INTERACTIVE GRADIENT DOMAIN TEXTURE BLENDING

by

Michael Welsman-Dinelle

Submitted in partial fulfillment of the
requirements for the degree of
Bachelor of Computer Science, Honours Co-op

at

Dalhousie University
Halifax, Nova Scotia
March 2007

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Computer Science for acceptance a thesis entitled "INTERACTIVE GRADIENT DOMAIN TEXTURE BLENDING" by Michael Welsman-Dinelle in partial fulfillment of the requirements for the degree of Bachelor of Computer Science, Honours Co-op.

Dated: March 30, 2007

Supervisors: ______________________________
Dr. Dirk Arnold

____
Dr. Stephen Brooks

Table of Contents

List of Figures

Abstract

Textures are two-dimensional images used to increase the realism of computer-rendered scenes. This thesis describes a texture blending tool capable of generating new textures by synthesizing existing textures. The blending tool performs two functions. First, it allows users to apply a blending algorithm that takes as input two tileable textures and a mask defining which regions to take from which texture. Poisson interpolation is then used to generate a tileable output texture that varies gradually between regions drawn from either of the two inputs. The tool also assists in user selection of textures and masks by using image characteristics to draw many similar options from a large texture database, a difficult problem when textures are not generated procedurally and similarity must be inferred from the images themselves. By varying the amount of randomness included in the selection of candidate texture and mask sets, the blending tool allows users to both explore the effect of blending different textures and to fine tune selections in order to produce a desired final result.

Chapter 1

Introduction

This thesis describes the creation of a system that can be used to generate new textures from a library of existing images and masks. Two existing real textures are selected from a database and combined to form a new texture. An algorithm performs a blending of two textures in such a way that the different regions of the two combined images are integrated seamlessly and realistically. Sample output from the blending algorithm is shown in Figure 1.1. The system also guides user selection of textures and masks for blending by providing candidate texture sets taken from the database. An example texture set can be seen in Figure 1.2. Candidate texture sets are created by calculating the similarity between textures in the database and a selected texture. Both the blending algorithm and texture classification and selection methods used to generate texture sets are described in detail in later sections.

Chapter 2, describes what textures are and why it makes sense to attempt to generate new ones. Chapter 3 gives a broad overview of related research, focusing on the problem of generating new textures from existing textures and the problem of selecting similar images from a large image database. Many different approaches exist to solve both problems. Texture blending using both morphing and wavelets are contrasted with the approach used for this thesis. Three broad ways of classifying images are described: some methods use frequency domain information, others use a statistical approach to recognizing specific features of images, and others use colour information.

Chapter 4 describes how the texture blending system was actually implemented. The blending algorithm and Poisson interpolation are described in detail. The colour histogram and Discrete Fourier Transform information used to classify images are also described, along with other minor implementation details of the project. Results are described and several examples of textures generated by the blending tool are given.

Chapter 6 describes enhancements and changes that could be made to the texture blender. These include adaptive binning mechanisms for the histograms used to compare images, luminance models that more closely take into account human colour perception, and many alternative techniques that could either replace or supplement the colour and DFT information used to decide which images are most similar.

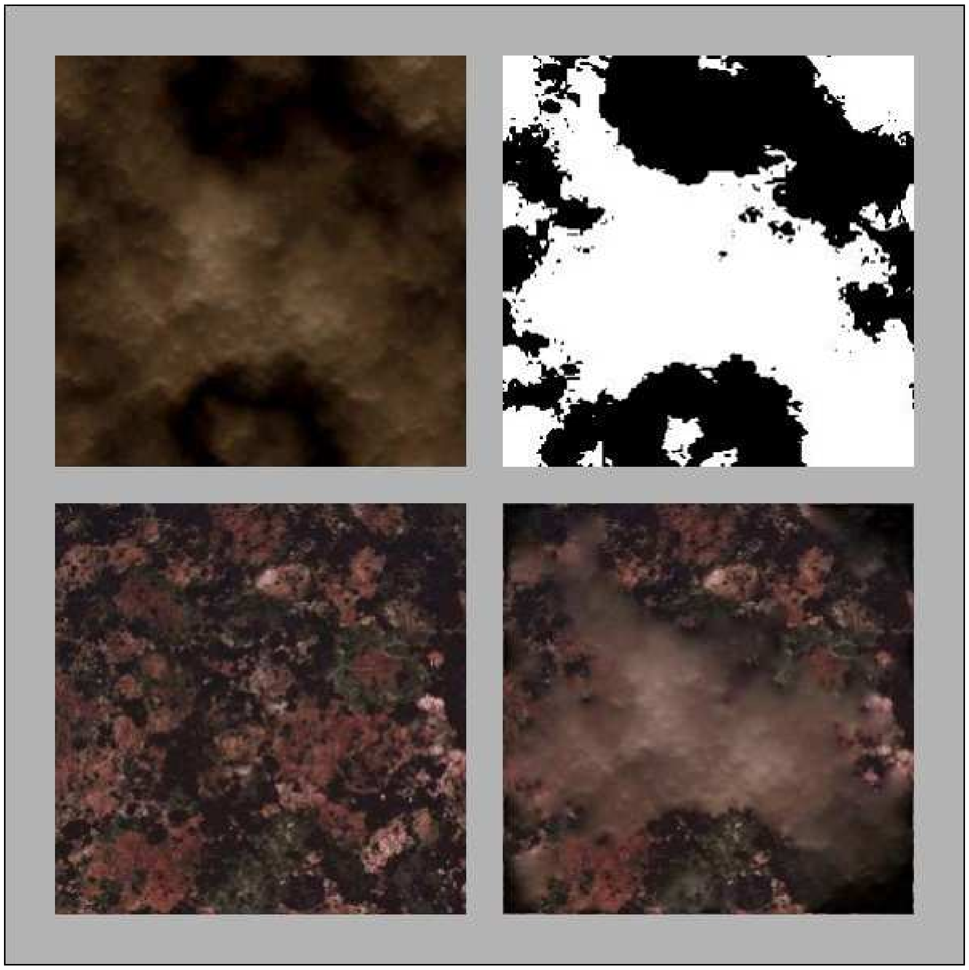


Figure 1.1: Sample output from the blending algorithm. The texture in the lower right was generated as output. The two other textures and the black and white mask were provided as inputs.

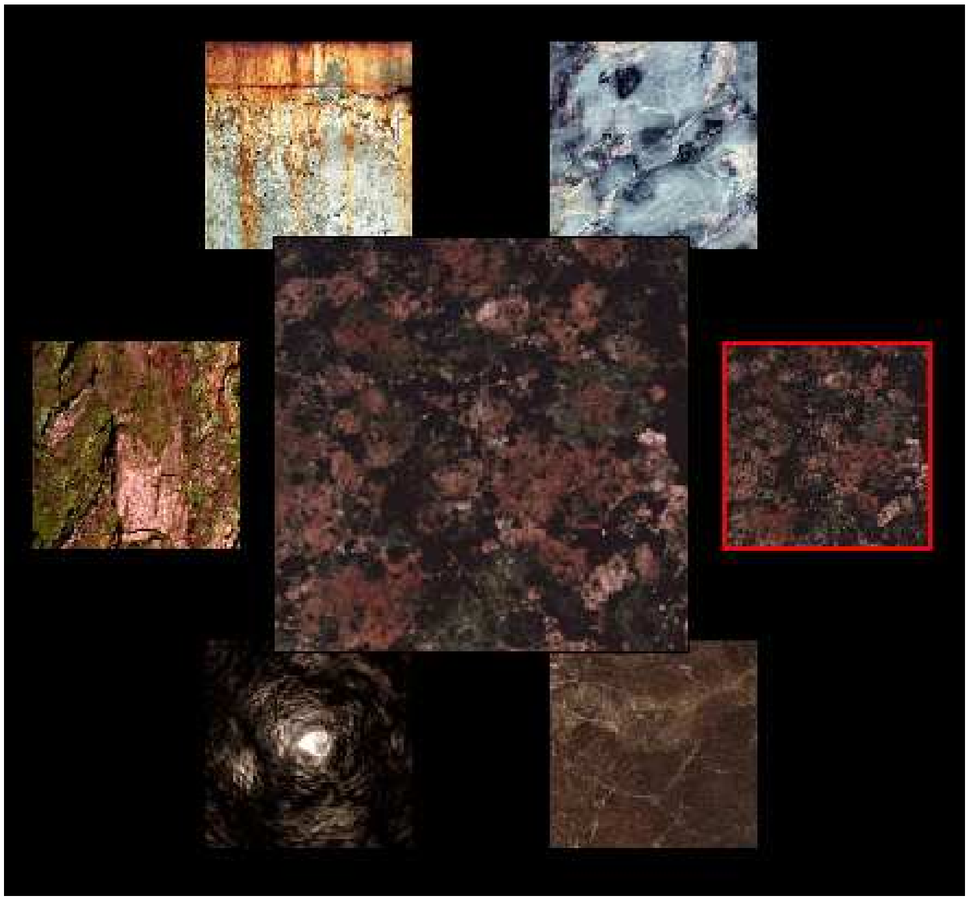


Figure 1.2: An example of a set of textures. The selected texture is outlined in red. Other textures were judged to be similar by a selection algorithm and were drawn from a database of diverse images.

Chapter 2

Textures

The term "texture" can loosely be described as an n-dimensional signal that exists within some finite domain. Textures contain some degree of repetition. For many applications, textures are repeated end to end. In order for a texture to be seamlessly repeatable, the differences across boundaries of the repeated tiles or segments must fall within the range of transitions that are found within the domain of the texture. This property can be inherent in a procedural texture generated by an algorithm or it can be created after the fact by hand or by applying an algorithm. Textures can define sound sequences, volumes, static two-dimensional images, or images that change over time (see [1]). One common application for static two-dimensional texture images is texture mapping, a process employed to add surface detail to computer-generated 3-D graphics, dramatically increasing the realism of rendered scenes. Sample texture images are shown in Figure 2.1.

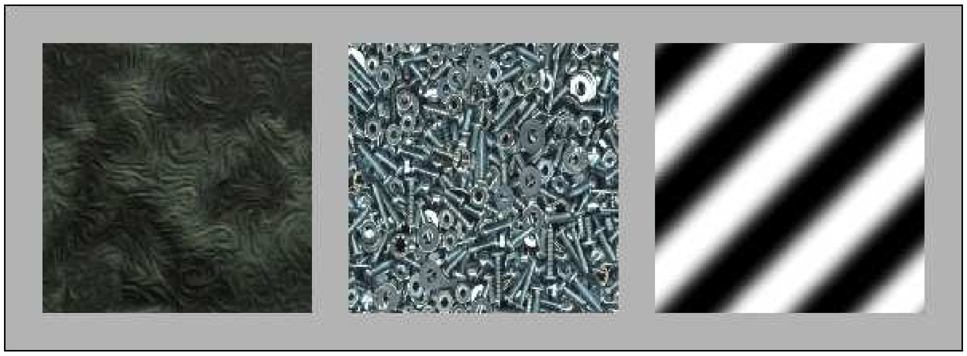


Figure 2.1: Some examples of tileable textures.

For the purposes of this thesis, the term "texture" always refers to seamlessly tileable digitized images. When textures do not tile seamlessly, there are jarring gaps in intensities along the edge of each tile that make it immediately apparent to humans that a given surface was generated with tiles and not as one large piece. The tiling process in 3-D computer rendering, as mentioned above, is known as texture mapping. Examples of surfaces modelled using texture mapping are shown in Figure 2.2. As a part of the texture mapping process, texture pixels, or texels, are used to define the values of a pixel that will be a part of a rendered polygon on the screen. Various interpolation approaches are used as a part of this process since there is rarely a one-to-one correspondence between a two-dimensional texture and a polygon that may be defined at any distance from and at any orientation relative to the viewer.

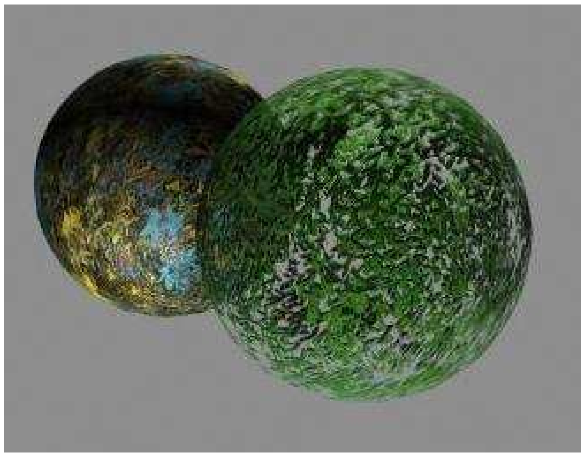


Figure 2.2: Texture mapping (left) and opacity mapping (right). Additional detail is added to the spheres by changing their colour based on a two-dimensional images rather than by using a large number of polygons. Image taken from [10].

Even two-dimensional textures used for the purposes of 3-D rendering can be used to represent many different kinds of data. For example, bump maps (see Figure 2.3) are textures which contain data that affects how lighting on a surface will be calculated. The fine local variations in lighting produced by applying bump maps give the impression of an uneven surface where light is reflected differently at different points. This is a much more accurate way of modelling real world surfaces than using a basic polygon with the kind of completely uniform surface that rarely exists in real life. The texture blending tool created for this thesis can also generate these other kinds of textures or maps, since they are still ultimately described as two-dimensional images, which are themselves simply two-dimensional arrays of integer pixel values. Visualization of the results of different texture mapping and 3-D rendering techniques, however, is outside of the scope of this project.

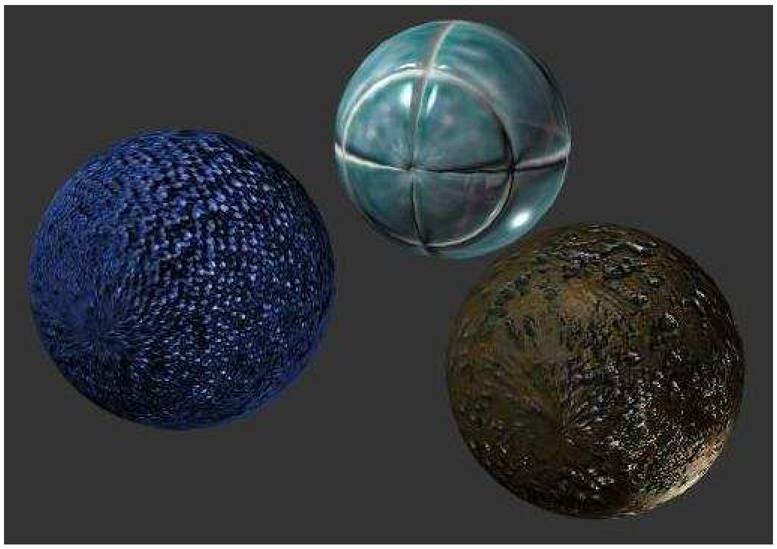


Figure 2.3: Bump mapping was used to give the appearance of small changes in the shape of the spheres. The bump maps define perturbations applied to the normal vectors used in lighting calculations. Lighting based on the altered surface normal gives the illusion that the orientation of the surface being rendered has changed. Image taken from [10].

Chapter 3

Related Work

3.1 Texture Blending

Textures can be generated through a variety of different methods. Often, textures are taken directly from photographs or are generated procedurally based on some known function. Another class of methods generate new textures by blending existing textures in some way. Blending algorithms vary, as does the degree of user interaction required in different systems. User interaction is an important factor to consider when the textures being created will ultimately be judged by human observers. Techniques requiring a high amount of interaction might fully rely on users to make selections of textures or regions of textures, while more automated systems can attempt to structure a database of textures and find similarities that would make them more amenable to blending. Ultimately, the value of a certain approach depends entirely on the type of texture that the user wants to generate. A large number of modelling applications attempt to model real world objects. In that case, the most desirable textures are those which give rendered shapes the most realistic appearance. In some other cases, textures are valuable as artistic tools and it is very difficult to say what kind of texture would or would not be considered useful.

An alternative approach to texture generation is described in [8]. In this system, textures are taken from photographs in order to provide a higher degree of realism. New textures are derived by interpolating between two existing textures. Textures can be thought of as vertices in texture space, and new textures are created by considering points along the lines that connect existing textures or vertices in that texture space. This is referred to as a simplical model since interpolation only takes place along a segment of an axis connecting two textures situated in texture space. An overall structure of points in space corresponding to textures and line segments connecting adjacent points is constructed by the system and is referred to as a simplical complex. To interpolate between textures, a warp function is used that attempts to

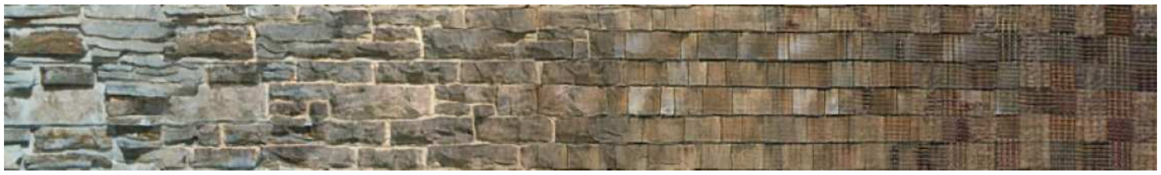


Figure 3.1: An example of the texture morphing method from [8]. This image was obtained by smoothly interpolating between four textures along a single path.

align features in the final result with features from the inputs. Additional techniques are used to maintain sharpness. Navigation and selection of new textures for interpolation is accomplished by moving between neighbours in the texture network. Sample results for this blending technique are shown in Figure 3.1.

The morphing approach itself uses a statistical approach that analyses the distribution of colours within regions of the input images. The regions are defined by a circular window divided into two. As the window is rotated, the properties of the colour distributions in the two half windows are compared. The importance of individual pixels to the overall calculation of the colour distributions is defined by a Gaussian distribution. The standard deviation for the distribution is chosen manually for each texture. The comparison algorithm generates feature maps for each texture and a warp computation is used to interpolate between two feature maps while minimizing alignment errors.

This approach to texture morphing does generate convincing results for certain types of textures but differs from this project in many significant ways. First, the morphing technique requires that source images be fairly similar and structured. It is unlikely that this technique would generate good results from dramatically different textures where features could not easily be aligned. Attempts to preserve sharpness may also be problematic for textures without high-frequency components. Next, there is a major difference in the way user input is handled. For the morphing approach, users must walk between adjacent textures and must manually define the weighting between two existing textures. For this project, the user simply makes selections of textures and the texture sets presented evolve over time.

Texture Mixing and Texture Movie Synthesis using Statistical Learning [1] describes another approach to texture blending. This approach relies on wavelets, which are used to subdivide the input textures into various frequency components. These components are structured using a tree and an algorithm is used to merge different components of the two input structures. This is very different from the Poisson blending approach used for this project. In [1], some convincing results are presented but they are mostly for very special cases, such as when the two source textures contain the same image at two different scales. Texture movies are also created using the same texture synthesis method.

3.2 Image Classification

In addition to these texture blending approaches, many methods of classifying images have been developed and compared. Classification can be performed based on colour and intensity information or frequency information derived from Fourier or wavelet transforms. Intensity in this case refers to the brightness of an image. Using a standard RGB colour model, higher RGB values correspond to brighter colours. Furthermore, luminance values can be calculated from RGB values to take into account the fact that the human eye's sensitivity to light varies depending on its colour [9]. Classification can also be performed by explicitly attempting to match specific features extracted from either of two images that are being compared. It is easier to objectively evaluate image classification methods than it is to evaluate textures since the classifications generated computationally can be directly compared to similar operations performed by humans.

Feature comparison is a popular way to measure similarities between images. In [5], blob-like features are extracted using a statistical approach that isolates image regions by comparing the changes in intensity between pixels. Regions are chosen based on the percentiles of intensities found within the image, with the most extreme intensity values tending to form blobs that are compared to measure similarity. The comparison process is statistical, relying on different measures of position, size, and shape. This method of analysis was judged based on its ability to accurately predict the category of an image taken from a library.

Another approach to feature analysis is described in [6]. Rather than attempting to isolate entire regions, the properties of high-contrast boundaries within an image are evaluated. These boundaries form unique contours that can be compared. Relations between contour segments such as distance and relative orientation can also be compared in a way that can account for scaling or changes in orientation.

These two methods are effective when it comes to analyzing certain types of images but neither one would be well-suited for the types of images that are used to create textures. In the first paper, percentile blobs were extracted from highly-structured scenes containing discrete objects. The second paper attempted to identify single objects such as a fish or a wrench. These images are very different from the types of images commonly used as textures. Many textures contain a high number of small elements that are all fairly similar, such as bricks or grains of sand. An approach that attempted to find regions in such textures would discover a large number of components too small to compare meaningfully. When comparing textures we also want to know more than what kinds of components are present; the actual objects depicted in a texture may be less important than its colour, scale, and contrast.

Rather than attempting to explicitly characterize the components contained within an image, other methods involve using Fourier or wavelet transforms to derive some information about the variation of intensity values within an image (see Section 4.3.2 for more details). In [7], the amplitude of the results of the Discrete Fourier Transform of images are compared using histograms, mirroring a technique used to compare colours in images. An explanation of how histograms are created from images is given in Figure 3.2. The Discrete Fourier Transform itself uses a binning process like histograms to discretize the resulting frequency domain information. This is extended by generating adaptive histograms that take the amplitude of the Fourier data into account. The accurate results obtained by this approach are particularly relevant since textures were compared rather than conventional photographs of entire scenes.

In [15], Fourier transforms are applied to analyze changes in intensity between pixels and their 8-pixels neighbourhoods. Overall textures are compared using a standard histogram method and the final results obtained were comparable to other standard methods of texture comparison.

The photo collage system described by [13] takes a hybrid approach, using both colour information and edge mapping. The goal of the system is to approximate a

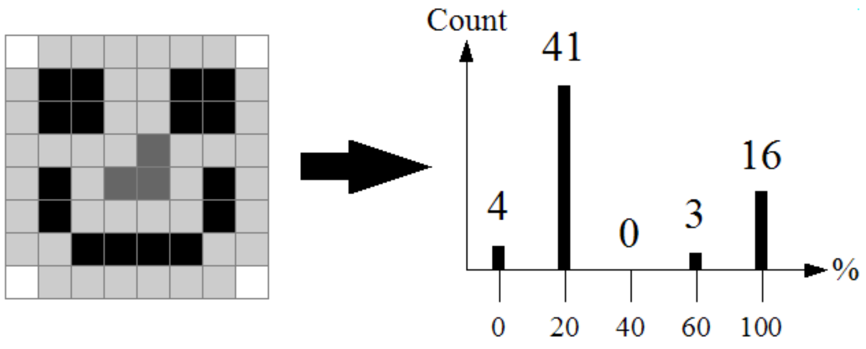


Figure 3.2: A histogram is created representing the intensity values of the 8x8 pixel image shown above. Counts are taken of the number of pixels that fall within a certain range of intensity values. These ranges are referred to as "bins". Real colour histograms are based on where pixel colour values fall within the full three dimensional RGB colour space. Bin sizes are arbitrary but there is a tradeoff between precision and the number of bins that must be processed when comparing the histograms of two images.

given image by tiling a large number of much smaller but still decipherable photographic images (see Figure 3.3). Colour is used both in the form of standard colour histograms and the average colour of an image. Edge mapping is the process of fitting polygons to high-contrast edges found within the images. The polygons making up the edge maps can then be directly compared.

This project again solves fundamentally different image classification problems. Images are classified in order to construct collages in which they may appear very small. Aspects such as average colour are very important at that scale since they

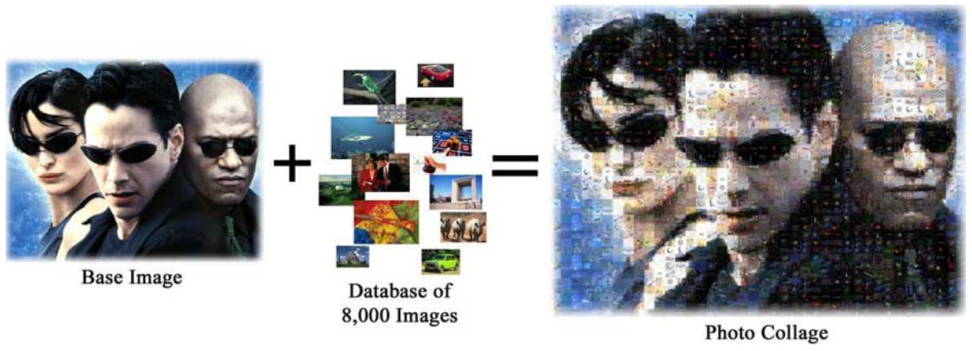


Figure 3.3: A photo collage created by the system described in [13].

closely model what a human sees. The colour histogram comparison, however, is also a suitable way to compare textures since colour is a major factor in image similarity that is largely missed by frequency analysis.

Chapter 4

Implementation

4.1 Overview

This section describes the implementation of the texture blender. The blender provides two primary functions. First, it allows a user to generate new textures based on two selected source textures and a mask. In addition, the blender can generate sets of textures from which the user chooses inputs for blending. A number of different components are used to create this system: the blending algorithm, a library of textures, algorithms for generating masks and candidate texture sets, and a user interface. A screenshot of the system is shown in Figure 4.1. It has a simple interface that presents the user with controls for texture and mask selection, a button to apply the blending function, and a number of additional controls that can be used to present new sets of textures and masks from which to make selections.

The texture blender window is dominated by three circular sets of images each set around a larger central image. From left to right, these sets of images are controls that allow the user to select the first input texture, mask, and second input texture. Selections are performed by clicking on an image. Once a selection is made, that image is displayed in the centre of the control. Initially, only a small number of potential textures and masks are presented to the user. The user can change these sets by manipulating the row of controls at the top of the blender window. The user can request a new set of candidate textures and masks by clicking on the "Suggest New Textures" button. When the user clicks on this button, an algorithm is applied which compares the current selection in each control to every texture in a library that is loaded when the blender is started up. The textures presented in the new candidate sets are chosen by an algorithm that computes the degree of similarity between textures. The most similar textures are returned. The algorithm takes several parameters that can be set through the user interface, such as result randomness, the size of the sets that will be returned, and what kinds of descriptors will be used to

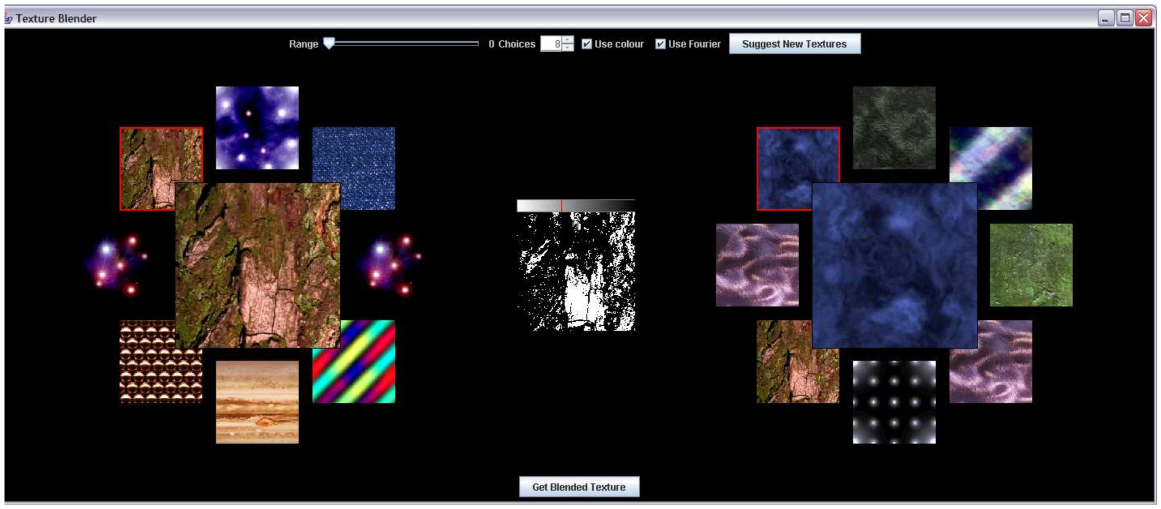


Figure 4.1: A screenshot of the blending application. The three circular groupings of images allow users to select two textures and a mask to be blended together. The controls near the top of the window allow the user to request a new set of textures to work with.

compare textures. Details about these descriptors are given in Section 4.3.

To select a new texture set, textures are placed in an ordered list based on distance from the current selection. Textures are sampled using a randomly generated number that is mapped to an integer texture index based on a desired probability distribution to influence the likelihood of choosing the various textures. When a texture is chosen it is removed from the list to avoid duplicates and the process is repeated until a set of textures of the desired size has been generated, at which point it is displayed for the user.

When the user chooses new textures with zero randomness, the probability of selecting the first texture is set to one and the probability of selecting the remaining textures is zero, ensuring that the closest textures will always be picked. When the user wants fully random results, the probability distribution is uniform over the texture set and at each step every texture is an equally likely choice. Intermediate randomness values are achieved by transforming between these two probability distributions.

This non-deterministic approach to texture selection allows for the possibility of new directions for texture blending not related to the current selections. This approach to generating random results also has the virtue of taking distance into account, so image distances contribute to the calculation of the probability distribution for selecting each output image.

Other interactive components of the texture blender include the slider that enables users to determine mask cutoffs, the option that allows them to control the size of the sets of suggestions presented, and the Fourier data display functions. Users can choose to display the real component of the DFT of the the current selection by right-clicking on it or the imaginary component by centre-clicking. The intensity values of the image shown are scaled logarithmically in order to make the transitions between values more visible. Since outlying DFT values can be several order of magnitude larger than the most common values, a linear scaling would have the effect of generating a mostly black image.

The mask cutoff determines which intensity values in the selection will be rounded to white and which will be rounded to black in the mask used to combine the two input textures. The effect of changing this cutoff is shown in Figure 4.8.

4.2 The Blending Algorithm

The blending algorithm used by the texture blender was implemented for [11] as a standalone command-line application. That application was used to perform the blending for this project. The application takes as input two images of equal size and a black and white mask. As output it creates a third image which smoothly combines the two originals using the mask to define which regions of the resultant image will be drawn from which of the originals. This smoothing is done using interpolation based on solving Poisson equations, and generates much more convincing results than simple layering, where user-defined pixel-by-pixel selections are drawn on top of one another without interpolation. Layering can be improved somewhat with techniques such as blurring along seam lines applied after the fact, but normally the seams remain very noticeable if the combined elements are at all dissimilar. The algorithm also ensures that rectangular images can be tiled seamlessly, allowing them to be used as textures. It is fully described in [11] and was used to convincingly combine visual elements such as objects or surfaces from different images. Mask-based texture blending is a novel application of this algorithm.

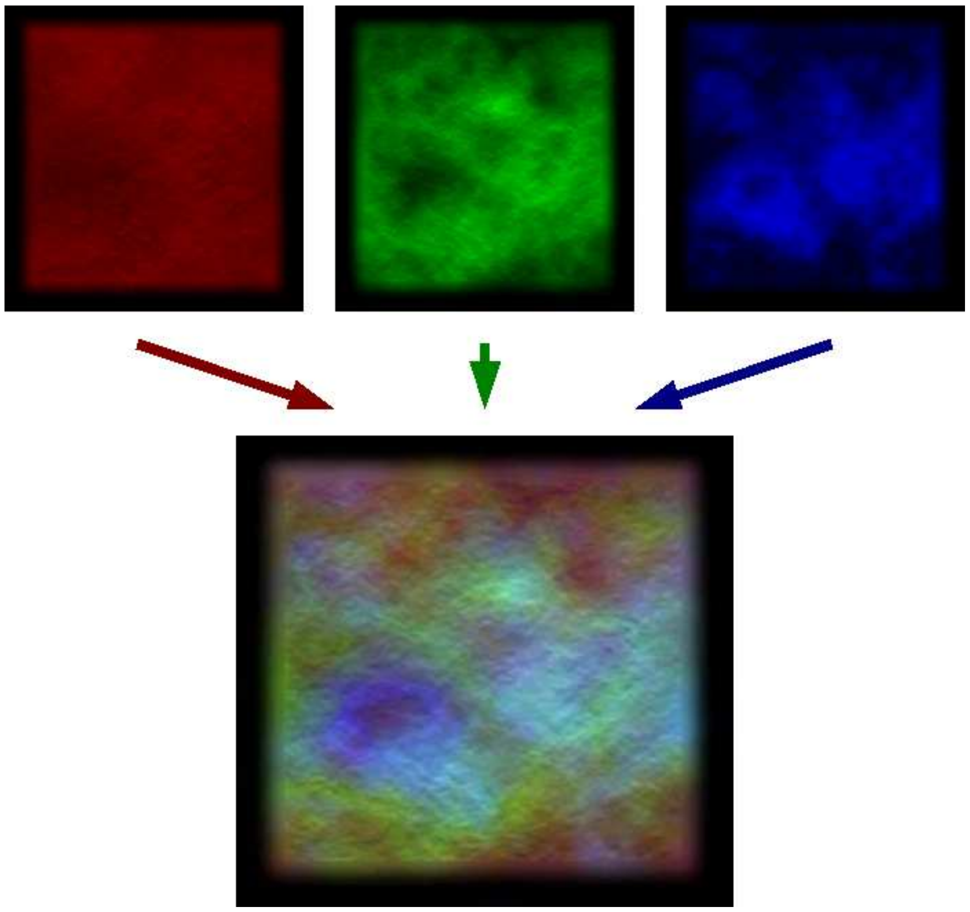


Figure 4.2: Red, green, and blue colour channels are combined to form an image containing pixels with mixed colour values. Pixel values for each colour plane can be considered integer values with a magnitude corresponding to the brightness of the pixel. Pixel values for the final image are defined as triplets of integer values, with one value being drawn from each of the corresponding points on the three colour planes.

For the purposes of the texture blending algorithm, each colour channel of the input textures is considered separately. This means that the red, green, and blue colour planes can each be considered functions defining single scalar values over the domain of the image (see Figure 4.2 for a sample image divided into colour channels). The problem of blending two textures amounts to the problem of independently blending their red, green, and blue components and then combining the results to form the output texture.

The result of the Poisson image editing process is a Laplacian, a function over the two-dimensional domain of the image with values corresponding to its second derivative [4]. Rapid changes in intensity in an image produce correspondingly higher values in its Laplacian. These values are of particular interest since second-order changes within an image are highly noticeable. Very gradual changes in image intensity are less noticeable and, conveniently, produce very small changes within the Laplacian (see Figure 4.3). Although the Laplacian values specified describe the rate of change of changes in intensity, when combined with actual border values the Laplacian can be used to fill in the rest of the image. This is possible because the border values give a fixed starting point from which it is possible to work inwards, finding successive intensity values such that the second order derivatives are the values of the Laplacian. The texture blending application uses the Laplacian defining the interior gradient of the second source texture to blend from the first source texture, generating the output texture.

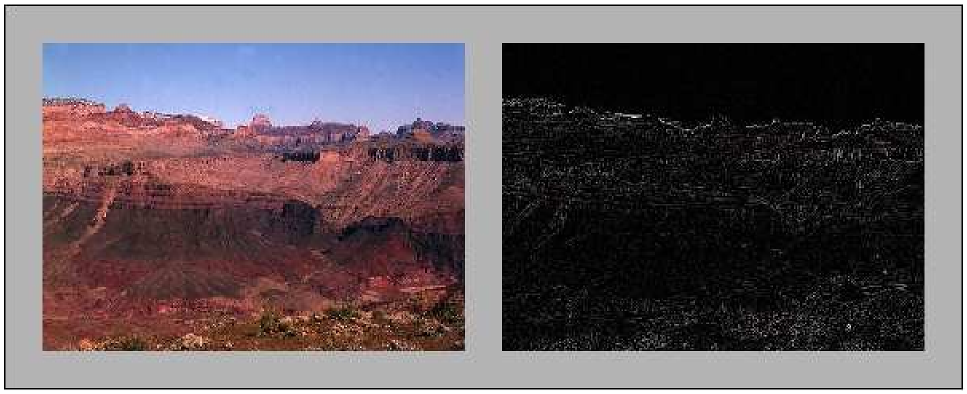


Figure 4.3: A scene before and after the application of a Laplace filter. High-contrast borders such as the outline of the top of the mountains produce high values in the Laplace form that correspond to lighter parts of the filtered image. Interior parts of the mountains and sky where contrast between pixels is lower are black. The difference between low- and high-contrast image regions is much more exaggerated than in the original image.

The gradient used to derive the final result of the algorithm is calculated by a Poisson partial differential equation. The boundary conditions required to generate a unique solution are known as Dirichlet boundary conditions. In general terms, they simply specify which values a function must take along the boundaries of its domain.

The solution of a partial differential equation is a function, so these constants are required to derive the unique set of fixed values that will define the final output image. Partial differential equations are relations between an unknown function of independent variables and its partial derivatives with respect to those variables [14]. Partial derivatives themselves are derivatives with respect to one variable in a mathematical term where all others are considered constant. Solving a partial differential equation amounts to finding the functions for which the given relation holds.

The Poisson equation can be thought of as solving a minimization problem: the solution defines the closest gradients to a certain guidance vector field while satisfying the boundary conditions. Equation (4.1) formally defines this minimization problem. Intuitively, the function describes the process of finding interior values on the image domain Ω with the given gradient operator ∇f, a guidance vector field v taken directly from the gradient field of a source image, and the boundary conditions.

$$\min_{f} \int\int_{\Omega} |\nabla f - v|^2 \text{with} f|_{\delta\Omega} = f^*|_{\delta\Omega} \tag{4.1}$$

In order to ensure that the final output texture is tileable, identical Dirichlet conditions must be enforced at opposite sides of the image domain. The east and west boundaries must be identical, as must the north and south boundaries. Because gradients are minimized over the surface of the final image, the corners are also seamlessly blended together and the tiling is convincing.

4.3 Image Classification

Another major aspect of this project is the problem of image classification. Humans are capable of classifying images based on a wide number of factors. They can quickly tell the difference between two images or recognize common image features even if they have been substantially altered in terms of colour, orientation, shape, size, or even partially occluded. They are also able to quickly perceive areas of contrast and discern patterns that they can match to other patterns that they remember seeing before. The combined features of an image form a mental impression that allows humans to categorize and compare textures. In order to make similarity decisions meaningful to a user, textures must be structured and compared in a way that generates similar results to what a human observer would produce. The advantage of using a computer to do the comparison is that a large database of textures can be analyzed much more quickly.

Computer images are displayed to users as arrays of coloured pixels. Humans can observe two general characteristics of these pixels: what colour values they have and how the colours vary across the surface of the image. As mentioned earlier, humans are keenly aware of both first- and second-order changes in intensity or colour values across the surface of images. In order to model human perception of images, the distribution of colours is compared along with Fourier data, which is related to the changes in colour found within an image.

Texture colour and Fourier data are used to compare textures. Colour data is drawn from the initial RGB representation of the images, and Fourier data is obtained by applying a Fourier transform to the intensities of the image pixels, which are equivalent to a monochrome greyscale image. The Fourier data provides a representation of the frequency and scale of elements present in the image. These two types of data can be used to define two dimensions in a texture space. Texture similarity can be judged by considering the Euclidean distance between corresponding points in texture space. In order for this distance to be meaningful, the two distances must have a similar scale. This was accomplished by transforming distances along each axis onto the interval [0,1], where the longest distance between textures in a single comparison is given a value of 1. This approach preserves the relative distance values and produces good results consistent with what a user would expect but is somewhat arbitrary. One alternative would be to allow users to weight the distances along the various axes, but for this to work they would need to try out many different weightings to get a feel for how they influence the results. This is a difficult problem inherent to any system that attempts to mix similarly heterogenous types of data to derive a single scalar value.

4.3.1 Comparison of Colour Histograms

Texture colours are analyzed using histograms, a common approach in computer graphics. A sample histogram is given in Figure 4.4. Colour histograms are constructed by first dividing the three-dimensional RGB colour space into an arbitrary number of bins. A larger number of bins allows for more accurate comparison but is slower. Comparing the full colour set of a 32-bit image with 8-bit red, green, and blue intensity ranges requires comparing 2^{24} separate values, which would severely limit the number of textures that could be compared in real time given the capabilities of current hardware. To mitigate this problem, each dimension is divided into 32 ranges, which means that there are only 32,768 comparisons when calculating the difference between two different histograms. This is 512 times faster than a basic approach where every single colour is considered. In the blender code, the number of bins is defined as a constant that can be altered based on the speed of hardware being used.

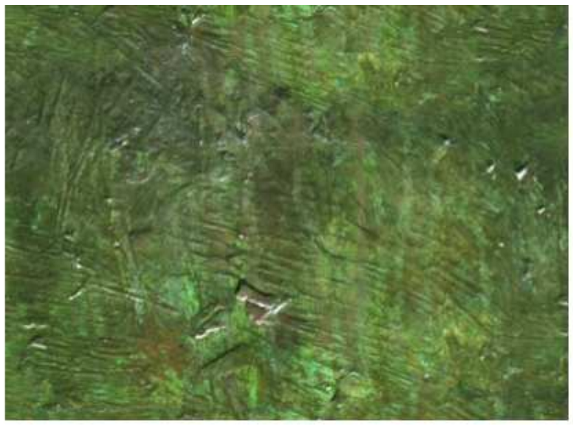
A colour histogram consists of bins with integer counts corresponding to the number of pixels in the original image that fall within a given range. These histograms do not contain any information about the spatial distribution of pixel colours within an image. An image consisting of black and white regions of equal size has exactly the same colour histogram representation as an image where the same number of pixels of the same colour are distributed differently (see Figure 4.5). This is why Fourier data is used in conjunction with the colour information. The Fourier data is obtained by applying a Discrete Fourier Transform to every image and then directly comparing the the results pixel by pixel. The Discrete Fourier Transform operation is described in the next section.

The difference between colour histograms is interpreted as a distance. For two textures A and B with colour histograms C_A and C_B with a bin count of c, the distance is defined as:

$$distance_{colour} = \sum_{r=0}^{c-1} \sum_{g=0}^{c-1} \sum_{b=0}^{c-1} |C_A(r,g,b) - C_B(r,g,b)| \tag{4.2}$$

4.3.2 Comparison of Fourier Transforms

Fourier transforms transform an image in the spatial domain into output that represents the image in the frequency domain. For a two-dimensional image, the output of the Fourier transform is also two-dimensional. The Fourier transform decomposes an image into its sine and cosine components [3], with each pixel corresponding to a certain frequency, i.e. the distance in pixels between a transition from low to high



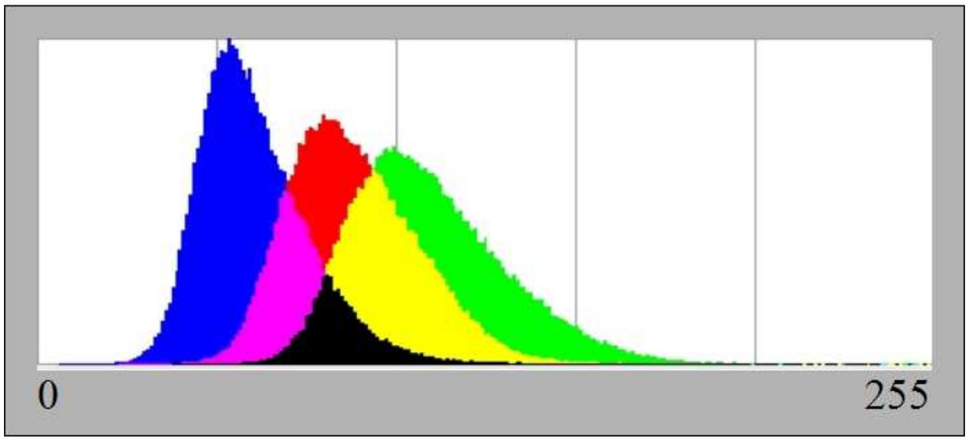


Figure 4.4: Colour histogram of the "bronze" texture shown above. The x-axis of the histogram represents intensity and ranges from 0 to 255. The green component of the pixel values present in the image is the farthest to the right because the pixels in the image tend to have higher green values, causing the overall image to appear green. The histograms used in the texture blending application take counts of pixels that fall into intensity ranges called "bins" rather than maintaining counts for every intensity value. The colour histograms implemented also use three-dimensional bins and consider the red, green, and blue values of each pixel together rather than separately as shown above.

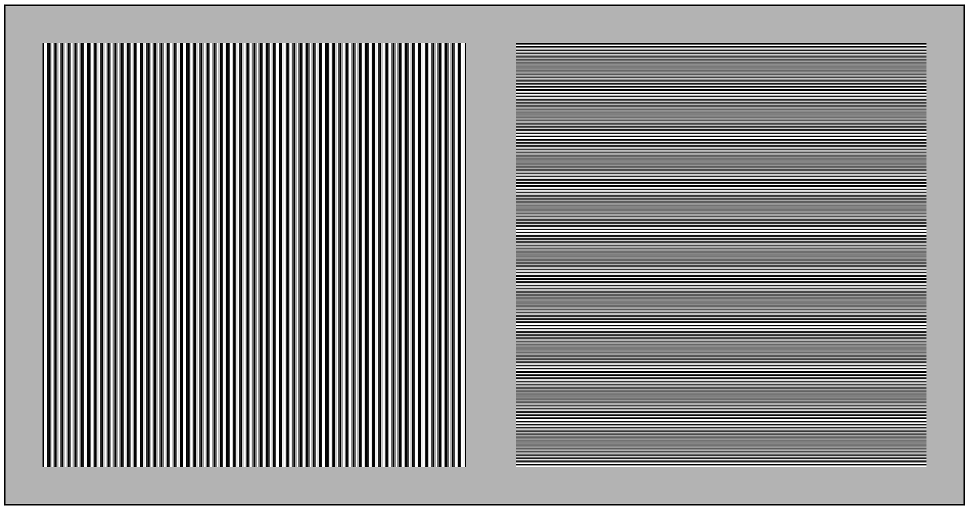


Figure 4.5: Two images with similar counts of black and white pixels and therefore roughly equal colour histograms. Despite the fact that the colour histograms are the same, the two images look very different since the frequency of stripes and the orientation differs between the two. Fourier analysis would pick up this difference.

colour intensity values and back. These intensity changes can be thought of as sine curves. The maximum range of these sine curves in 24-bit images is $[0, 255]$ rather than $[-1, 1]$ and is derived from the 24-bit RGB colour range, $[0, 0, 0]$ to $[255, 255, 255]$. The Fourier transform of an image gives us insight into its geometric characteristics and any periodic variations in intensities that it contains. An example of a texture and its Fourier transform are given in Figure 4.6.

The Discrete Fourier Transform (DFT) is a discretization of the Fourier Transform and is useful for textures since computer images are themselves discrete in nature. Each point corresponds to one specific frequency, and the number of frequencies depends on the size of the input image. The DFT operation takes as input the pixels of an n by m-sized image and produces as output a restricted n by m matrix of frequency values. For a square image of size n by n, the DFT operation for determining one value (x, y) in the matrix can be described by the following equation (see [3]):

$$F(x,y) = \frac{1}{n^2} \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} f(k,l) e^{\left(\frac{-2\pi i}{n}(ki+li)\right)} \tag{4.3}$$

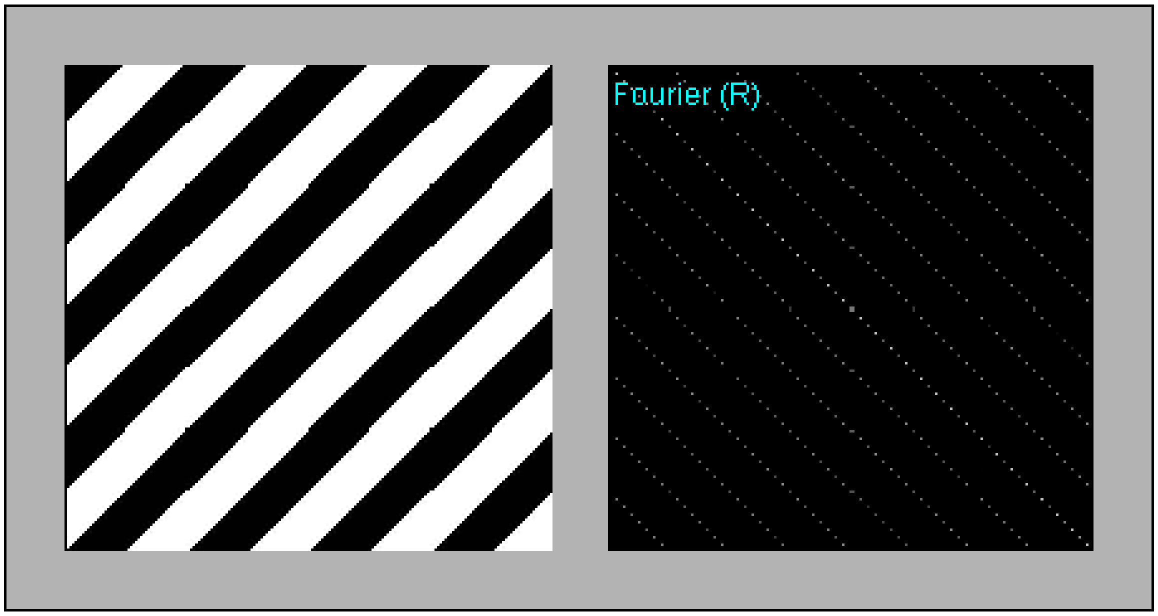


Figure 4.6: A striped texture and the real component of its Discrete Fourier Transform. The pixels that appear in the DFT correspond to sine waves of a given frequency and range that when added together produce the hard white and black stripes shown on the left.

The exponential term is the Fourier basis function. Each element in the DFT is obtained by multiplying the basis function by the entire image and summing the result. An existing implementation of the DFT was used for this project [12].

The DFT as defined above requires O(m^4) operations, where m is the size of the original square image in one dimension, but it is possible to reduce the running time of the algorithm to O($n^2 \log n^2$) operations by employing the Fast Fourier Transform. Many DFT implementations also require images that are square with the length of both sides being a power of two in order to optimize and simplify the algorithm. The DFT implementation used for this project employs the Fast Fourier Transform and is restricted to square images with width and height equal to a power of two (i.e. width = height = 2^n for an integer n). Since many texture-related algorithms found in both hardware and software used for rendering already typically requires textures of size 2^n for this is not a significant limitation.

DFT information is used in the blending system to compare the similarity of two different textures. This is accomplished by simply summing the difference of the DFT values of two textures A and B over their entire domain:

$$distance_{fourier} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} |F_A(i,j) - F_B(i,j)| \quad (4.4)$$

4.3.3 Distance Calculation

For the purposes of this project, distances are computed by considering two dimensions defined by texture colour and Fourier data. For each of the two dimensions a single scalar value is calculated by summing the difference across histogram bins or pixels. Distances in each dimension are then scaled to the interval [0,1]. Overall distances are calculated using the following equation:

$$distance_{total} = \sqrt{{distance_{fourier}}^2 + {distance_{colour}}^2} \quad (4.5)$$

This distance calculation can be scaled up for a texture space with more than two dimensions simply by adding the squares of the additional distance terms.

Textures are not placed explicitly at a certain point in texture space, but are defined in terms of each other. This is sufficient since the goal is to find out which textures are most similar. To do this, we simply need to compare the selected texture to every other texture in the library. Comparisons like this would not be sufficient to partition the texture space or to compare more than two textures. For that it would be necessary to associate each texture with a particular point in texture space. Colour data would somehow have to be transformed onto one or more axes, as would Fourier data. RGB colour is inherently three dimensional and so colour would likely need to be defined over three dimensions instead of one for comparison purposes, and some fixed number of absolute measures would have to be devised. One possibility would be average colour. Textures could easily be positioned in three dimensions in absolute terms based on this value. Other similar potential measures of colour, Fourier, and other kinds of data are given in Section 6.

4.4 Code Structure and Development

This section describes the implementation of the texture blender and related software components. The texture blender itself was written in Java. Blender elements can roughly be divided into two parts: classes that are used to model textures or implement texture-processing algorithms and classes that define application or user interface components. Several versions of the texture blending application were built, all using the same basic texture blending algorithm. Additionally, a tool was created to explore ways of generating masks used to combine textures and many small demos were created to test individual components later combined to create the texture blender.

The final texture blending application is made up of many components (a high-level diagram of the system is shown in Figure 4.7). First, there is the stand-alone texture blending application, which processes image files and produces a file as output. A Java "wrapper" class was implemented so that this application could be called from other parts of the Java code using only a single method with images as parameters. Next, there are components that handle the processing and storage of textures. These functions are necessary to quickly suggest sets of textures to the user based on some measure of similarity to a selected texture. The most basic of the processing components is the texture library component, which is a data structure that can be queried to return texture sets based on Euclidean distance. The actual distance calculations are defined by other classes and could easily be modified. For this thesis, distance calculations were performed using colour histograms and discrete Fourier transform information, but many alternative approaches are possible and are discussed in Section 6.

Other major components of the texture blending project include the custom user interface components and the code of the blender itself. These components were implemented by extending the Java Swing JPanel class; other user interface components were built using basic Swing components and layouts. They display an arbitrary number of textures or masks in a circular pattern similar to how numbers are arranged on a clock face. When the user moves the mouse over one of these textures it shifts out slightly, making it more visible. Users can select a texture from the control by clicking on it. The selected texture is displayed in the centre of the control. The user can also choose to view Fourier information for the selected texture by right- or centre-clicking on its full-sized representation in the centre of the control. This was useful for testing the Fourier transform and is also useful for analysis since an image's

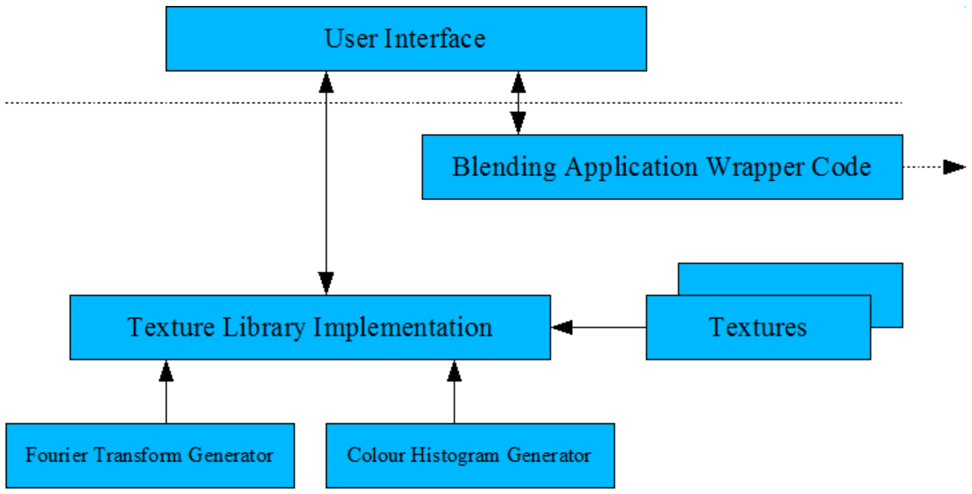

User Interface
Blending Application Wrapper Code
Texture Library Implementation
Textures
Fourier Transform Generator
Colour Histogram Generator

Figure 4.7: A diagram depicting the components of the texture blender and how they work together. The texture library is built from a directory of texture files. When the user requests new texture sets by manipulating the UI, the library performs the required distance calculations using the underlying Fourier and colour structures and returns a result. When the user requests a blended texture, the current selections are passed to the blending application via a wrapper class.

Fourier transform is not as obvious as its colour profile.

The basic circular texture selection control was extended slightly to allow users to select masks. The mask control similarly draws a set of masks in a circular fashion around a central selection. The potential masks drawn are greyscale versions of textures. In addition to the normal selection features, users are presented with a slider they can use to define the cutoff point for binarization of the greyscale texture. Figure 4.8 shows how different cutoff points affect how a mask is derived from an image. This feature is important because the intensity values of different textures vary dramatically. A fixed intensity cutoff would cause light textures to become completely white and dark textures to become completely black. A dynamic intensity cutoff based on the average intensity or other measures would not allow for as much flexibility as a slider. With the slider, the user is given far more control over the mask and therefore over the final texture produced by the blender.

The texture library is another important part of the blender. When they click on the "Suggest New Textures" button, the library is queried for new textures for each of the three controls. The current selections remain the same, but the other

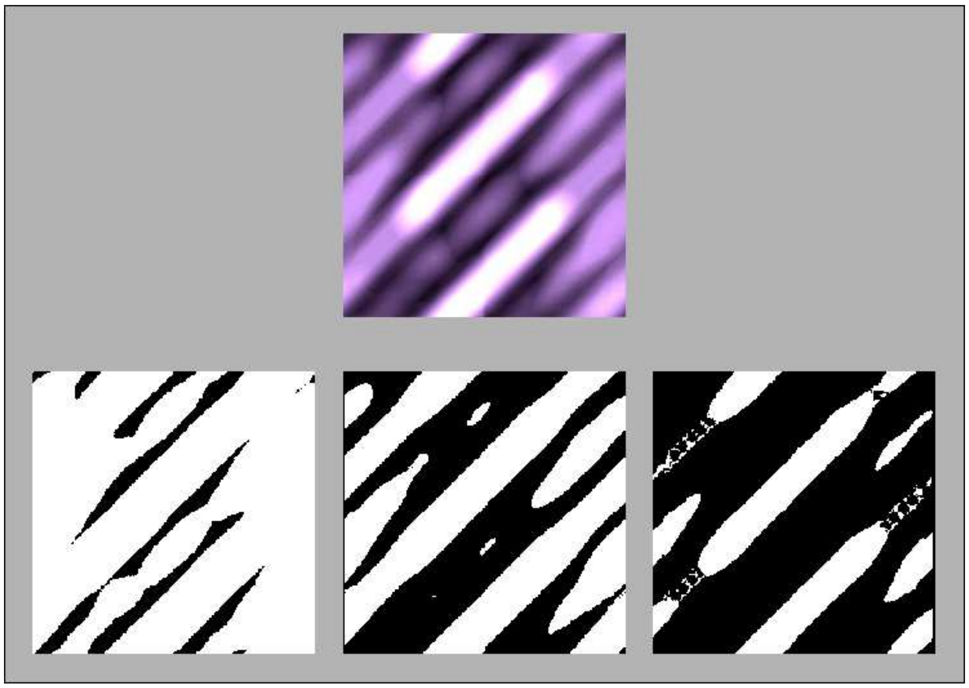


Figure 4.8: A source image is shown above. Below are binarizations of the image produced by setting the cutoff at 25%, 50%, and 75% brightness.

textures and masks displayed are the results of the library query. The library itself is built on startup when the user is prompted to select a directory full of textures. Information such as colour histograms and Fourier transforms is derived for each texture and stored to speed up query response times, which are dependent on how quickly distance calculations can be made.

Earlier implementations of the blender are included in the project and used slightly different approaches to mask selection and library queries. The original version included a much more limited mask control that was tied to one of the circular texture controls. When the user made a selection from the first control, the mask source was changed to that texture. The user could then only manipulate the intensity cutoff for that greyscale image using the slider. The newer version still allows the user to select one of the textures from either of the two controls as a mask source but also allows for a completely different selection, which is much more flexible (see Figures 4.9 and 4.10). Similarly, the original version included a field where users could enter "ranges" for library queries. The field that defines the query set size in the final version was still included but defined the minimum number of textures that would be returned.

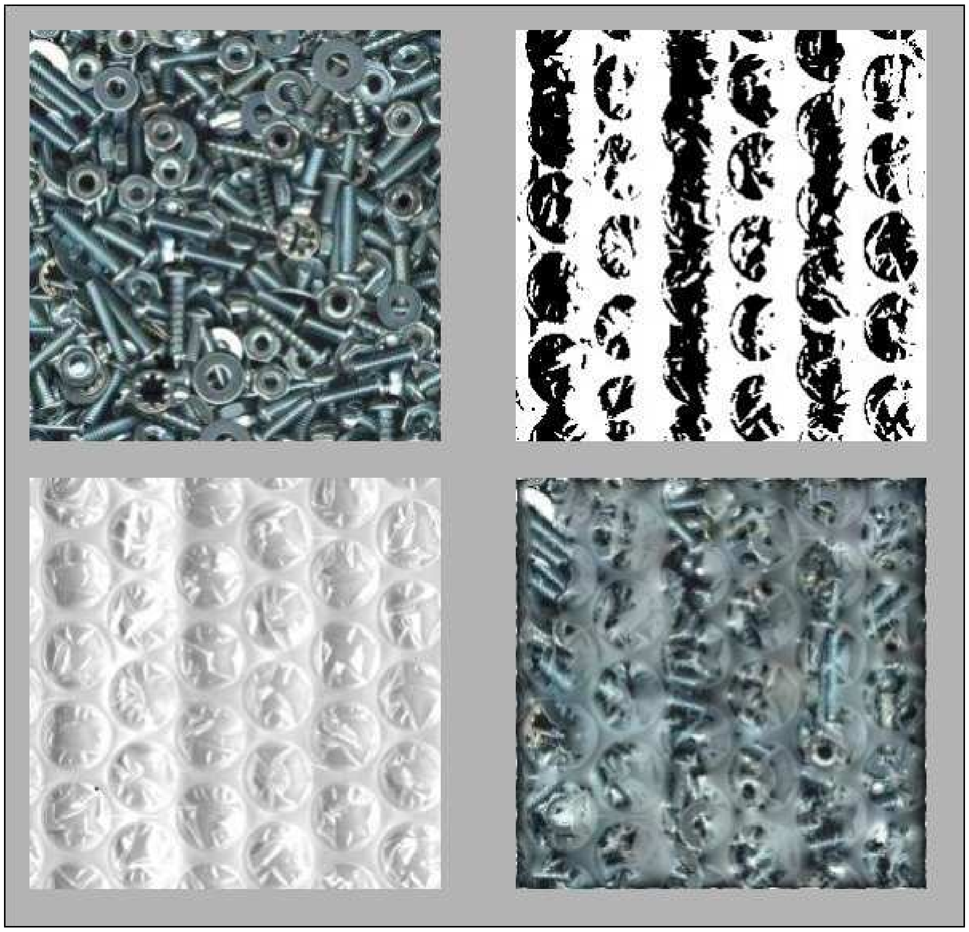


Figure 4.9: The output texture (lower right) was generated using a mask derived from the bubble wrap texture. The constituent regions of the output are chosen based on the structure of the bubble wrap texture.

A query in the original texture blender would return the n closest results, where n was defined as the maximum of the number of textures within the given range of the selection and the minimum result size. Users could not introduce randomness to the results, so to obtain texture choices far away from any current options they had to increase the range, potentially causing a large number of textures to be returned.

Finally, in addition to the blender, a small tool was created to explore how different filters could be applied to textures to generate masks used for the blending algorithm. Ultimately this was not used to build blender itself. The project also includes a number of demos of components such as the custom texture controls or the Fourier transform functions.

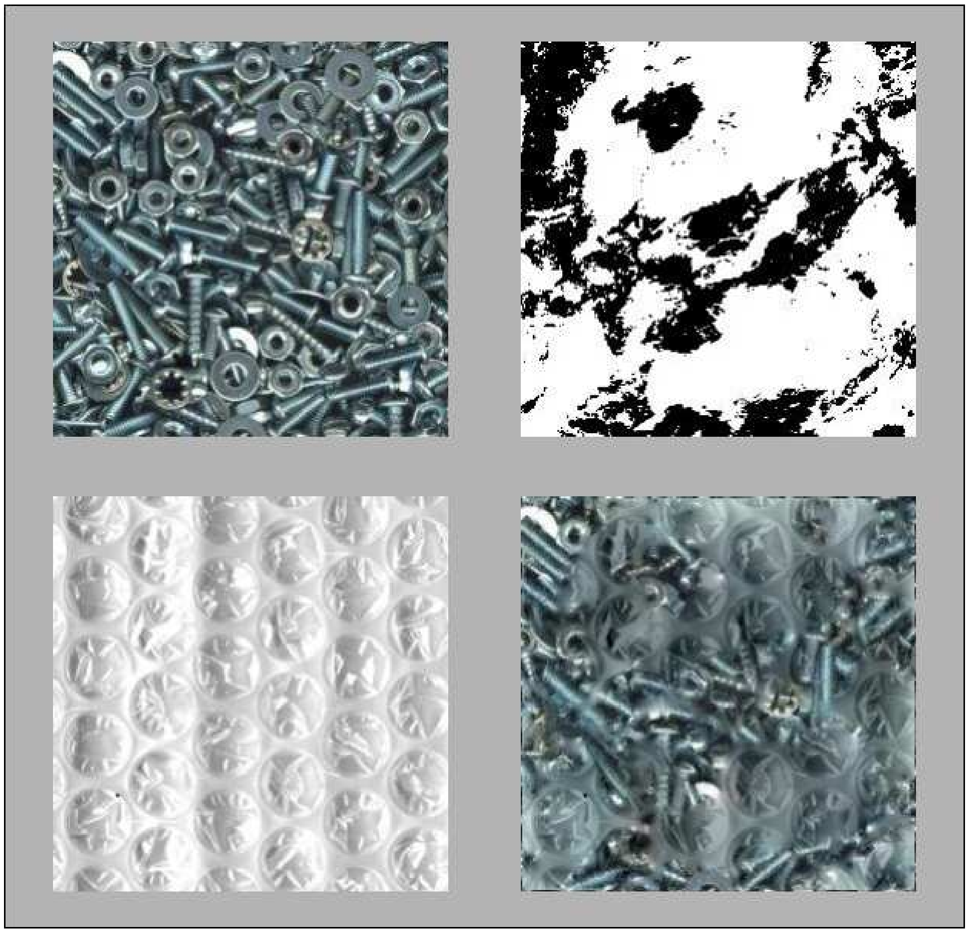


Figure 4.10: The output texture (lower right) was generated using a mask derived from a third image. The regions drawn from either source texture are independent of the structure present in either of the two images.

Chapter 5

Results

The primary product of this thesis is the texture blender, which has been successfully completed and can produce very realistic output textures. The tool can be used to work towards some desired final texture by iteratively selecting textures and masks and then requesting new sets of candidate textures to choose from. One can also explore how the blending procedure works by quickly trying out different textures and masks.

Sample blending results from different sets of input textures and masks are given in Figures 5.1 to 5.5. In these examples, the Poisson interpolation technique produces convincing output, preserving relevant parts of the structure of the two images and combining them using gradients that do not create the impression of any harsh edges. As a result, these images look more or less natural, as if they themselves could have been taken from photographs.

Not every type of input texture is well-suited to this texture blending technique, and sometimes specific pairs of images and masks do not go well together. Ultimately, the value of a given texture depends on its intended use, but certain kinds of input textures are likely to produce better results than others. Images that are highly structured over their entire surface such as the mountain scene in Figure 4.3 do not tend to create good textures since they cannot be tiled convincingly. Few surfaces have the strong periodic patterns that would accurately be modelled by those types of textures. Images with very sharp contrast between neighbouring regions can also be difficult to blend convincingly.

Furthermore, some highly geometric procedural textures are not well-suited to this technique. Sharp edges are not well-suited for blending and the basic method used to compare Fourier data can easily over-estimate the difference between textures with DFT values that spike sharply at different points. Similarly, textures with narrow or nearly-identical colour distributions are not well-suited to the colour histograms implemented since the histogram bins are of equal size over the entire colour space. Adaptive histograms would overcome these limitations, avoiding comparisons of irrelevant areas of the colour space and increasing the number of bins devoted to the actual colour distributions of the textures.

It would be very difficult to measure the desirability of the textures generated by this technique since the output is so specific to the huge number of possible input textures and masks. The value of the resulting textures depends highly on the application they are used for. Some realistic textures have been generated using this approach, and they could be used to model surfaces that cannot be photographed directly. Other textures generated are simply visually interesting and are still valuable despite the fact that they do not correspond to any known real world surface.

Similarly, no formal attempt was made to measure how well the texture blender's image classification system works. Intuitively, the texture sets do seem appropriate, as shown in the sample results (Figures 5.1 to 5.5). When comparing colour data, the blender suggests textures of similar colour, and when comparing DFTs the blender suggests textures with similar features. For example, the Fourier measure for a striped texture will tend to return other textures that also have stripes, star textures will be matched to star textures, etc. When both measures are used together, the potential images presented make sense and closely match a set that a human might come up with. Ultimately this is one of the best measures of how well the system works since the goal of image classification is to reproduce the way humans associate and analyze images. Figures 5.6 to 5.7 show a small set of similar textures drawn from a larger texture database using three measures of similarity: combined Fourier and colour data, just Fourier data, and just colour information. The benefit of creating an automated system to perform this kind of texture selection is that it is can be much faster than a human. Hundreds of textures can be sorted through in a few seconds with the texture blending tool.

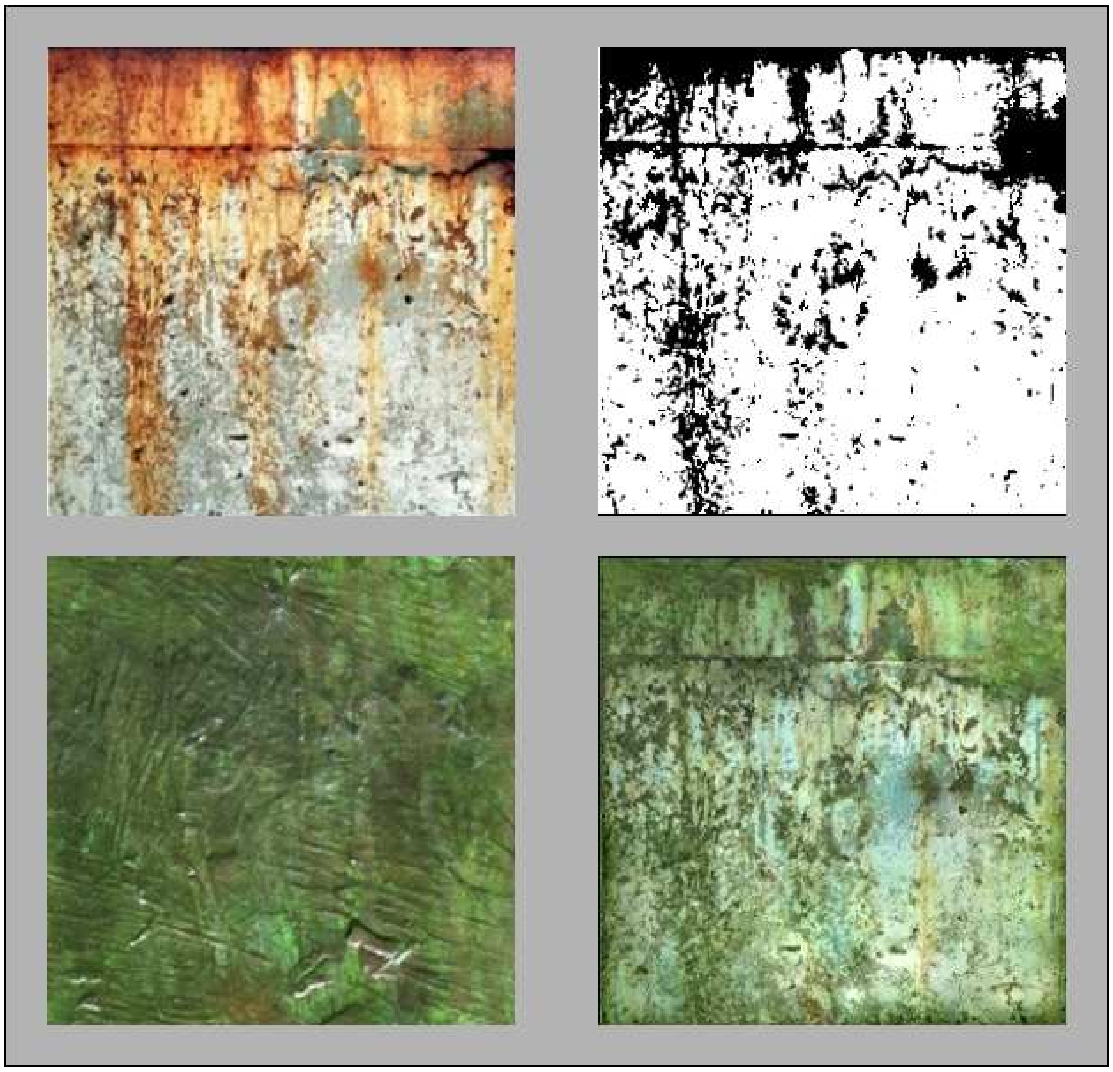


Figure 5.1: Sample texture blending results. Two input textures, one taken from a picture of a bronze surface and another from a picture of a rusty concrete surface are combined using the mask shown in the upper right.

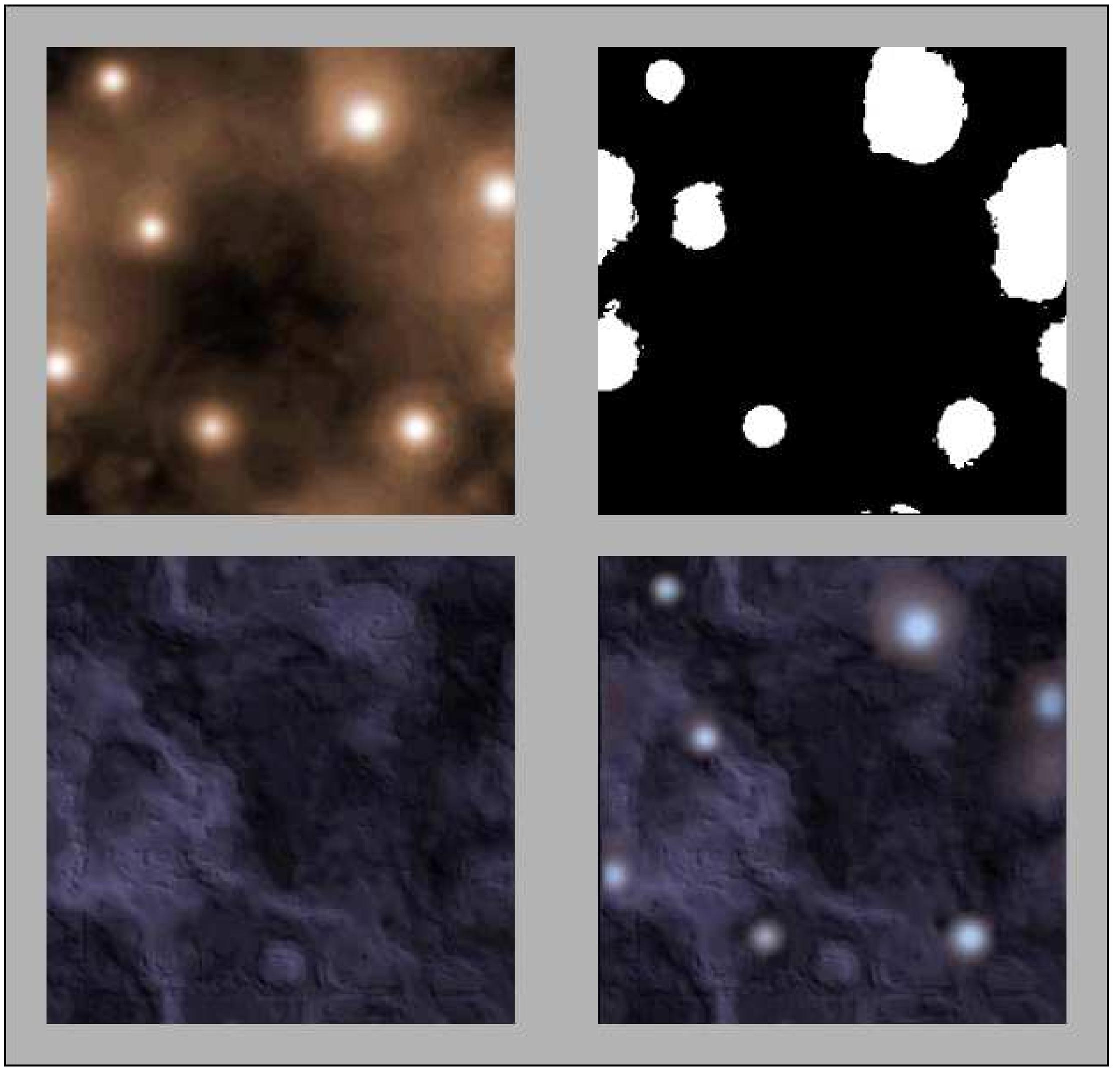


Figure 5.2: More blending results. Bright spots from the second texture are applied to the first.

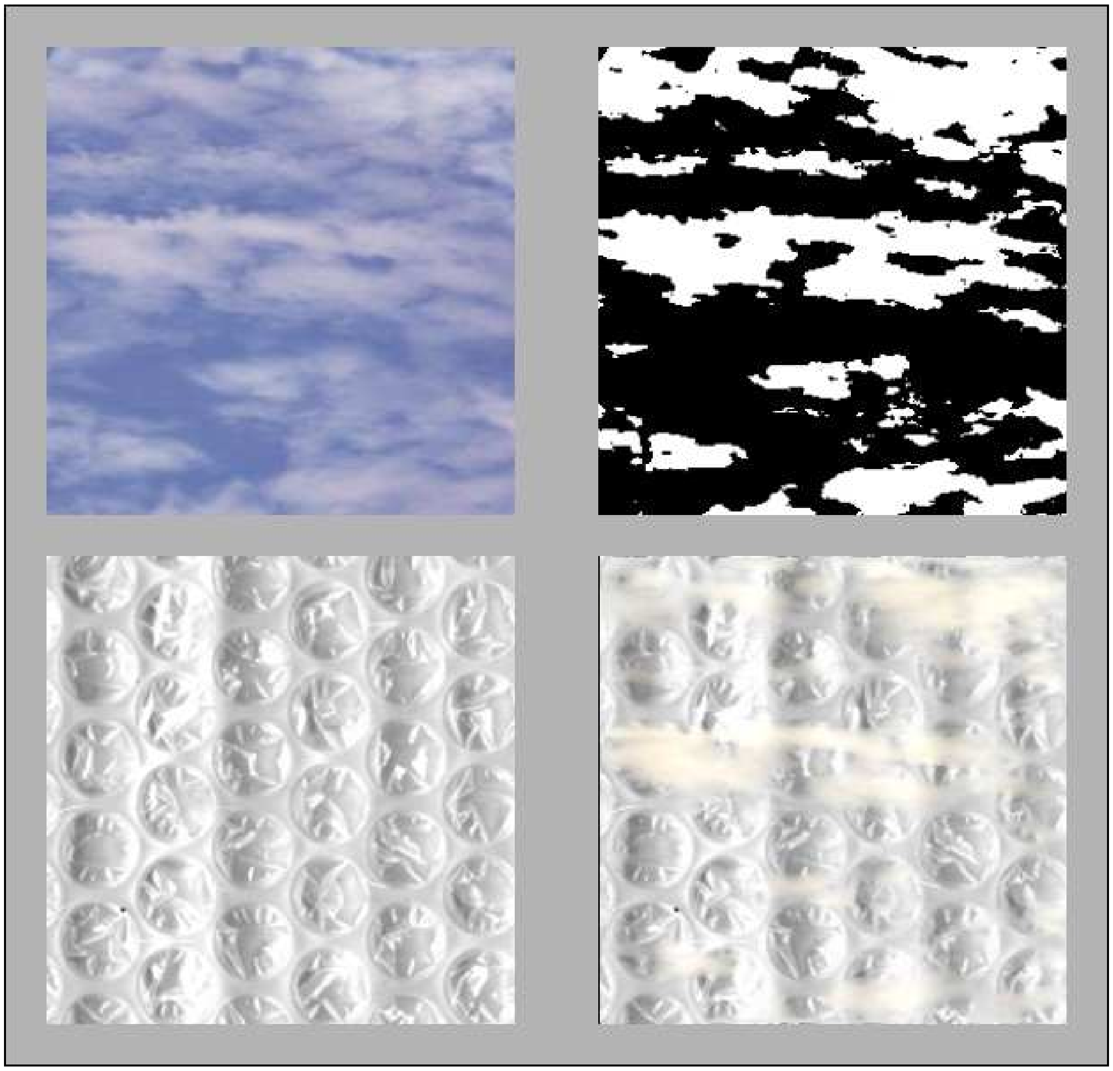


Figure 5.3: Lighter areas from the cloud texture are combined with the photograph of bubble wrap, giving the impression of folds.

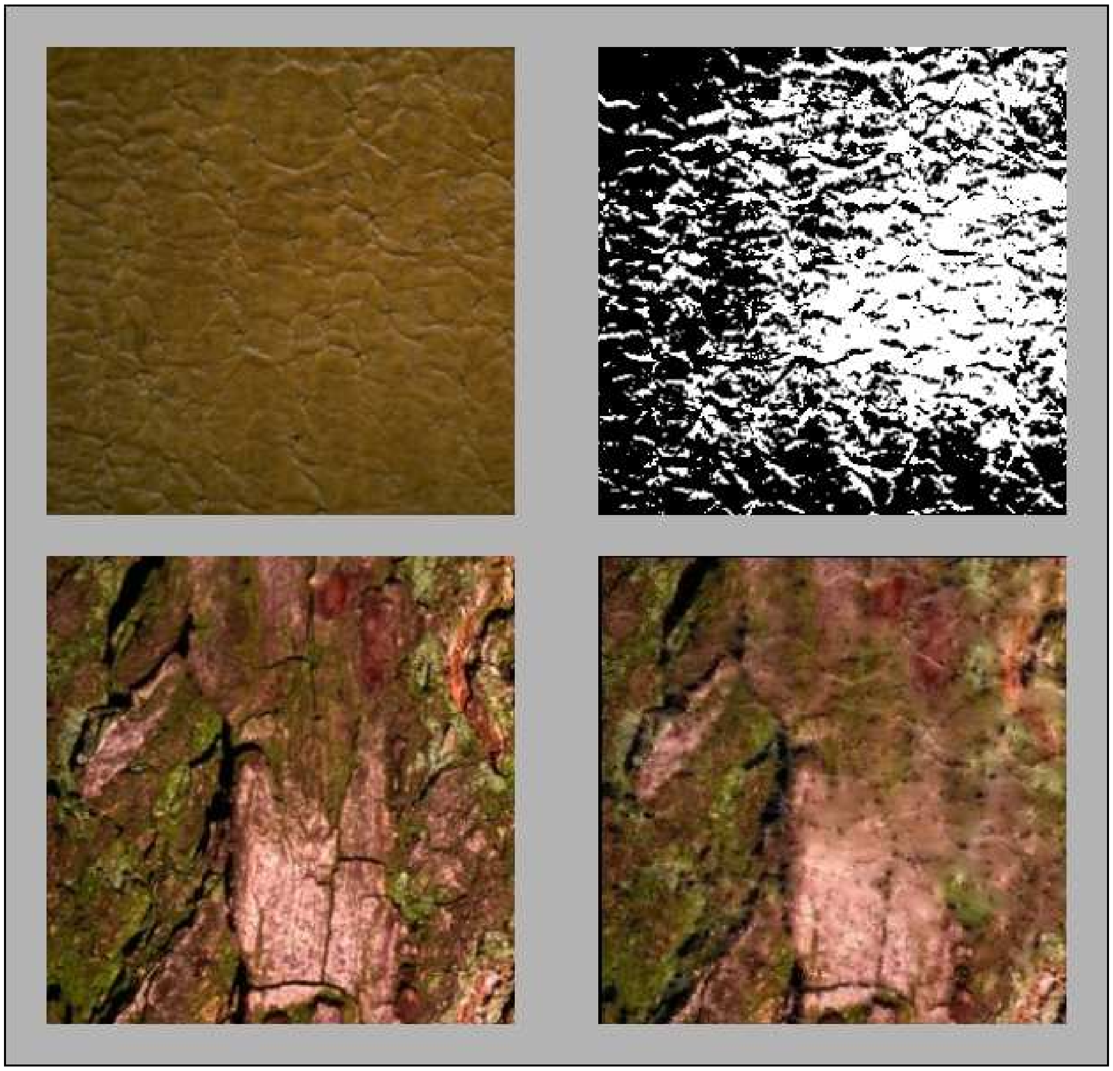


Figure 5.4: Tree bark is combined with ”elephant skin”, resulting in a slightly glossy looking texture.

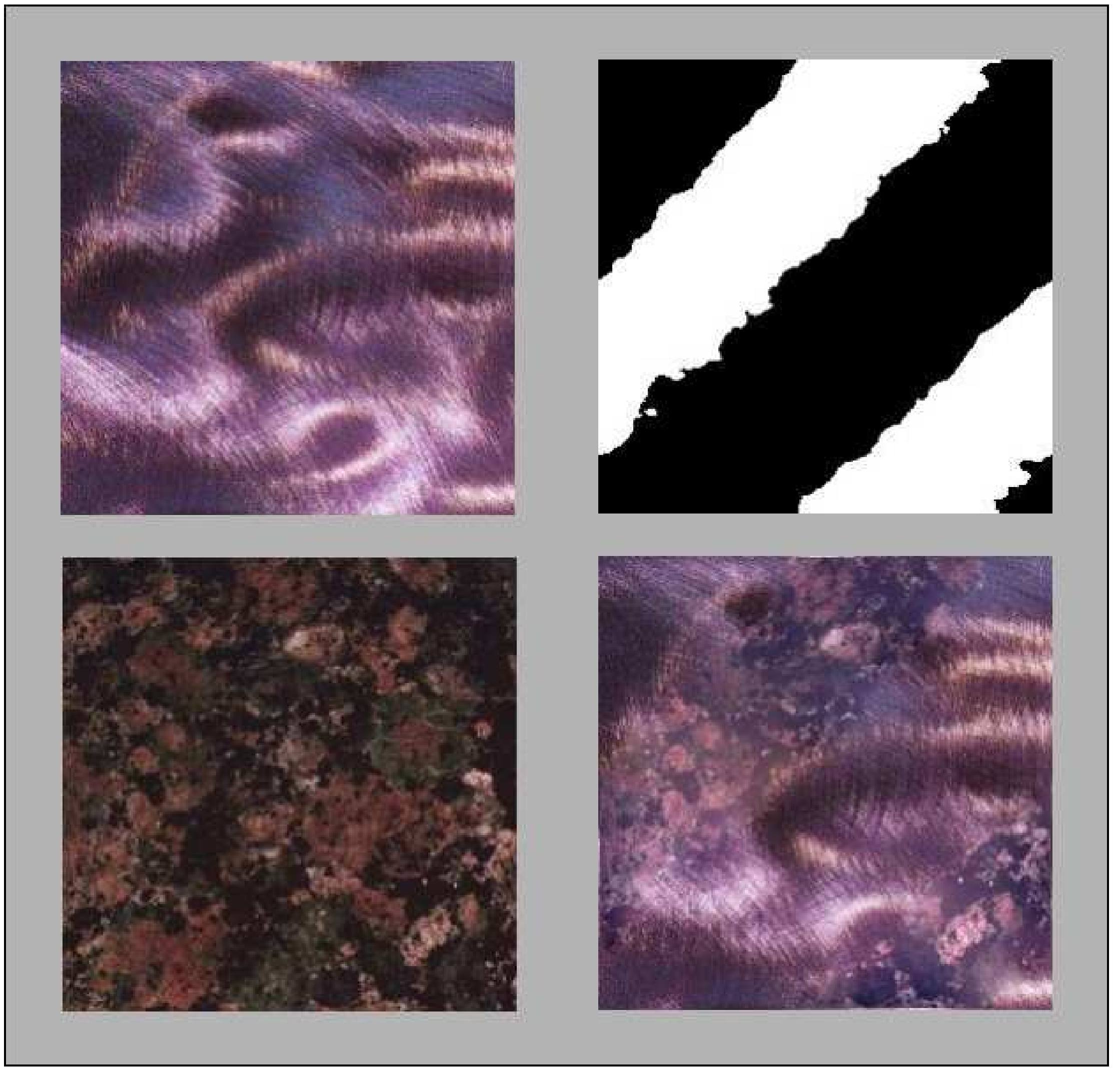


Figure 5.5: A marble texture is combined with a brushed metal texture. The mask used was taken from a third image.

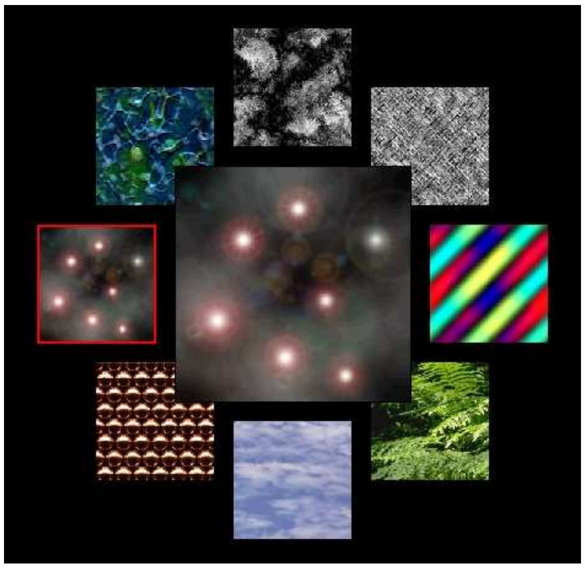


Figure 5.6: A selection is made from a random group of textures.

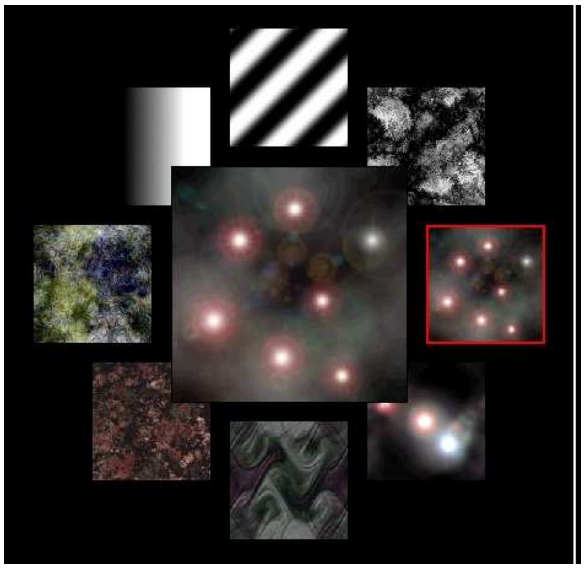


Figure 5.7: This group of textures was selected based only on colour profile. The reds present in the stars of the original likely contributed to the selection of the marble texture. The Fourier transforms vary considerably within this texture set.

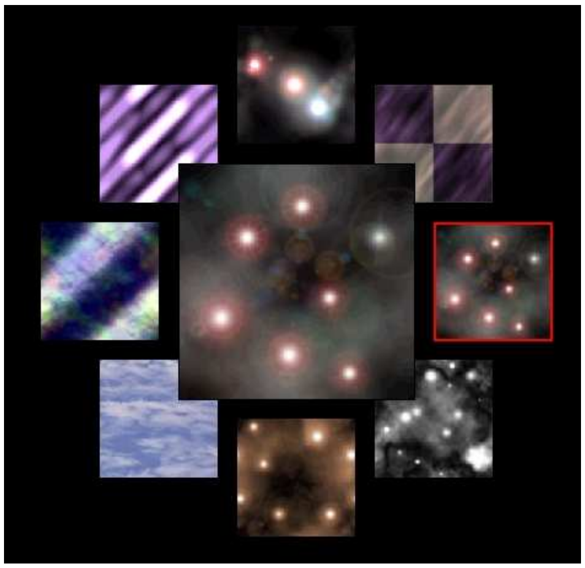


Figure 5.8: This time, only the Fourier comparison is used to provide matches. Similar textures are presented based on their frequency characteristics. Different star textures are presented since they have similar Fourier transforms. The stripe textures are also included because the major features of the original texture were roughly organized along a diagonal axis ($y = x$ or $y = -x$). Some textures, such as the clouds in the lower right, have a similar structure to the original but very different colours. Only intensity or greyscale values are used when computing Fourier transforms.

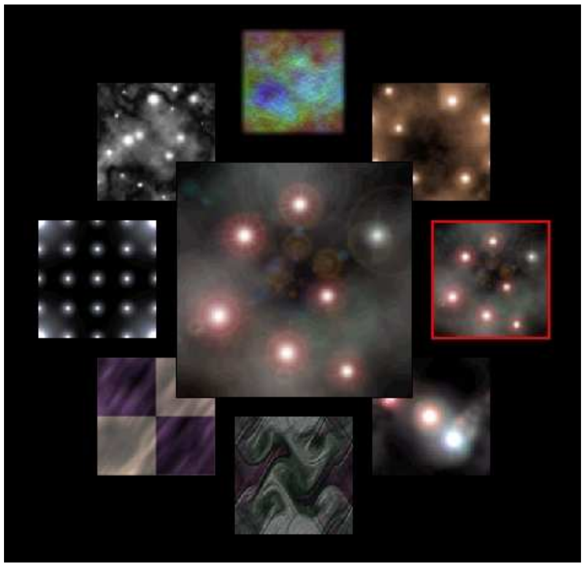


Figure 5.9: This final set of textures was selected using both colour and Fourier information, and contains results that appear to be closer than either of the other two measures. Outlying textures such as the clouds, which have a very different colour profile, have been eliminated.

Chapter 6

Future Work

Texture blending and image manipulation and classification are very wide domains and there are a large number of alterations that could be made to this texture blending project. These changes could affect the way the textures are blended, the interface, features, and options made available to the user, and the methods used to compare textures and masks in order to generate a set of suggested inputs for blending.

One obvious extension would be to allow the user to select a non-binary mask to be used to combine two textures. A greyscale mask could define the degree to which different areas of the two input textures would be blended. A full colour mask equivalent to a regular image could also be used to extract different colours from each of the two images. For example, a red mask might fully extract the red pixel data from the first image while taking blue and green data from the second. This could generate textures with different elements from each input image over the entire texture area. A greyscale mask would similarly allow for more gradual transitions between blended textures. A white to black gradient would, for example, gradually begin by drawing only on the first texture and then transition to the second image as pixel Y values increased.

The method used to select candidate textures is fairly involved and ways of comparing images have been heavily studied both for image processing and for computer vision applications. There are many alternatives to the Fourier transform for extracting structural data from an image, such as Steerable or Gabor wavelets.

Fourier data itself can be used in many ways. In [15], Fourier transforms were applied to small portions of an image. It is also possible to consider the phase and magnitude of the Fourier data rather than real and imaginary components. It is thought that the magnitude contains most of the information that would be relevant to comparing textures, so it would be possible to consider only the magnitude rather than both real and imaginary components. Making this change might both increase the accuracy of comparison and reduce the overhead required, since half as many values would need to be compared when calculating the difference between Fourier transforms using the current algorithm implemented.

Another addition that could help to increase the accuracy of image comparisons would be to attempt to estimate the orientation of features in candidate textures, as described in [2]. This estimation is done by calculating the total energy of the Fourier spectrum in each direction, beginning at the origin (the centre of the DFT array) and moving outward at different angles using polar coordinates. The greatest total energy and the associated angle for which it was obtained could be stored as two scalar values that could easily be compared between textures, and the actual calculation could be done in a constant number of steps depending on the step size. The drawback of this additional measure is that it would be unhelpful for classifying textures that either contain elements with no orientation or contain elements that are all oriented differently. Certain textures, however, are highly directional. One way of dealing with this problem would be to allow the user to toggle this feature on or off, as they can toggle the Fourier and colour comparisons. It would also be possible to use statistical methods to indicate the distribution of total energy values for different angles; the total energy value is likely meaningful for textures where it is significantly higher in one direction and less meaningful when it is roughly the same in every direction.

One major advantage of the Fourier orientation measure is that it consists of only two values that can be compared very quickly. Calculating the angle of maximum total energy takes thousands of calculations but these can be performed when a texture is first loaded into the library. For all subsequent uses, only the orientation and energy value must be compared. This is much faster than the current colour and Fourier comparisons which involve hundreds of values even after the histograms and transforms are precomputed. This is highly significant since distance calculations must be done in real time and since the total number of distance calculations scales up with the size of the texture database. Current methods for calculating texture difference effectively limit the size of database to hundreds of textures due to time constraints. If distance calculations were faster the libraries could effectively become much larger, although pre-computation would still take a significant amount of time (currently it is in the range of hundredths of a second per texture).

Beyond the orientation comparison, other approaches to comparing the Fourier data could be implemented such as the adaptive binning process found in [7]. This same adaptive binning approach could be used to generate colour histograms that vary based on the distribution of colours found in an image, as discussed in [13]. Furthermore, luminance could be used as a measure of image brightness when calculating image Fourier transformations [9]. Luminance values more closely match what humans perceive as brighter colours and would allow visible features of images to be extracted more accurately.

Another important piece of work that was not carried out as a part of this project was an attempt to formally evaluate how well the system chooses similar textures. This could be done by having human observers attempt to order textures from a small set based on a few different features that are also used by the blending tool or based on texture categories. The category approach was used to evaluate the systems described in Section 3 but the images being recognized generally depicted single real-world objects, not abstract patterns or surfaces. The texture blender already has checkboxes that can be used to turn colour or Fourier comparisons on or off, making it easy to isolate each individual measure to see whether or not it has an impact.

Masks are yet another part of the texture blender that could be extended significantly. Currently, distances between masks are calculated in the same way that distances between textures are calculated and masks are derived from textures simply by applying a certain cutoff and then rounding RGB values to $[0, 0, 0]$ or $[255, 255, 255]$. Different ways could be devised to generate masks based on other filters applied to existing textures or totally independent processes. It would also be conceivable to implement greyscale or even colour masks that affect the weighting of components taken from the two source textures rather than simply defining which image will be the source for which region. Currently, this is not supported by the blending algorithm.

Chapter 7

Conclusions

Realistic looking textures can be generated by combining existing images. This is accomplished by applying an algorithm that derives a resultant texture from two input textures. A mask is used to determine which regions are taken from which of the two sources. At that point, Poisson interpolation is applied to create smooth colour gradients within the image and to ensure that the output image can be tiled. This approach is much more realistic than simpler methods such as direct layering, which generates jagged looking edges, or averaging, which creates a washed out look in the final output. Other more complicated methods of mixing textures have been studied but they create very different results and have limitations not present when using the Poisson interpolation method.

In order to evaluate this algorithm, a tool was created that allows users to select textures and masks to blend. In addition to simply applying the blending algorithm, the tool can also be used to find and display sets of images and masks that can be used as input. This is done by analyzing the content of the images and masks and deriving a measure of similarity from a number of key features. In the case of the images, a colour profile is created. For masks, only the intensities in the image are considered. In addition to that direct colour and intensity information, Fourier data is used to compare the changes in colour and intensity over the surfaces of the images. This data is used to compute distances between potential textures and masks. The closest matches are offered in the result, allowing users to fine-tune their result. They can also iteratively ask for suggestions, effectively walking through texture space. An additional feature allows users to introduce some degree of randomness into the selection process, enabling users to take larger steps between less similar textures as they are trying out different textures and masks.

Ultimately, this texture blending tool generates convincing output images (see Figures 5.1 to 5.4) that appear natural and could be used to model real world or imaginary surfaces. The texture selection feature of the tool also allows users to quickly create something close to a desired output simply by carefully choosing from amongst a large database of existing textures (see Figures 5.6 to 5.9). However, many possible extensions and alterations could be made to the system to improve performance, either by making the texture blending itself more flexible or by improving the accuracy and speed of the texture classification system.

Bibliography

[1] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):120–135, Apr-Jun 2001.

[2] D. V. S. Chandra. Target orientation estimation using Fourier energy spectrum. *IEEE Transactions on Aerospace and Electronic Systems*, 34(3):1009–1012, Jul 1998.

[3] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. Image transforms. Retrieved March 15, 2007, from http://homepages.inf.ed.ac.uk/rbf/HIPR2/tranops.htm, 2003.

[4] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. Spatial filters - Laplacian. Retrieved March 15, 2007, from http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm, 2003.

[5] N.R. Howe. Percentile blobs for image similarity. *IEEE Workshop on Content - Based Access of Image and Video Libraries*, 2:78, 1998.

[6] L. Jia and L. Kitchen. Object-based image similarity computation using inductive learning of contour-segment relations. *IEEE Transactions on Image Processing*, 9(1):80–87, Jan 2000.

[7] F. S. Lim and W. K. Leow. Adaptive histograms and dissimilarity measure for texture retrieval and classification. *International Conference on Image Processing*, 2:825–828, 2002.

[8] W. Matusik, M. Zwicker, and F. Durand. Texture design using a simplicial complex of morphable textures. *Proceedings of ACM SIGGRAPH 2005*, 24(3):787–794, Jul 2005.

[9] S. McHugh. Histograms tutorial - luminance and color. Retrieved April 18, 2007, from http://www.cambridgeincolour.com/tutorials/histograms2.htm, 2007.

[10] G. Scott Owen. Hypergraph: Surface mapping. From the ACM SIGGRAPH Education Committee. Retrieved March 26, 2007, from http://www.siggraph.org/education/materials/HyperGraph/mapping/surface0.htm, 1999.

[11] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *International Conference on Computer Graphics and Interactive Techniques*, pages 313–318, Jul 2003. ACM SIGGRAPH 2003 Papers.

[12] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge (UK) and New York, 2nd edition, 1992.

[13] F. Tauheed. Photo collage and image similarity quantification. *Student Conference On Engineering, Sciences and Technology*, pages 127–131, Dec 2004.

[14] E. W. Weisstein. Partial differential equation. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/PartialDifferentialEquation.html, 2004.

[15] F. Zhou, J. F. Feng, and Q. Y. Shi. Texture feature based on local fourier transform. *International Conference on Image Processing, 2001*, 2:610–613, Oct 2001.