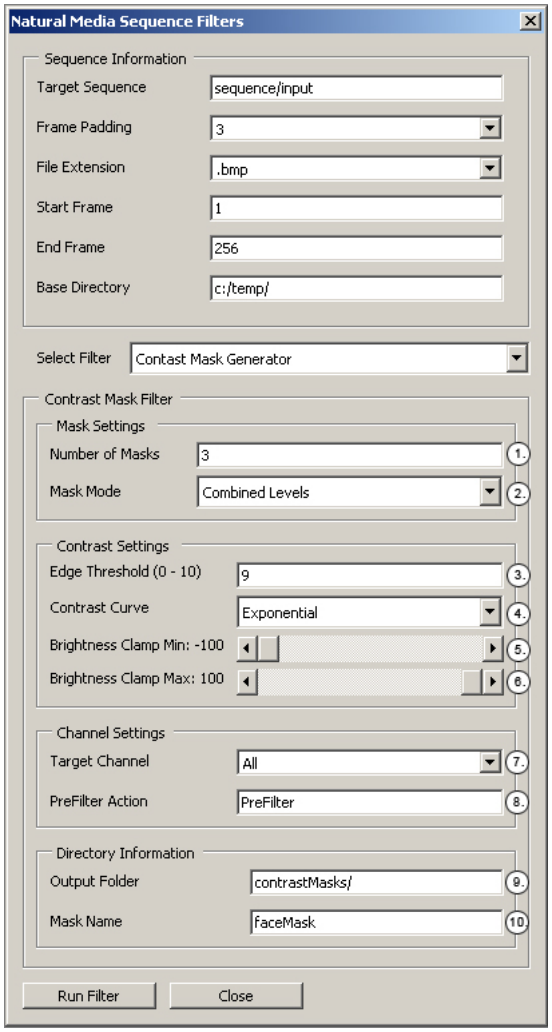


Figure C.3: The Contrast Mask Filter's GUI.

6. Brightness Clamp Max: This slider allows the user to shift the maximum brightness value of the contrast levels extracted. Combined with the minimum clamp, a range of contrast levels can be specified.

7. Target Channel: This drop down box allows the user to select the channel that the contrast mask filter will extract the levels of contrast from. 8)PreFilter Action: Any Photoshop action can be specified here and ran before the contrast levels are extracted. This is particularly useful when particular areas might require masking or adjusting before the contrast mask extraction begins.

8. Output Folder: This field allows the user to specify the output directory that the resulting filtered sequence will be written to. This folder is relative to the base directory.

Media Mask Generator

The media mask generator uses the contrast masks created by the contrast mask filter and combines them with an animated texture sequence, producing a mask that extends the contrast masks by a specified tolerance and style, into the texture. This filter is extremely useful for extending splatters of paint over the boundaries of an object, creating painterly edges.

1. Media Mask Mode: This drop down box allows the user to select whether the resulting mask is the product of both the cropped texture and the input mask or just the texture.

2. Artifact Range: Any textural artifacts extending out of this range from the input mask edge are removed.

3. Artifact Drop-off: This value specifies the drop off rate of the artifact range. This allows the artifacts that fall upon the boundary to be faded off.

4. Crop Edge Smooth: This value specifies how much the artifact range (crop edge) is smoothed to reduce sharp edges.

5. Artifact Drop-off Mode: Presented with a choice between and ‘Fade and ‘Dissolve, the user can select the style with which artifacts found along the artifact range will be removed. ‘Dissolve breaks them down into small pixel fragments and ‘Fade tapers off their opacity.

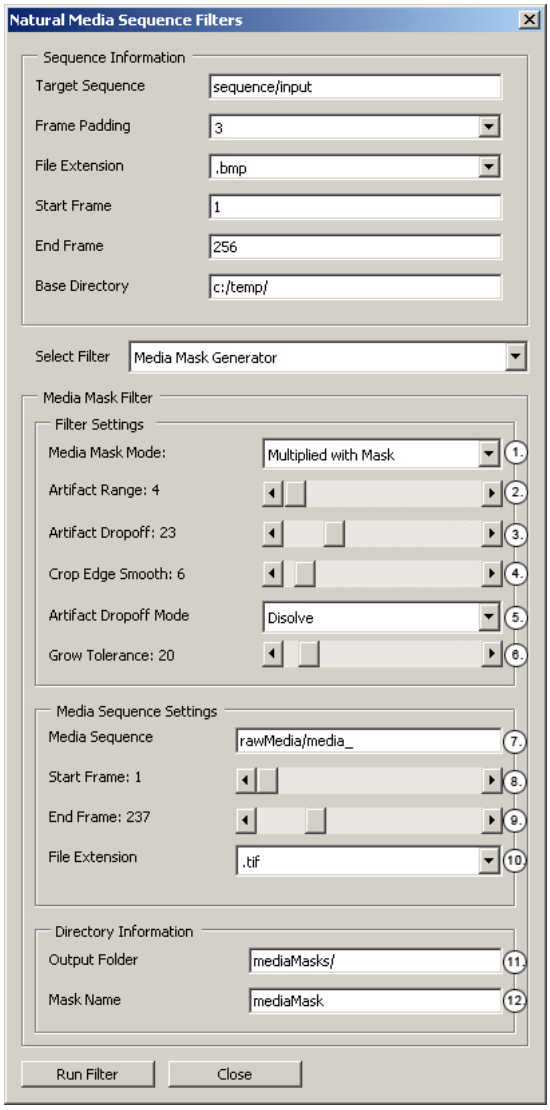


Figure C.4: The Media Mask Filter's GUI.

6. Grow Tolerance: This parameter is critical to the filter and specifies to what tolerance the edges of the contract mask will be extended into the texture. Large values will extend the edges further.

7. Media Sequence: This field allows the user to specify the name and directory of an animated texture sequence relative to the base directory that will be combined with the contrast mask sequence to produce the result.

8. Start Frame: Specifies the start frame of the texture sequence.

9. End Frame: Specifies the end frame of the texture sequence.

10. File Extension: This drop down box is used to select the file format of the input texture sequence.

11. Output Folder: This field allows the user to specify the output directory that the resulting filtered sequence will be written to. This folder is relative to the base directory.

12. Mask Name: The name of the resulting output masks.