APPROVAL SHEET

Title of Thesis: Perceptually Oriented Patch Based Texture Synthesis

Name of Candidate: Patrick Gillespie
Master of Science, 2006

Thesis and Abstract Approved: _____

Dr Marc Olano
Assistant Professor
Department of Computer Science and Electrical Engineering

Date Approved: _____

Curriculum Vitae

Name: Patrick Gillespie.

Permanent Address: 5118 Meadow Creek Terrace, Ellicott City, Maryland 21043.

Degree and date to be conferred: M.S., 2006.

Date of Birth: April 3, 1982.

Place of Birth: Las Cruces, New Mexico, USA.

Secondary Education:
Howard High School, Ellicott City, Maryland, USA, May 2000.

Collegiate Institutions Attended:
September 2004 – August 2006 University of Maryland Baltimore County M.S., August 2006.

September 2000 – May 2004 University of Maryland Baltimore County B.S., Magna Cum Laude, May 2004.

Major: Computer Science.

Professional Positions Held:
Teaching Assistant. CSEE Department, UMBC. (August 2004 - June 2006).
Intern. Northrop Grumman. Baltimore, MD (June-August 2005)
Intern. Northrop Grumman. Baltimore, MD (June-August 2004)
Intern. Northrop Grumman. Baltimore, MD (June-August 2003)

ABSTRACT

Title of Thesis: Perceptually Oriented Patch Based Texture Synthesis

Patrick Gillespie, Master of Science, 2006

Thesis Advisor: Dr. Marc Olano

Patch-based texture synthesis methods work by taking patches of texture and finding the best way to "stitch" them together so that the seam between the two patches is hardly noticeable. The color difference along a cut between the two patches is usually used as the metric for finding the best seam between the images and traditionally red, green, and blue values are used in determining the color difference.

This works well for randomized textures, but can leave artifacts in textures that include definite objects within the patches. We have made use of knowledge of the human visual system to find better seams between patches. We have used perceptually uniform color spaces, edge detection, an alternative graph structure, and the contrast sensitivity function as factors in graph cuts in order to find better seams between images. A user study showed that the new metric significantly improved seam invisibility in images that were spliced together.

Perceptually Oriented Patch Based Texture Synthesis

by
Patrick Gillespie

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
2006

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Marc Olano, for his guidance, help, and advice along the way to finishing this thesis. I would like to thank Dr. Penny Rheingans for helping me to conceive the user study and for looking at some of my images and giving me her advice. I would also like to thank Dr. Laura Stapleton for answering my emails about how to analyze my user study results. Lastly, I'd like to thank each of the professors who agreed to serve on my committee, Dr. Marc Olano, Dr. Penny Rheingans, and Dr. Richard Chang.

TABLE OF CONTENTS

List of Figures

1 INTRODUCTION

1.1 MOTIVATION

In some situations it is necessary to have large amounts of natural looking texture, for example, the textures that represent the ground and walls in 3D simulators. Such textures could be represented by smaller textures that repeat, however, this leads to an unnatural repeating pattern on the object the texture is applied to. To solve this problem one could simply use very large textures; however, this could lead to memory issues and one does not always have an infinite amount of memory at their disposal. Therefore, a method is needed to create large amounts of natural looking texture that is memory efficient.

1.2 TEXTURE SYNTHESIS

Texture Synthesis is a way of taking a small sample of a texture and generating more of it in an image that is usually larger than the original sample and that looks like a natural extension of the original texture. For example, given a small image of some olives, a texture synthesis algorithm should be able to create an infinite amount of texture that looks like this. Figure 1 demonstrates an example of this. Image A is the input and larger images B and C are generated outputs. Image B, the tiling of the sample image, could be considered a simple form of texture synthesis, however, this leads to a noticeable repeated pattern, which is unnatural and something texture synthesis algorithms strive to avoid. Image C shows the output of a graph cut texture synthesis algorithm.

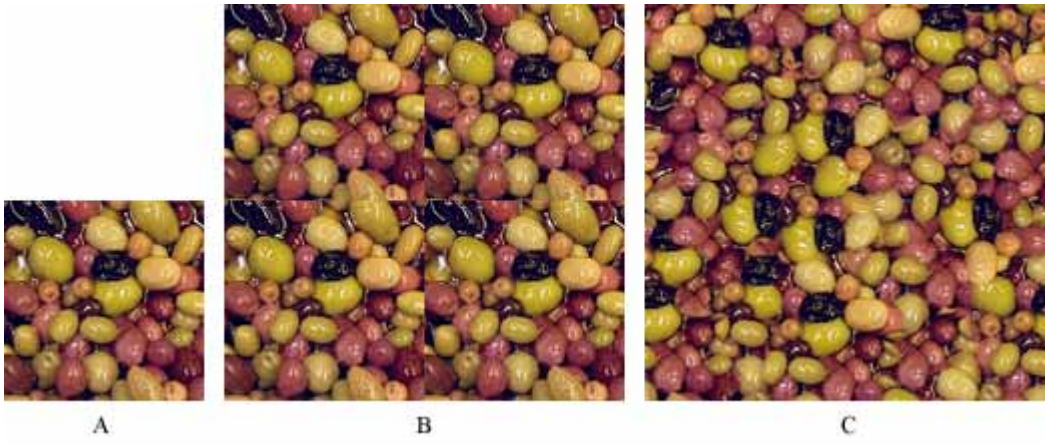

Figure 1: Texture Synthesis Example. Samples B and C are generated using sample A. B is created by tiling the image sample while C is created through graph cut texture synthesis.

Texture synthesis algorithms can be roughly divided into pixel based and patch based algorithms. In pixel based algorithms, an output texture is constructed pixel by pixel. This usually works by using feature matching to determine the color of the next pixel or by modeling the texture as a Markov Random Field and creating the texture by probability sampling [Efros and Leung, 1999; Heeger and Bergen, 1995; Wei and Levoy, 2000]. These algorithms are generally slow and do not work well for textures with noticeable structure in them, such as brick walls. In patch based algorithms, an output texture is constructed by placing patches of the input texture into the output texture and then blending it so that the new patch seamlessly fits in as well as possible. These algorithms have the benefits of being quick and good at constructing any type texture. Most recent texture synthesis algorithms are patch based.

Xu et al. [2000] presented the first patch based texture synthesis algorithm, known as the Chaos Mosaic. This algorithm works by randomly placing patches of texture and blending the edges to avoid artifacts. This method works well for stochastic textures, but

fails when inputs have noticeable structures in them. Ashikhmin [2001] created a pixel based algorithm which encouraged verbatim copying of patches from the input sample. This algorithm expanded on an earlier pixel based method put forward by Wei and Levoy [2000]. Though this method was very fast, it had the drawback of not working for all types of inputs. It narrowed its class of workable textures to those that were quasi-repeating and naturally occurring, such as grass or pebbles. Other attempts have been made to bridge the gap between pixel based and patch based texture synthesis, such as Hybrid Texture Synthesis, put forward by Nealen and Alexa [2003]. This method uses patches to maintain the global structure of the texture while using pixel based techniques to avoid artifacts along seam edges. This algorithm produces good results, but at the cost of speed.

Later patch based texture synthesis algorithms, such as Image Quilting [Efros and Freeman, 2001], Wang Tiles [Cohen et al., 2003] and Graph Cut Texture Synthesis [Kwatra et al., 2003], have relied on intelligently placing patches of texture and finding the best seam between them. In Image Quilting, an output image is created in raster scan order by having new patches overlap old ones slightly, and then using a dynamic programming algorithm to figure out the best seam between them. Wang Tiles generates a set of tiles from the input where each tile's edge will match up with at least two other tiles from the set. These tiles are then used to tile the output texture. Tile creation is done using a novel method of stitching certain patches of input texture together.

In Graph Cut Texture Synthesis a texture patch is placed somewhere on the existing output texture, pixels in unfilled areas of the output texture are copied over automatically, and a cut between the patch and the existing texture determines which pixels from the overlapping region are copied over onto the output texture. Figure 2 gives a visual description. The image on the far left shows the initial output texture. The image in the middle shows a second patch being placed onto the output texture. The discolored area represents the area where the two patches overlap, a cut will need to be made to see which pixels from which patches go in the final output. The image on the far right shows what the texture looks like after a cut has been made. The seam is left in this image only to show where the cut was made.

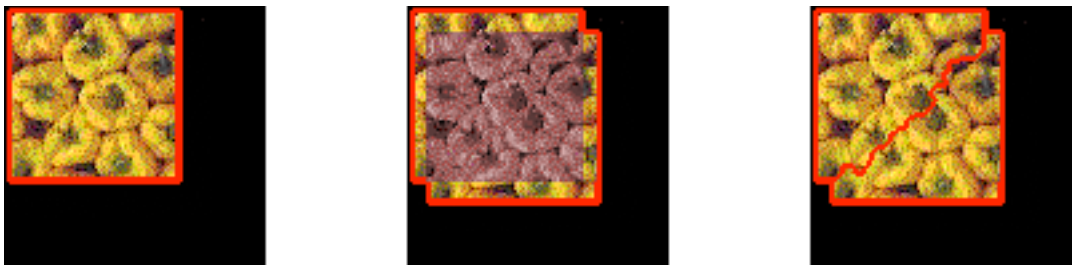

Figure 2: An example showing how a cut is determined when a new input sample is placed onto the output image.

The Graph Cut Texture Synthesis algorithm uses three different techniques in order to find the best place to put a new patch. The first method is to randomly place a new patch anywhere. This leads to fast output results, but can cause the output to suffer in quality. The second method is an exhaustive search of the output texture using the Sum of Squared Differences (SSD) metric between pixels in order to find the optimal spot. This works quite well but is time consuming. A technique using Fast Fourier Transforms has been developed to speed up this process [Kilthau et al. 2002; Soler et al. 2002]. The final

method is to look at unfilled regions in the output texture and to find the best way to place a patch over it.

Image Quilting and Wang Tiles uses a dynamic programming algorithm to find the best seam while Graph Cut Texture Synthesis sets up a graph of nodes representing the overlapping region where solving its Min Cut [Ford and Fulkerson, 1963] is equivalent to finding the optimal seam. Either method of seam finding could be used in each of these algorithms' creation process. However, the graph cut method is applicable in any dimension and can incorporate old seams into its calculation of the best cut. Graph Cut Texture Synthesis is widely considered the best method for texture synthesis and its method of finding the best seam will be the focus of this study.

1.3 CONTRIBUTION OF THESIS

- We will use knowledge of the human visual system and a new graph set up to develop a new graph cut metric for seamlessly stitching images together.
- We will carry out a user study to determine the benefits of the new metric over a generic RGB metric.

1.4 ORGANIZATION OF THESIS

Chapter 1 has introduced texture synthesis, explained why it is important, described the different algorithms behind it, and stated the contribution of the thesis. Chapter 2 will survey the related work on graph cut texture synthesis. Chapter 3 will look at work that has gone into understanding the human visual system. Chapter 4 will explain the methods used to incorporate knowledge of the human visual system into graph cut texture

synthesis. Chapter 5 will show the results. Chapter 6 will show the results of a user study. Chapter 7 concludes the thesis and discusses possible future work.

2 RELATED WORK ON SEAM FINDING

2.1 GRAPH CUTS

In Image Quilting, a dynamic programming algorithm is used to find the best seam from one side of the overlap region to the other. This seam determines how each patch contributes to the overlapping region. The path it finds is where the colors are the most similar between patches. Figure 3 demonstrates this by showing two patches of texture that have an overlapping region, in this region a cut is found that stitches them together.

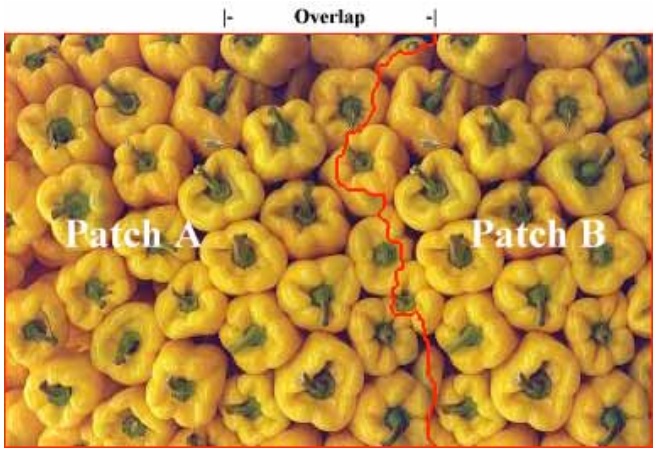

Figure 3: Two overlapping patches, the best cut is found between them using a dynamic programming algorithm.

Graph Cut Texture Synthesis casts this into a graph cut problem by having each pixel in the overlapping region represent a node in an undirected graph. The graph is constructed such that there are edges between adjacent nodes and these edges represent the color differences between the old and new patches. Let A and B represent patches, and let $A(x)$ and $B(x)$ represent pixel colors at position x in those patches. The matching cost M between adjacent pixels s and t is then described as:

$$M(s,t,A,B) = \|A(s) - B(s)\| + \|A(t) - B(t)\|$$

Where $\|x\|$ is an appropriate norm, such as an absolute value, that is to be chosen by the programmer. Source and sink nodes are also created. These are nodes that represent an anchor for the two different patches. Nodes on the boundary to the output texture are linked back to the source node, while nodes that border empty regions in the output are linked to the sink. If no such nodes exist, as is the case if a patch is placed over a region of pixels that is filled with texture, then the programmer must choose a method for determining how the nodes link back to the sink. Constraining at least a single pixel from the new texture is suggested as a possible option. Once this graph representing the overlapping region is created, the min cut / max flow problem [Ford and Fulkerson, 1963] is solved for it to determine where the cut between the two patches should be made. Figure 4 shows an example graph representing two patches of texture, *A* and *B*. The nodes between them represent the area where the patches overlap and the red line indicates where the cut is made.

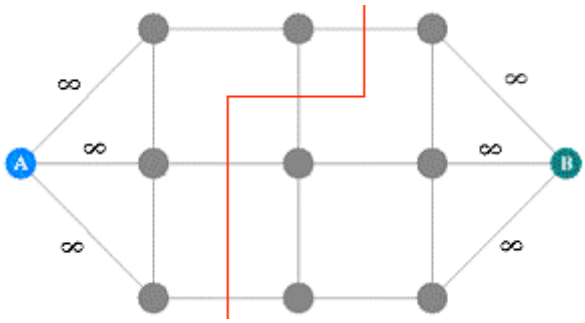

Figure 4: Graph Cut Example

2.2 SETTING UP THE GRAPH

Kwatra et al. [2003] proposed in the original Graph Cut Texture Synthesis paper that simple color differences be used in determining the weights for edges in the graph. In addition to the simple color difference metric, Kwatra et al. also proposed a more

advanced cost function for setting edge weights that factored in the frequency content of the texture patches. Another perceptual variable they sought to factor in was that of old seams. They accounted for this by adding in "seam nodes" to the graph along the locations where old seams existed. These "seam nodes" then linked to the nodes on both sides of the seam and had an extra edge arc back to the incoming patch. All of the edges protruding from a seam node had the same value, the color difference that existed between the two nodes on either side of the old seam. This technique worked well for covering potentially visible seams in the output texture; Figure 5 shows an example where seam nodes are created along an old seam and linked back to the source node so that the algorithm takes them into account when finding the best cut between the patches.

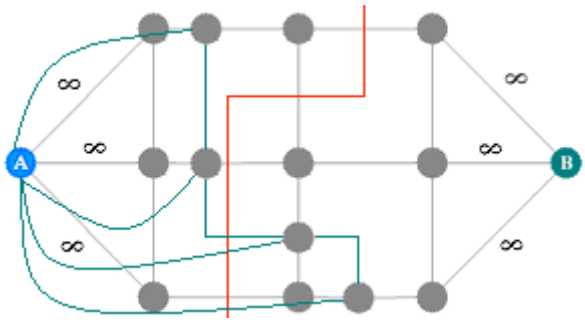

Figure 5: Graph cut example with an old seam taken into account. The green lines indicated what is added in by the old seam. The red line is the new cut.

Agrawala et al., in their framework for combining multiple photographs to create the optimal looking image [2004], introduce another cost function for determining edge weights that takes many factors into account such as color gradients, edges, and RGB color difference. No comparisons are given as to how this function compares the simpler versions. This function also has an inertia component that is specific to the creation of a photomontage and would not make sense in texture synthesis.

3 PERCEPTUAL ISSUES

Most work in texture synthesis has focused on finding new methods for creating large amounts of output textures very quickly. Accuracy is also an important part of the equation, and is dealt with as so, but perceptual techniques are not usually taken into account. For pixel based algorithms, most of the attention to accuracy is based on finding neighborhoods of pixels in the input image similar to the neighborhood one is working on creating in the output image. Other than using uniform color spaces, perceptual techniques do not pose any obvious advantages for pixel based methods.

For most patch based methods, specifically Graph Cut Texture Synthesis, a seam is found between the incoming patch and the rest of the output texture for determining what gets copied over. Since we are dealing with patches of texture that may contain noticeable structures, making use of the human visual system can allow one to find a seam that is much less noticeable between the patch and output texture.

3.1 COLOR SPACES

The RGB color space combines red, blue and green light to create the different colors a user sees. Though it is simple to use and implement on hardware, it has many drawbacks [Hall, 1989]. One of these drawbacks is that it is a device dependent model, meaning RGB colors rendered on one device might not necessarily be the same as RGB colors rendered on a different device [Johnson and Fairchild, 1999]. Another drawback is that it is not a uniform color space. Color differences between different sets of colors that are mathematically the same distance apart in the color space may not be perceived as having the same difference [Hill et al., 1997].

In 1931, the International Commission on Illumination (CIE) created the CIE XYZ color space. This color space was based on experiments done by William David Wright and John Guild that had participants visually analyze colors and adjust them so that they appeared the same. Wright measured the vision of ten subjects in this manner and published his results in 1929; while Guild measured the vision of seven subjects and published his results in 1931. The CIE XYZ color space that resulted from these studies was special in that it was device independent and based on human observations. The X, Y, and Z values correspond approximately to red, green and blue light.

Though the CIE XYZ color space provided an accurate device independent way of specifying a color, it did not provide an accurate way to calculate color differences. The mathematical distance and perceptual distance between CIE XYZ colors could have a

discrepancy of up to 8,000%. This meant that the non-uniformity of the color space was approximately 80:1 [Reddy, 1996].

Providing a way to accurately measure perceptual distances in a color space was first put forward in 1942 by Douglas L. MacAdam. MacAdam constructed an experiment in which a trained observer was shown two colors, one was fixed and while the other was adjusted by the observer. The observer was asked to adjust this color until it matched the fixed color. MacAdam had the observer do this for 25 different colors [Wyszecki and Stiles, 1982].

Since the human eye is not ideal, the observer did not always correctly in match the two colors. However, as one can see in Figure 6, all of the results that came out of this experiment fell into elliptical regions surrounding the fixed colors on the CIE chromaticity diagram [Wyszecki and Stiles, 1982]. Each ellipse varied in size depending on the test color. In the diagram, the ellipses appear as 10 times their actual size in order to show variety in sizes and orientations that occurred.

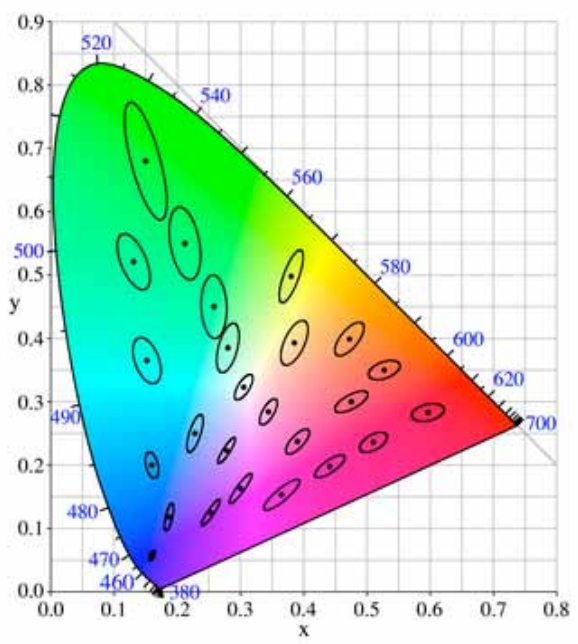

Figure 6: CIE 1931 (x,y)-chromaticity diagram with MacAdam Ellipses plotted. Each ellipse appears as 10 times its actual size [Wikipedia, 2006].

These ellipses represent regions of indistinguishable color from that of their center point. In the years following these results, several attempts were made at mathematically constructing MacAdam ellipses at certain target points. The most successful of these was done by Chickering in 1967 and 1971 [Wyszecki and Stiles, 1982].

In 1976, the CIE introduced two new color spaces that were more perceptually uniform than those that had come before, L*a*b* and L*u*v*. In L*a*b*, the L* component represents a luminance value while the a* and b* components represent red/green and yellow/blue chrominance, respectfully. In L*u*v*, the L* component also represents luminance while the u* and v* components represent chrominance.

These color spaces were built on the CIE XYZ color space and incorporate color metrics from the MacAdam ellipse study. The color spaces were a great improvement over the perceptual non-uniformity of the CIE XYZ color space, having a perceptual non-uniformity of around 6:1 [Reddy, 1996]. In addition to this, an improved color difference calculation for the L*a*b* color space was later developed. It was called the CMC Color Difference Formula. This formula was developed from information gathered from colorant industries and claimed to be more accurate for small color differences [Reddy, 1996]. Since then, several other formulas for color difference have been developed for the L*a*b* color space, such as CIE94 and CIEDE2000 [Sharma et al., 1995], which claim to improve upon past results. These developments in the color difference calculation have led L*a*b* to be the most popular uniform color space.

Several studies have used uniform and device-independent color spaces and have shown how they generally aid in producing better results [Moroney and Fairchild, 1995; Nischik and Forster, 1997; Serup and Agner, 1990; Takiwaki et al., 1994; Weatherall and Coombs, 1992]. Weatherall and Coombs [1992] used the L*a*b* color space to measure and analyze the skin colors of 81 different people. They found that it could be quite useful for identification purposes and speculated that it would be useful in cosmetics, genetics, and disease diagnosis. Moroney and Fairchild [1995] tested a variety of different color spaces on JPEG image compression and found they got the best results when they used the L*a*b* and L*u*v* color spaces.

3.2 CONTRAST SENSITIVITY FUNCTION (CSF)

Visual acuity is defined as one's ability to perceive fine details. One way to measure visual acuity is through a contrast grating, which is a set of smoothly alternating white and black bars. Two parameters govern how perceivable the bars are: the contrast between them and their spatial frequency. When these parameters are mapped onto a two dimensional plain a distinct curve forms between the areas that are distinguishable from each other and the ones that aren't. Figure 7 shows a visual representation of this. The y axis represents contrast sensitivity, which is simply the reciprocal of contrast. As y increases, the contrast decreases. The x axis represents spatial frequency. As x increases, so does the spatial frequency. The mathematical function representing this curve is known as the contrast sensitivity function (CSF).

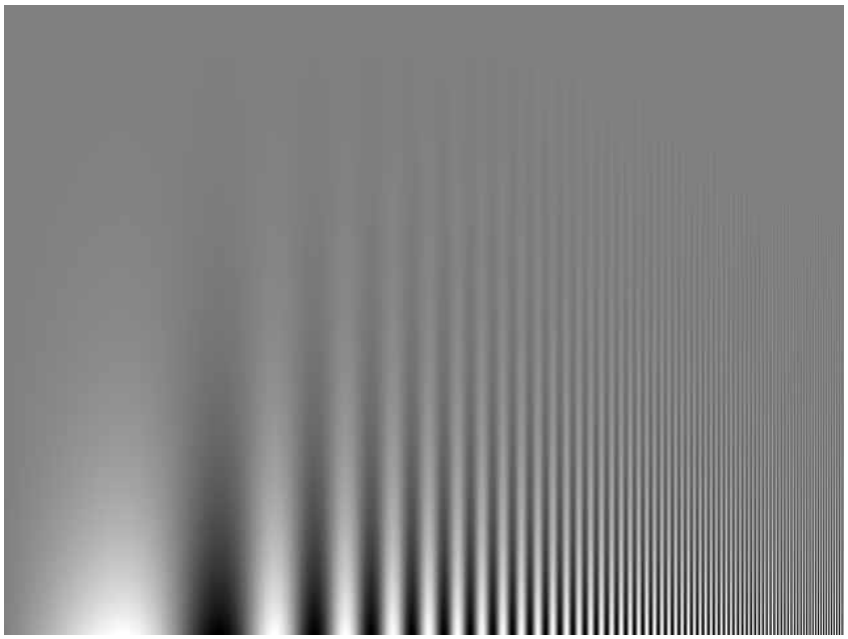

Figure 7: A visual representation of the Contrast Sensitivity Function [Ohzawa, 2006].

Since the CSF models how our visual system perceives changes in frequency and contrast, it has been used in a variety of areas within graphics to help enhance the results. Some examples include image compression [Banerjee and Evans, 2002], visual masking [Ferwerda et al., 1997], simplification of polygonal meshes [Williams et al., 2003] and artistic rendering [DeCarlo and Santella, 2002].

3.3 CONTRAST

Contrast describes how well an object sticks out from its background. There are two main ways of calculating contrast: Weber Contrast and Michelson Contrast. The Michelson Contrast is most commonly used in measuring contrast in repeating patterns while Weber Contrast is most commonly used to measure the contrast of an object that is up against a background of a uniform color [Peli, 1990]. The two contrast functions also have different domains, Michelson Contrast has a domain of [0, 1] and Weber Contrast has a domain of [-1, ∞]. For these reasons, Michelson Contrast is often the choice for calculating the contrast parameter in the CSF. Michelson Contrast is represented mathematically as

$$\frac{L_{\max} - L_{\min}}{L_{\max} + L_{\min}}$$

Here L_{max} is the maximum luminance over a given area and L_{min} is the minimum luminance over a given area. If your area is a single pixel, it was found through testing that using a scaled down version of that pixel's luminance value in the CSF was a reasonable substitute for contrast sensitivity.

3.4 SPATIAL FREQUENCY

Spatial frequency describes how often an object repeats itself over a given interval. It is usually measured in cycles per degree (c/deg), where a degree represents one degree in one's field of vision. In Figure 8, for example, if the contrast grating shown in part (A) took up one degree of one's vision field, then it would be measured as having 3.5 c/deg since it represents 3.5 cycles of black and white bars. The bars in (B) occur at a more frequent rate over the same amount of space and would thus be measured as having 7.5 c/deg.

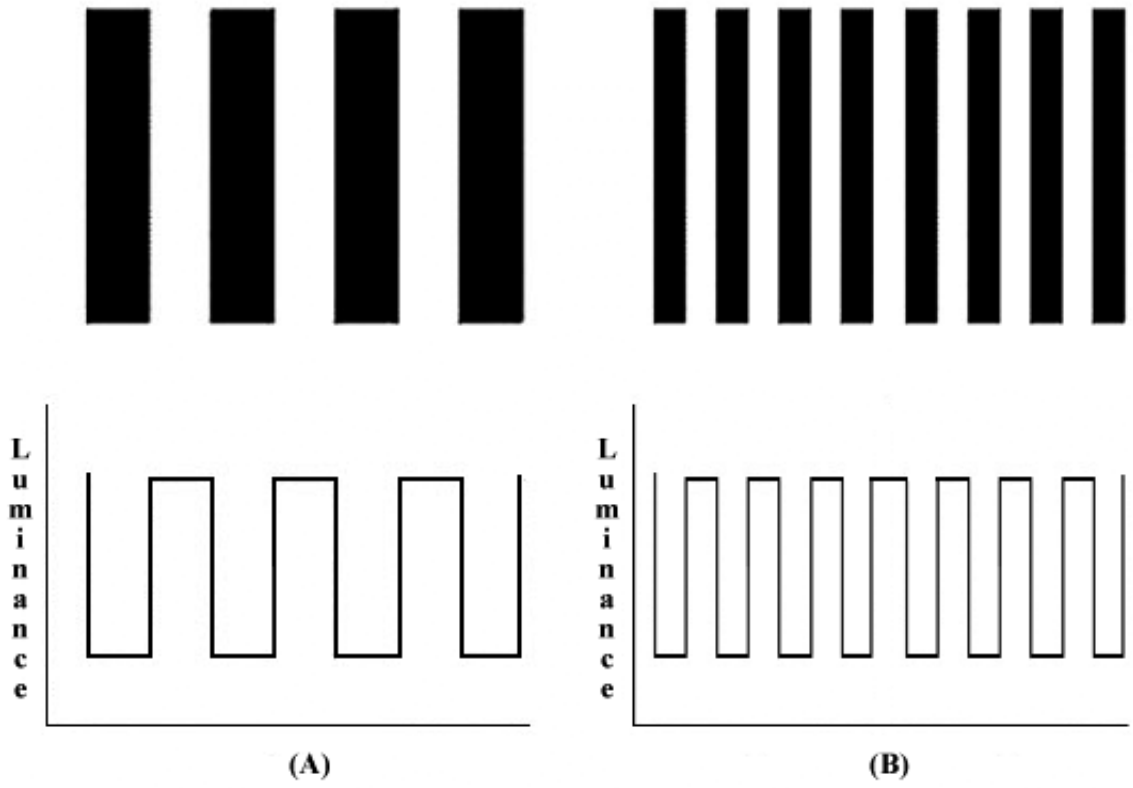

Figure 8: Two contrast gratings showing different spatial frequencies.

3.5 CALCULATING THE SPATAL FREQUENCY OF AN IMAGE

There are multiple ways one can extract spatial frequency information from an image. For this study, we used a method proposed by Reddy [1996] for calculating the c/deg parameter for the different locations on the images. Fourier Analysis was not used because it returned much more data than needed. Reddy argues that his method is advantageous to Fourier Analysis in that it is restricted to the "fundamental features" of the image, or the lowest frequencies, instead of being overloaded by every spatial frequency in the image. He argues that if these lower frequencies are not visible to the human observer, than the high frequencies will not be visible either. His method is also easier to interpret and apply. Reddy's method has three basic steps:

1. Find all of the visual features in an image.
2. Extract all of the relevant spatial frequencies from each feature in terms of cycles per pixel (c/pixel).
3. Scale the relative frequency values into units of c/deg.

For step 1, each pixel in the image is associated with a boolean value which indicates if it has been processed yet. A value of true means it has been processed, a value of false means it has not been processed. Once initiated, the algorithm attempts to grow each pixel into a feature by associating it with its adjacent pixels if they have a color value with a just noticeable difference (JND) of a certain amount. A JND is defined as the smallest amount something must be changed in order for it to be visible to a human observer. Reddy adopts a JND of 3 for the color difference in L*a*b* space, though he

states that this parameter can be tweaked and raised to as much as 10, which would represent a substantial difference. Each pixel can only be visited once. After every pixel has been processed, the image is successfully segmented into a collection of features.

During step 2, spatial frequency, in terms of c/pixel, is calculated for each feature. Since we are dealing with 2D objects, Reddy suggests measuring the feature from different orientations and then factoring this information together. To figure out the c/pixel for a particular orientation, the longest stretch of contiguous pixels which go from one side of the object to the other in that orientation is recorded. Remembering the contrast grating, we note that one bar is only half a cycle, therefore, our stretch of contiguous pixels only represents half a cycle. So if you have five contiguous pixels at a particular orientation, the cycles per pixel is going to be 1/2 * 1/5 = 1/10 c/pixel for that orientation. Reddy outlines a function he calls the Relative Spatial Frequency (RSF) function to describe this:

$$RSF(\theta) = 1/2 * \beta(\theta)$$

Here $\beta(\theta)$ represents the number of contiguous pixels at an orientation θ. One can also simplify this algorithm and get good results by combining steps 1 and 2. This is done by calculating the feature length relative to each pixel at each pixel. Therefore, one simply needs to visit each pixel once and measure the different spatial frequencies instead of finding all of the different features and then measuring their spatial frequencies. Combining the two steps tends to yield better results.

During step 3, all of the spatial frequencies that were found in terms of cycles per pixel are translated into cycles per degree. In order to do this, one first needs to know how much space the display device takes up in the user's field of view (FOV). This can be solved by knowing the size of the display device and the distance the user is from it. Once we have this information the size of the device in the FOV can be derived from the viewing geometry. This is shown in Figure 9 where the angle of the eye that the display takes up is calculated by a simple arctan function which is shown below.

$$FOV_{vertical} = 2 * \tan^{-1}\left(\frac{height * 0.5}{dist}\right)$$

$$FOV_{horizontal} = 2 * \tan^{-1}\left(\frac{width * 0.5}{dist}\right)$$

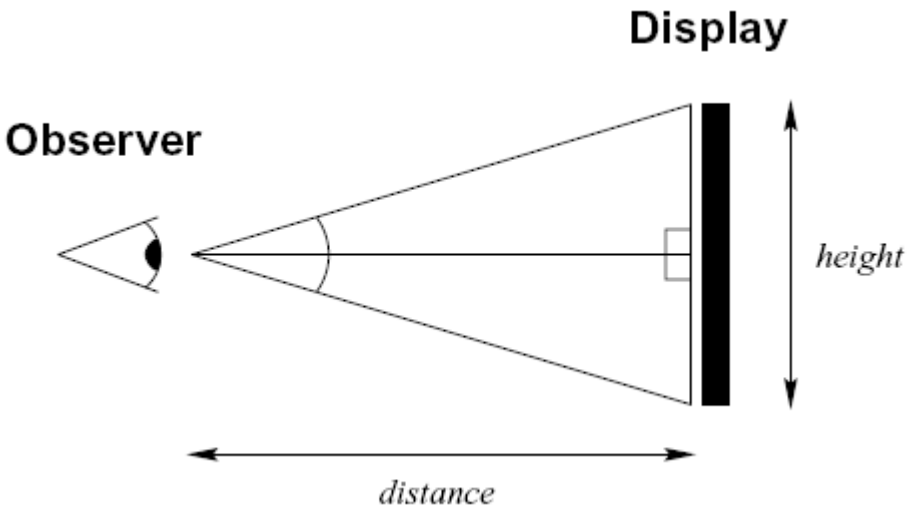

Figure 9: A user looking at a display device. Simple geometry can be used to determine how much space it takes up in his FOV [Reddy, 1996].

Next we need to compute scaling variables based on the screen's resolution:

$$S_{horizontal} = \frac{res\_width}{FOV_{horizontal}} \qquad S_{vertical} = \frac{res\_height}{FOV_{vertical}}$$

Here *res_width* and *res_height* are the screen's resolution in pixels. Finally, if we assume we are dealing strictly with just horizontal and vertical frequencies, we can recombine them into one absolute spatial frequency using the $SF(\theta)$ function:

$$SF(\theta) = \sqrt{(S_{horizontal} RSF_{horizontal})^2 + (S_{vertical} RSF_{vertical})^2}$$

This function uses Pythagoras' theorem to return the absolute spatial frequency for the feature. Ways of factoring in the spatial frequencies at other orientations are also given by Reddy.

3.6 EXAMPLE CALCULATION OF SPATIAL FREQUENCY

Let us assume we have a display screen that is roughly 33.3 cm in width and 20.8 cm in height. If a viewer is sitting 50 cm away from the screen, then his FOVs are:

$$FOV_{vertical} = 2 * \tan^{-1}\left(\frac{20.8 * 0.5}{50}\right) = 23.5$$

$$FOV_{horizontal} = 2 * \tan^{-1}\left(\frac{33.3 * 0.5}{50}\right) = 36.8355$$

Assuming a resolution of 1250x800, the scaling variables are:

$$S_{horizontal} = \frac{1250}{36.8355} = 33.9347$$

$$S_{vertical} = \frac{800}{23.5} = 34.0426$$

Assuming a feature within the image has 1/20 cycles per pixel horizontally and 1/24 cycles per pixels vertically (a feature that is 10 pixels long and 12 pixels tall), we would get the following absolute spatial frequency for that feature:

$$SF(\theta) = \sqrt{(33.9347 * (1/20))^2 + (34.0426 * (1/24))^2} = 2.2115$$

Therefore, the resulting feature would be assigned a spatial frequency of 2.2115 cycles per degree.

3.7 PUTTING IT ALL TOGETHER

Researchers have created several different models for approximating the curve of the CSF. One of the more popular models was developed by Manos and Sakrison [DeCarlo and Santella, 2002] and is as follows.

$$A(f) = 1040\left(0.0192 + 0.114f\right)e^{-(0.114f)^{1.1}}$$

This function will return the maximum possible contrast sensitivity, defined as one over the contrast, which is visible at a particular frequency *f*. The domain of this function forms a smooth curve which is shown in Figure 10. Contrast sensitivity values that rest above the curve represent areas where changes are not distinguishable to the human eye. Values the rest below the curve, represent areas where changes are visible to the human eye. Since the curve represents the border between visible and invisible, the further below the curve an area gets, the easier it is to detect changes is that area. In the equation, 1040 is simply a scaling factor. One can modify this value so that the contrast sensitivity axis has a domain of [0, 1] by changing this value to 2.6.

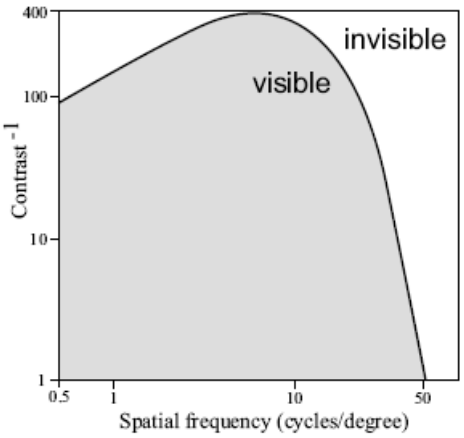

Figure 10: The curve of the CSF as given by Manos and Sakrison's proposed formula [DeCarlo and Santella, 2002].

Therefore, given the contrast and spatial frequency of a given area of an image, one can determine if changes in that area are visible to the human eye, and if so, by how much.

3.8 EDGES

Edge detection is a form of feature extraction which tries to highlight the basic structure of the objects within an image. Algorithms to do this usually work by looking at the luminance intensity changes. A variety of edge detection algorithms exist that optimally process different types of image content. Agrawala et al. [2004] first incorporated edges into a seam finding metric by dividing the color difference cost by the edge cost if the colors and edges matched in the two images. No specific definition was given for what represented a color match or in how this positively affected the output.

4 METHOD

Using knowledge of the human visual system, a new metric was developed to improve upon the standard RGB based metric. A uniform color space, edges, the contrast sensitivity function, and an alternative way of structuring the graph were used in finding a more optimal seam.

4.1 FACTORING IN A UNIFORM COLOR SPACE

For the color difference, the L*a*b* color space is used. Since the RGB color space is device dependent and may vary from computer to computer, an absolute version of this color space, known as sRGB was assumed so the color conversions could be consistent. The monitors for viewing the images were calibrated and set up to use the "sRGB Color Space Profile:" Once an absolute space is defined for the RGB values, they can easily be converted into the XYZ color space via a transformation matrix. The conversion from the XYZ color space to the L*a*b* color space is slightly more complex. The basic equations can be seen in Figure 11. X_n, Y_n, and Z_n are the values of the reference white point in XYZ space.

$$L^* = 116(Y/Y_n)^{1/3} - 16, if\ (Y/Y_n) > 0.008856$$
$$L^* = 903.3(Y/Y_n), otherwise$$

$$a^* = 500(f(X/X_n) - f(Y/Y_n))$$
$$b^* = 200(f(Y/Y_n) - f(Z/Z_n)),$$
$$f(t) = t^{1/3}, if\ (t) > 0.008856$$
$$f(t) = 7.787t + 16/166, otherwise$$

Figure 11: XYZ to L*a*b* conversion equations.

The basic color difference calculation for two colors in L*a*b* space is defined as follows [Hall, 1989]:

$$simpleLABDifference = \sqrt{\left((L_1 - L_2)^2 + (a_1 - a_2)^2 + (b_1 - b_2)^2\right)}$$

Calculation of the color difference in RGB space can be calculated in a similar fashion:

$$simpleMetric = \sqrt{\left((R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2\right)}$$

This simple definition of a color difference for RGB space will be used as the simple metric for the user study in chapter 5. As stated in chapter 2, more advanced color difference formulas exist for the L*a*b* color space [Sharma, 2005], however, these were not used this in study since a beneficial effect was not observed. During experimentation it was found that squaring the color difference seemed to aid in finding a better path, therefore, this enhancement was added into the new metric.

Figures 12 and 13 demonstrate the power of switching to a more uniform color space such as the L*a*b* color space. The same image, in this case an image of the cartoon character Waldo hidden within a grassy maze, is laid on top of itself with a 75% overlap. Figure 12 demonstrates the cut found by using the RGB cut metric described above while Figure 13 demonstrates the cut found by using the L*a*b* cut metric described above. Here we find that the cuts found are dramatically different. The RGB cut is a mostly horizontal cut that goes through the bottom of the image, leaving many of the maze characters mangled over the seam. The L*a*b* cut, on the other hand, takes a much more complex path. It moves in an upside down U path, hugging the grassy edge of the maze for much of its journey. Some of the maze characters still have limbs that are chopped off, but not nearly as many or as egregiously as the RGB cut.

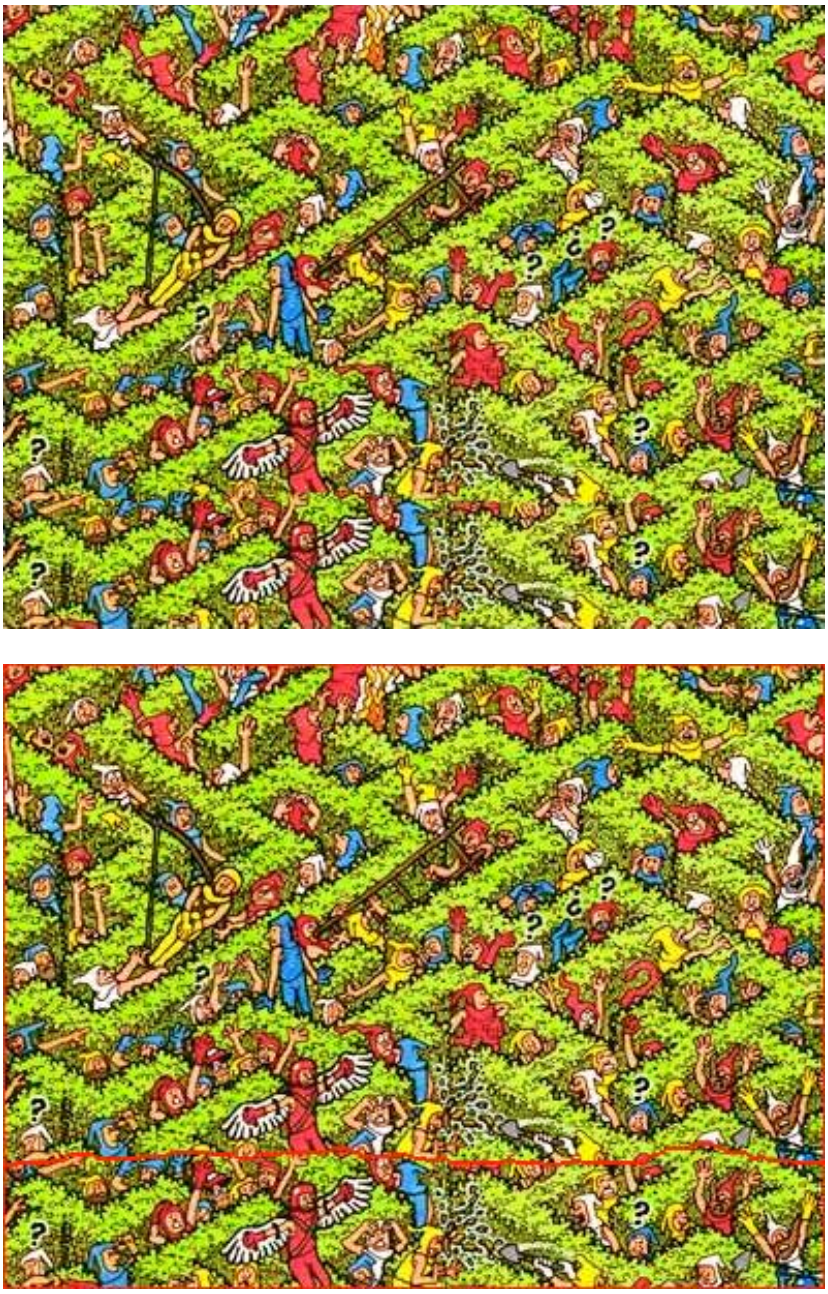

Figure 12: Waldo images cut together using the traditional RGB Cut metric. The image overlap is 75%. The top image is the output while the bottom image shows where the cut was made.

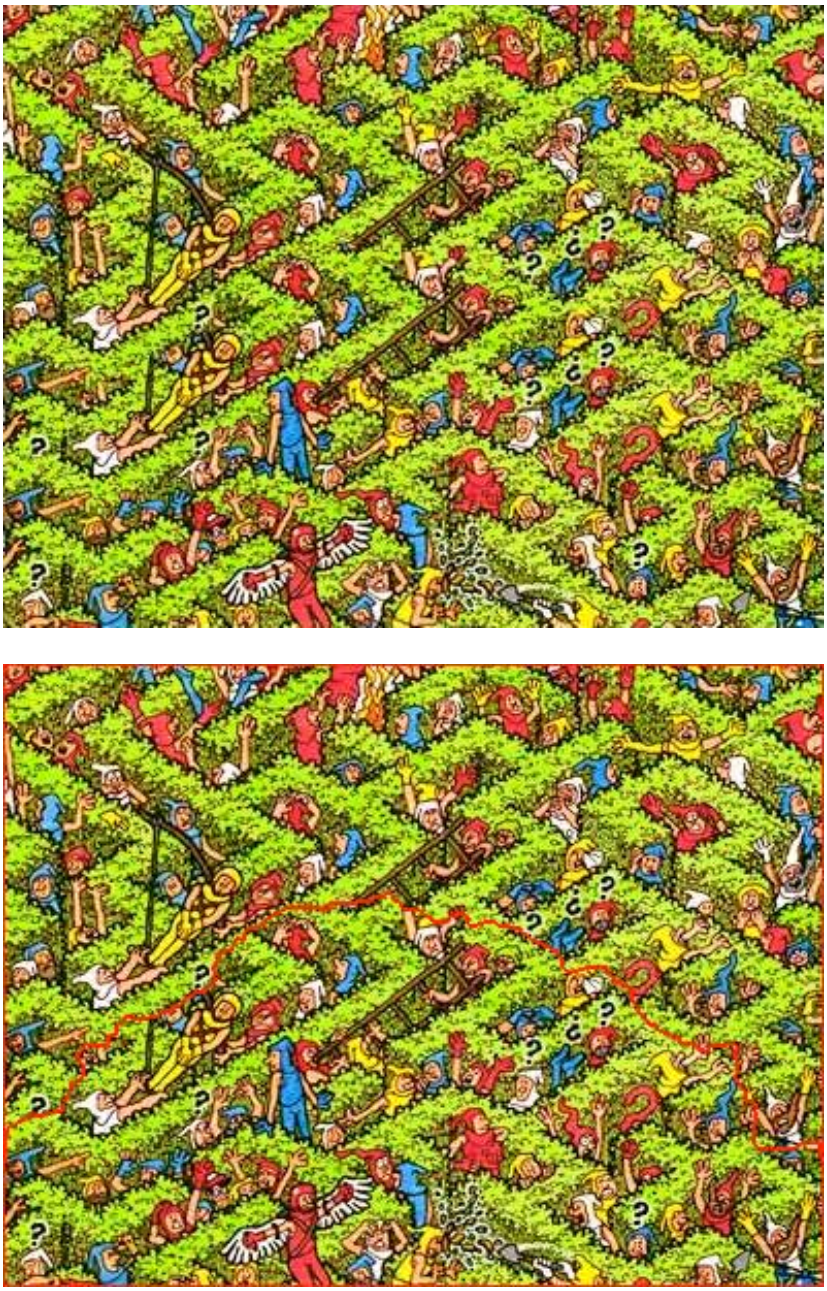

Figure 13: Waldo images cut together using a cut metric that involves the L*a*b* color space. The image overlap is 75%. The top image is the output while the bottom image shows where the cut was made.

4.2 FACTORING IN EDGES

The Sobel edge detection algorithm was used to find the edges in the overlapping images. This edge detection algorithm was chosen because it was the same one used by Agrawala et al. [2004] in their work with edges and improving seams. The choice to factor edges into the metric is based on whether or not the squared color difference between adjacent nodes is less than 400. This choice was made after testing a number of images and seeing what threshold they responded best to. When increasing this value it was found that factoring in edges with large color differences across the seam could hurt the final image, and making this number too small resulted in edges not being factored into images they could help.

Once the decision to use edge intensity in the cut is made, it is factored in by looking at each image's "edge map". The edge map is simply the output of the Sobel edge detection algorithm on the image. The minimum intensity between the images at position p is scaled down by a factor of 10 and used for position p's edge intensity. The color difference metric is divided by the edge intensity to make it more likely a seam will be found on an edge. The choice to scale down the edge intensity was made after testing a variety of images and using difference scaling factors to see what best worked. Through testing it was also found that using division instead of subtraction on the color difference was more beneficial for factoring in edges, therefore, division was used in creating the new metric. A value of 1 is used for the edge intensity if the calculated edge intensity value is less than 1.

Around half of the time this metric appears to have no effect on the seam that is found on the image. When applied to the Waldo images shown before, the same seam is found. However, there are many cases where it has a positive effect and in almost no cases does it seem to have a negative effect. Figures 14 and 15 show an example of how factoring in edges has improved the seam. In Figure 14, the seam maneuvers through the crowd, cutting a couple of the crowd members into each other and leaving two visible seams in the sky. In Figure 15, the seam carefully curves around the different individuals and even steps around the US Flag. Figure 16 shows an overlay of the two seams on top of each other and of the edge map that was used.

Figure 14: A crowd image cut together using a cut metric that involves the L*a*b* color space. The image overlap is 75%. The top image is the output while the bottom image shows where the cut was made. Due to the images length, the left sides of both images have been truncated since the seam did not venture into this area.

Figure 15: A crowd image cut together using a cut metric that involves the L*a*b* color space and edge maps of the images. The image overlap is 75%. The top image is the output while the bottom image shows where the cut was made. Due to the images length, the left sides of both images have been truncated since the seam did not venture into this area.

Figure 16: The top image shows the overlaying of the two different seams that were found by using edges and by not using edges. The bottom image shows the "edge map" that was used for determining where the edges were in the picture. Note that the edge map is only a single instance of the input image and that the results were truncated on the left hand side for size purposes.

4.3 FACTORING IN THE CSF

The CSF tells one how well they are able to see change in a given area. Using this function can allow us to avoid cutting through areas in an image where a user would be able to detect change. To factor in this component, CSF maps of an image were generated to show where the user would most likely see the most change. Figure 17 shows some CSF maps which were created to show which areas of the image were judged to be high areas of change. The brighter the area, the more change is visible, the darker the area, the less change is visible.

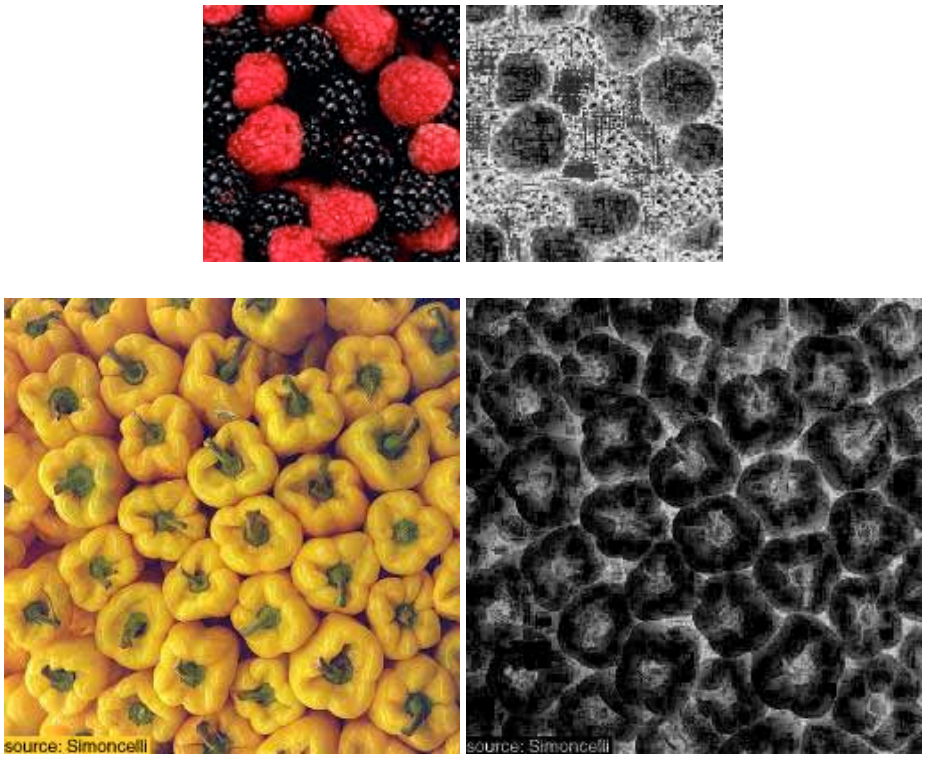

Figure 17: Images and their corresponding Contrast Sensitivity Function maps.

Once the CSF information about the image is obtained, the question is left as to how to factor it into the cut metric. One can reason that areas deemed invisible in both images

would blend well together if the colors also match. One can also reason that areas of extreme change in both images would probably not go well together unless they were blended together at the edges. Therefore, an additive approach was taken in factoring in the intensities of the visible differences in the images at particular locations. The maximum distance a value can be away from the CSF curve was taken and scaled down to 50. This was decided by experimenting with different ways of weighing this parameter and seeing what had the best effect. It was found if its value was too low, it had no effect, and if its value was too high, it could adversely effect the seam, therefore a domain of [0, 50] for a CSF value was used and added onto the color difference, since it seemed to work well. Through experimentation it was found that the best way to incorporate edges into this was to divide the sum of the squared color difference and the CSF value by the edge intensity. This is because the CSF value is often high around image edges, and when image edges line up it is often a good place to make a cut.

As with factoring in edges, CSF information does not always have a noticeable effect on all images. Many times the cut is simply jittered or slightly improved in certain areas. However, there are plenty of cases, just as in factoring in edges, where this information helps to find a superior seam. Figures 18 and 19 demonstrated how factoring in CSF intensities can improve the cut that is found. Figure 18 shows an image and the corresponding CSF map that was generated. Figure 19 shows how factoring this information in with the L*a*b* color information improved the cut over simply using an L*a*b* difference. The top image in Figure 19, which simply uses L*a*b* color information, shows a cut that recklessly cuts through the bottom of one of the two towers

shown in the center of the image. The cut slices the building to which the tower is attached. The cut that uses CSF information avoids this flaw and instead creates a longer seam that merges the two buildings from which the towers stem.

Figure 18: An Image and its corresponding Contrast Sensitivity Function map.

Figure 19: An image cut together with just L*a*b* color differences (top image and yellow cut) and an image cut together with L*a*b* color differences and CSF information (middle image and red cut).

4.4 AN ALTERNATIVE WAY OF SETTING UP THE GRAPH

The traditional way of setting the edge values on the graph is defined by the following function:

$$M(s,t,A,B) = \|A(s) - B(s)\| + \|A(t) - B(t)\|$$

Here we're telling the Max Flow algorithm that we want to find a seam through areas that are similar in both image A and B. However, this will not always lead to finding the least visible seam since similar areas in both images may not exist all the way along the cut. We can alter this concept with a slight change in how we look at the color difference between the pixels:

$$M(s,t,A,B) = \|A(s) - B(t)\| + \|A(t) - B(s)\|$$

In this version of the function we are looking at how the pixels across from each other differ in color. Therefore, we are telling the Max Flow algorithm to find a seam where the color difference across the seam is the most similar, thus leading to the possibility of a least visible cut between the images.

Using this new approach, we get slightly different seams than with the old approach, though both seams usually appear equally as useful. Other times, a better cut is found, as in Figure 20. Here one can see that near the middle of the cut, the original metric cuts off one of the tentacles of one of the sea creatures. In the new metric, a less linear path is found leading to a much less noticeable seam.

Figure 20: An example of using the old graph set up vs the new graph set up. The top image is cut using the old metric and the middle image using the new metric. The bottom image shows the seams for each image. The red represents the new metric and the yellow represents the old metric.

4.5 PUTTING IT ALL TOGETHER

With the alternative way of setting up the graph in place, the final improved metric for finding the best seam between two images is described by the following equation:

$$newMetric = \frac{(L * a * b * ColorDifference^2 + CSFValue)}{EdgeWeight}$$

Here we take into account a uniform color space, how things change within the image, and the boundaries that exist within the image. Through experimentation with various images, it was found that the squared color difference should have the most effect, then the edge weight, and then finally the CSF value. This way of setting up the function was decided through experimentation. Some parameters may help some images more than other; however, this function was weighted to help improve the overall outcome of the images it was tested on. Using the ideas expressed in this chapter, we can compute seams that are much less visible to the human observer.

5 RESULTS

Using techniques that take into account the human visual system, one can find seams between images that are much less visible than if one just takes into account the RGB values of the pixels. The following examples will demonstrate this. Each resulting picture comprises of an image placed over itself with a horizontal overlap of about 75%. The best cut between the images was then found using both the simple metric and the new metric. Both cuts are set to start and end at the same place: at the rightmost end of the overlap.

Figures 21-25 give examples of where the new metric significantly improves the cut that is found. In Figure 21, we see the new metric winds around the fish while the old metric leaves a noticeable seam in the fish it splices together at the bottom of its cut. Figure 22, 23 and 25 show how a more complex route is found via the new metric. In these cases, the old metric simply fails to find a good path and noticeable errors can be seen in the images. In Figure 24, the old metric finds a good path until it opts to cut through one of the sheep, resulting in an image error that sticks out to the human observer.

Figures 26 and 27 show cases where the new metric and old metric find seams that are similar to one another. Figure 28 gives an example where the new and old metric end up almost agreeing on the same seam to use for the overlapping images.

No cases were found where the old metric finds a good seam and the new metric does not.

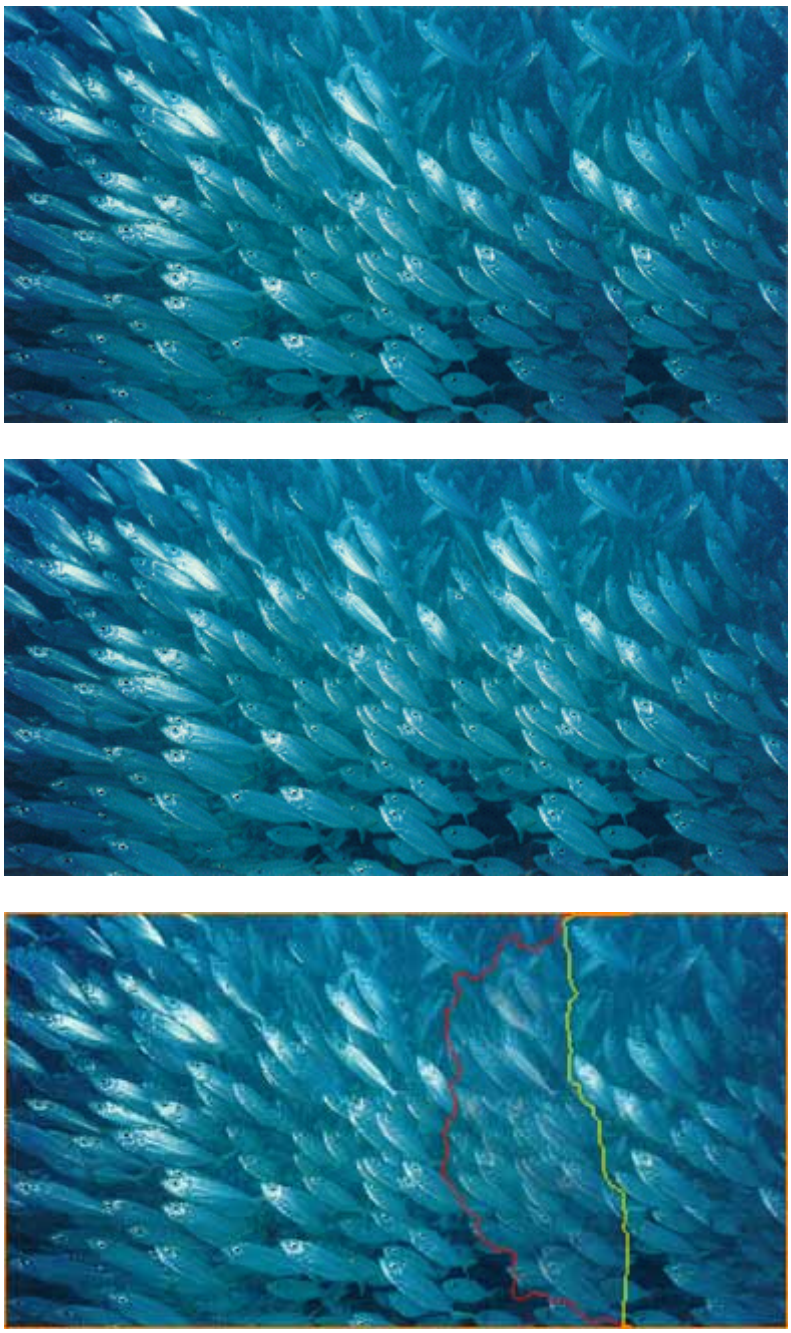

Figure 21: An image cut together using the standard cut metric (top) and the new metric (middle). The bottom image shows where the cuts were made. The old metric cut is in yellow and the new metric cut is in red.

Figure 22: An image cut together using the standard cut metric (top) and the new metric (middle). The bottom image shows where the cuts were made. The old metric cut is in yellow and the new metric cut is in red.

Figure 23: An image cut together using the standard cut metric (top) and the new metric (middle). The bottom image shows where the cuts were made. The old metric cut is in yellow and the new metric cut is in red.

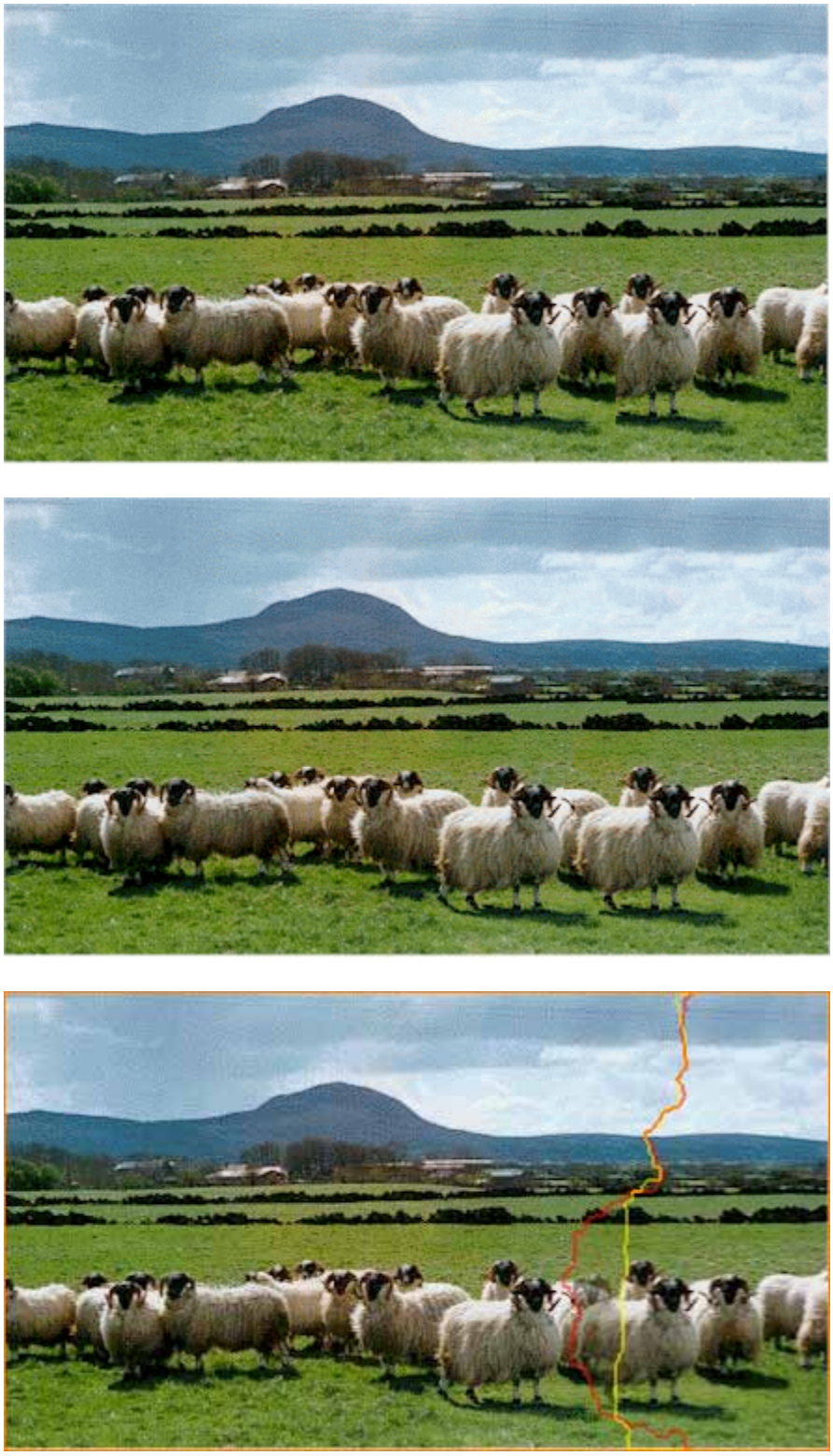

Figure 24: An image cut together using the standard cut metric (top) and the new metric (middle). The bottom image shows where the cuts were made. The old metric cut is in yellow and the new metric cut is in red.

Figure 25: An image cut together using the standard cut metric (top) and the new metric (middle). The bottom image shows where the cuts were made. The old metric cut is in yellow and the new metric cut is in red.

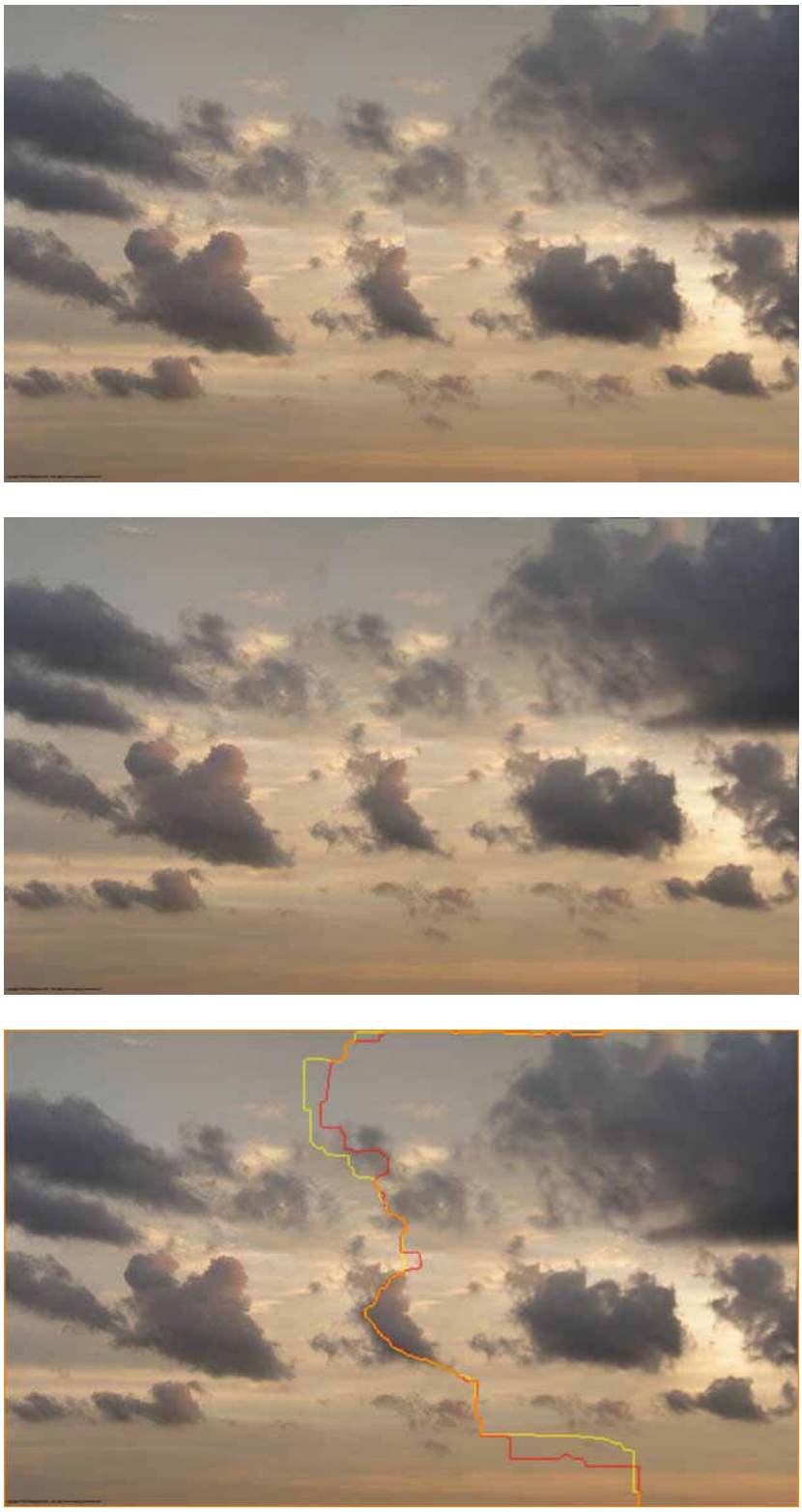

Figure 26: An image cut together using the standard cut metric (top) and the new metric (middle). The bottom image shows where the cuts were made. The old metric cut is in yellow and the new metric cut is in red.

Figure 27: An image cut together using the standard cut metric (top) and the new metric (middle). The bottom image shows where the cuts were made. The old metric cut is in yellow and the new metric cut is in red.

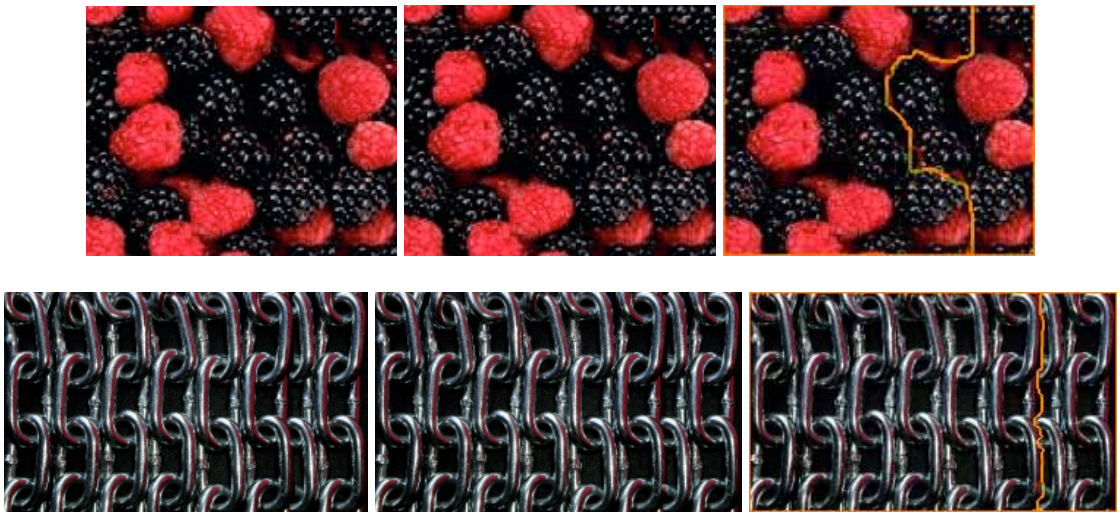

Figure 28: Here we see an example of where the old and new metrics find almost the same cut. The images cut together with the old metric are on the left and the ones cut together with the new metric are in the middle. The images on the right show where the cuts were made. The old metric cut is in yellow and the new metric cut is in red.

6 USER STUDY

In order to determine if a seam metric that incorporated knowledge of the human visual system made a significant difference over the simple seam metric, a user study was conducted. This study consisted of users sitting in front of a computer screen and indicating whether or not they saw a seam in the image in front of them. The image they viewed consisted of a picture overlapping itself by around 75%, this picture was stitched together using either the simple metric or the new metric. To avoid bias in image selection, the user would see the image twice during the test: once cut together with the simple metric and once cut together with the new metric.

6.1 USER STUDY DESIGN

The test contained 58 different image pairs, or a total of 116 images for the user to look at. The images with seams were divided up into three categories: luminance heavy images, scene images, and pseudo-pattern images. Luminance heavy images consisted of 8 pairs of cloud images. Scene images consisted of 8 pairs of photographs of real life scenes, for example, cities or crowds. Pseudo-pattern images consisted of 14 pairs of images that had a pattern like structure. The pattern images are the types of images for which one would most likely use for texture synthesis. In addition to these, 28 pairs of seamless images were put into the test. These images also fell into one the three basic categories and were used as a control group.

At the start of the test, the user was shown an image with a seam and it was explained to them what a seam was and how to tell if they saw one. They were informed on how the images were created and they were told to ignore repetition in the image as a sign that they had seen a seam. They were also told not to dwell too long on each image, to try and see if they could see a seam within 10-20 seconds. This time limit was not enforced, but the time a user spent looking at each image was recorded. To avoid image loading affecting this, all of the images were preloaded before the test began. Once the test was completed, the user was thanked and the results were emailed to an account to be assessed later.

6.2 USER STUDY RESULTS – RAW NUMBERS

A total of 15 people participated in the study. Users varied in ages ranging from 21 to 60. Seven of the users were female while eight of the users were male. Results varied from participants claiming to see as few as seven images with seams to claiming to see as many as 46 images with seams. On average, a participant claimed to see 24.07 images with visible seams.

This average breaks down on the type of images participants saw seams in is as follows: 13.47 images cut together with the simple metric (out of the 30 they were exposed to), 7.2 images cut together with the new metric (out of the 30 they were exposed to), and 3.4 images that had no seam (out of the 56 they were exposed to). The average time it took before a participant clicked the “yes” button to seeing a seam was 7.76 seconds for images cut together with the simple metric, 14.11 seconds for images cut together with

the new metric, and 16.87 seconds for images that did not have a seam. The average time it took to complete the test was 23 minutes and 27 seconds.

6.3 USER STUDY RESULTS – DISCUSSION

In general, participants seemed to have an easier time identifying seams in the luminance images. Many of the participants claimed they had an easier time finding the seam when it was one of the cloud images. Some commented that they thought this was because there was less going on in these images. They felt these images were easy to look at and that some of the pattern images gave them a headache. Looking for a seam in the pattern images was more of a difficult task rather than something that jumped out at them. Figure 26 shows a pair of images where the new and simple metric find a similar seam; however, more users saw the simple metric seam in this image. One user commented that this was because the simple metric made a mistake in the center of the image, which jumped out at them when they viewed the image.

The results also tend to show the participants saw more seams in images cut together with the simple metric for the pseudo-pattern and scene images. The majority of the time the simple metric's seam was seen more times than the new metric's seam in these images, though the data was not as striking as that of the luminance images. As for the images with no seams, though a typical participant claimed to have seen a seam in a few of these images, no particular fake image seemed to confuse users, and the number of false positives was always less than the true positives. Only three participants did not have any false positives.

In the previous Chapter, a number of figures were displayed demonstrating how the new metric and the simple metric compared to one another. Below is a table that shows how many users saw seams in those images.

	Number of participants who saw a seam in the image (simple metric version)	Number of participants who saw a seam in the image (new metric version)
Figure 21 (fish)	3	3
Figure 22 (radishes)	4	7
Figure 23 (clouds)	15	5
Figure 24 (sheep)	4	1
Figure 25 (old city)	9	3
Figure 26 (clouds)	15	6
Figure 27 (peppers)	11	2
Figure 28 (raspberries)	1	2
Figure 28 (chains)	0	0

As one expects, with this sample of images, the users tend to see the seam more often in the images cut together with the simple metric. There are a few unexpected results though. Though the seam in the new metric radishes image is more complex, more users claimed to see it than the simple metric version. The fish image results are also a little surprising. The same number of users claimed to see the seam in both versions. This means that although the simple metric makes a mistake, near the bottom of its cut, the mistake is in a place where many users tend not to look.

6.4 USER STUDY - SIGNIFICANCE

In order to access how significant the results were from the user study, a paired t-test was performed on the data. A paired t-test tells one if there is a statistical difference between two paired groups. This is shown by the following equations [Mathworld, 1999]:

$$\hat{X}_i = X_i - \overline{X}$$
$$\hat{Y}_i = Y_i - \overline{Y}$$

$$t = (\overline{X} - \overline{Y}) = \sqrt{\frac{n(n-1)}{\sum_{i=i}^{n} (\hat{X}_i - \hat{Y}_i)^2}}$$

The t-value will indicate if the difference between the groups is most likely caused by chance or not. Once computed, one needs to assign an alpha level and the degrees of freedom. In social research, the alpha value is commonly set to 0.05. This value means that 5% of the time you would find a statistical difference between the groups even if none existed. For a paired t-test, the degrees of freedom is calculated by taking the number of people in the group and subtracting one. Once these 3 values have been obtained, one needs to plug them into a table of significance for the t-distribution and see if the t-value is greater than what is listed. If it is, then there is a significant difference in the groups. A table of significance can be found in most statistical text books [Devore, 2000; Walpole, 1976]. Using an alpha level of 0.05 and 14 for the degrees of freedom, we get a value of 1.76 from the table of significance. Computing the data from the user study, we get a t value of 1.8. 1.8 is greater than 1.76, therefore there is a significant difference between the two groups, and thus the new metric has shown itself to be an improvement over the simple metric.

7 CONCLUSIONS AND FUTURE WORK

We have presented a new method for doing graph cuts between images that incorporates knowledge of the human visual system. The new metric utilizes a uniform color space, takes into account edges within the images, and extracts and uses information from the contrast sensitivity function. In addition to this, a novel way of modifying the graph set up is used to create less visible seams. A user study was carried out to compare the effectiveness of this system with a simple metric and it shows the new metric is a significant improvement.

Improving the seams between images has applications in patch based texture synthesis and digital photo modifications. Using techniques that take into account how we see, one can find the most optimal way of cutting images together. Future work could look into analyzing the images themselves and then selecting an optimal cut metric based on the image data. For example, an algorithm could detect that an image, or even an area of an image, is luminance heavy, and then do a cut based on luminance or contrast values for that area. This would be much more complex then simply relying on a single cut metric, but would allow for the specialization of cut metrics for particular types of images or image areas. Information on how we see could also possibly be incorporated into the placement of texture patches for texture synthesis.

8 BIBLIOGRAPHY

Aseem Agrawala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, Michael Cohen. “Interactive digital photomontage”, ACM Transactions on Graphics (TOG), v.23 n.3, August 2004.

M. Ashikhmin. “Synthesizing natural textures”, 2001 ACM Symposium on Interactive 3D Graphics, pp. 217–226. March 2001.

Serene Banerjee and Brian L. Evans. “Tuning JPEG2000 Image Compression for Graphics Regions”, Image Analysis and Interpretation, pp. 67-71. 2002.

Y. Boykov, O. Veksler, and R. Zabih. “Fast approximate energy minimization via graph cuts”, IEEE Transactions on Pattern Analysis and Machine Intelligence v.23 n.11, pp.1222–1239. 2001.

Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. “Wang Tiles for image and texture generation”, ACM Transactions on Graphics (TOG), v.22 n.3, July 2003.

Doug DeCarlo and Anthony Santella. “Stylization and abstraction of photographs”, ACM Transactions on Graphics, v.21 n.3 (Proceedings of SIGGRAPH 2002), pp. 769–776. 2002.

Jay Devore. “Probability and Statistics for Engineering and the Sciences: (Fifth Edition)”, Brooks Cole. 2000.

Alexei A. Efros and William T. Freeman. “Image Quilting for Texture Synthesis and Transfer”, SIGGRAPH 2001, pp. 341–346. 2001.

Alexei A. Efros and T. Leung. “Texture synthesis by non-parametric sampling”, In International Conference on Computer Vision, v.2, pp. 1033–1038, Sep 1999.

James Ferwerda, Sumant Pattanaik, Peter Shirley, and Donald Greenberg. “A Model of Visual Masking for Computer Graphics”. In Computer Graphics, Proceedings, Annual Conference Series, ACM SIGGRAPH, 1997.

L.R. Ford and D.R. Fulkerson. “Flows in Networks”, Princeton Univ. Press, 1963.

R. Hall. “Illumination and Color in Computer-Generated Imagery”, Springer, Berlin, 1989.

Bernhard Hill, Th. Roger, and F. W. Vorhagen. “Comparative Analysis of the Quantization of Color Spaces on the Basis of the CIELAB Color-Difference Formula”, pp. 109-154, April 1997.

David J. Heeger and James R. Bergen. “Pyramid-based texture analysis/synthesis”, Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp.229-238, September 1995.

Garrett M. Johnson and Mark D. Fairchild. “Full-Spectral Color Calculations in Realistic Image Synthesis”, IEEE Computer Graphics and Applications, v.19 n.4, pp. 47-53, July/August 1999.

S.L. Kilthau, M. Drew, and T. Moller. “Full Search Content Independent Block Matching Based on the Fast Fourier Transform”, In ICIP02, I: 669–672. 2002.

Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, Aaron Bobick. “Graphcut textures: image and video synthesis using graph cuts”, ACM Transactions on Graphics, v.22 n.3 (Proceedings of SIGGRAPH 2003), pp. 277-286. 2003.

Mathworld. “Paired T-Test”, 1999. < http://mathworld.wolfram.com/Pairedt-Test.html >

N. M. Moroney and M. D. Fairchild. “Color space selection for JPEG image compression”, Journal of electronic imaging, v.4 n.4, pp. 373–381. 1995.

Andrew Nealen and Marc Alexa. “Hybrid texture synthesis”, In Proceedings of the 14th Eurographics workshop, pp. 97–105. 2003.

M. Nischik and C. Forster. “Analysis of Skin Erythema Using True-Color Images”, IEEE Trans. Med. Imaging, v.16 n.6, pp. 711-716. 1997.

Izumi Ohzawa. “Make Your Own Campbell-Robson Contrast Sensitivity Chart”, July 2006. <http://ohzawa-lab.bpe.es.osaka-u.ac.jp/ohzawa-lab/izumi/CSF/A_JG_RobsonCSFchart.html>

E. Peli. “Contrast in complex images”. JOSA A, v.7, pp. 2032-2040. 1990.

Martin Reddy. “A Measure for Perceived Detail in Computer-Generated Images”. Technical Report ECS-CSG-19-96. Jan 1996.

Martin Reddy. "Perceptually Optimized 3D Graphics". IEEE Computer Graphics and Applications, v.21 n.5, pp. 68-75. 2001.

J. Serup and T. Agner. “Colorimetric quantification of erythema—A comparison of two colorimeters (Lange Micro Color and Minolta Chroma Meter CR-200) with a clinical scoring scheme and Laser-Doppler flowmetry”, Clin. Exp. Dermatol., v.15, pp. 267–272, 1990.

G. Sharma, W. Wu, and E. N. Dalal. “The CIEDE2000 Color-Difference Formula: Implementation Notes, Supplementary Test Data, and Mathematical Observations”, Color Research and Application, v.30 n.1, pp. 21-30. February 2005.

C. Soler, M.-P. Cani, and A. Angelidis. "Hierarchical pattern mapping". ACM Transactions on Graphics v.21 n.3, pp. 673–680. July 2002.

Murray R. Spiegel. “Theory of Statistics”, McGraw-Hill Book Company. 1961.

H. Takiwaki, L. Overgaard, and J. Serup. “Comparison of narrowband reflectance spectro-photometric and tristimulus colorimetric measurement of skin color—Twenty-three anatomical sites evaluated by the DermaSpectrometer and the Chroma Meter CR-200”, Skin. Pharmacol., v.7, pp. 217–225. 1994.

Nick Vavra. “Texture Quality Extensions to Image Quilting”, 2003.

R.E. Walpole. “Elementary Statistical Concepts”, New York, NY, Macmillan Pub. Co., 1976.

I. L. Weatherall and B. D. Coombs. “Skin color measurements in terms of CIELAB color space values,” J. Invest. Dermatol., v.99, pp. 468–473, 1992.

Li-Yi Wei and Marc Levoy. “Fast texture synthesis using tree-structured vector quantization”, Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp.479-488, July 2000.

Wikipedia. “MacAdam ellipse”, Wikipedia, The Free Encyclopedia. 14 July 2006. <http://en.wikipedia.org/wiki/MacAdam_ellipse>

Nathaniel Williams, David P. Luebke, Jonathan D. Cohen, Michael Kelley, and Brenden Schubert. “Perceptually guided simplification of lit, textured meshes”. Proceedings of the 2003 symposium on Interactive 3D graphics. April 2003.

G. Wyszecki and WS Stiles. “Color Science: Concepts and Methods, Quantitative Data and Formulae”, John Wiley & Sons, Inc., New York, second edition, 1982.

Y. Xu, B. Guo, and H. Shum. “Chaos mosaic: Fast and memory efficient texture synthesis”, Tech. Rep. MSR-TR2000 -32, Microsoft Research, 2000.