Applying the Compass Operator to Texture Edge Detection

Stephanie J. Wojtkowski
Swarthmore College
jill@sccs.swarthmore.edu

April 29, 2002

Abstract

Segmenting contiguous regions of similar texture from images is a problem whose solution continues to elude computer vision researchers. Methods such as Laws' texture energy measures, cooccurrence matrices, Fourier transforms and wavelet transforms have been applied with only moderate success. This thesis proposes an adaptation of Ruzon and Tomasi's compass operator [8] to texture images for more effective texture edge location. The operator is highly successful at finding color edges and possesses some characteristics that make it well-suited for texture segmentation. To adapt it to texture analysis, a convolution filtering method like Law's texture energy measures or cooccurrence matrices is first applied to the image. The compass operator is then applied to find edge magnitudes and directions in the filtered image. A robustness test shows that this method is relatively impervious to noise. However, the method is sensitive to the distance metric that is used for comparing histograms. Both the L^2 norm and histogram intersection were used unsuccessfully due to the nature of natural texture images. In contrast, the dynamic time warping method provided an accurate distance measure that produced successful results.

1 Introduction

The field of color segmentation has been well-explored, primarily because useful metrics for color are well-defined, providing an accurate characteristic that can be used for color segmentation. However, texture segmentation continues to be a growing area of research because of its complexity. A typical segmentation will isolate individual texture primitives, a result which is both time consuming and uninformative. Segmenting texture in a more meaningful way presents a more difficult problem for several reasons. First, researchers have thus far been unable to devise a detailed, comprehensive metric for describing textures. The existing metrics, such as energy, entropy, coherence, etc. are not easy to interpret because they represent local characteristics that change as you move through a texture. Most often it is not clear exactly what texture feature each of these metrics represents, and when it is clear the classification is crude at best.

Another reason why texture segmentation is more difficult than color segmentation is that a texture does not comform to one value under a filter, as a color does. Instead, the texture is subject to a distribution of values when filtered. An operator like the Sobel operator will not be able to detect this sort of edge because it does not take into account the distribution of values around the edge. Thus, a histogram matching technique is more suited to this application. The proposed compass operator uses histogram analysis to detect edges.

A third difficulty is that one texture could be locally similar to another but globally different. Many filter operators focus on the local characteristics near a pixel to detect an edge and would not be able to detect these kinds of edges. The proposed compass operator will pick up on larger trends in the image without being dominated by local characteristics.

2 Related Work

Texture analysis is typically conducted using two types of methods: statistical and syntactic methods. Included within the statistical methods are filtering techniques such as Laws' texture energy measures or cooccurrence matrices and frequency based techniques like Fourier and wavelet transforms. These techniques currently dominate the field of texture analysis due to their documented success. Included among syntactic techniques are shape chain grammars and primitive grouping. Section 2.1 and Section 2.2 describe these techniques in greater detail. Section 2.3 describes common methods for interpreting statistical data.

2.1 Statistical Methods

Statistical methods attempt to use the results of localized texture metrics to make statements about the composition of an image. Characteristics such as texture energy or contrast might be computed and then evaluated to segment texture regions. Statistical methods have been used for texture segmentation with varying degrees of success. They include Law's texture energy measures, cooccurrence matrices, and Fourier and wavelet transforms.

2.1.1 Laws' Texture Energy Measures

Law's texture energy measures are a set of filters designed to identify specific primitive features such as spots, edges and

ripples in a local region [1]. The origin of the Laws' filters are three vectors:

$$L_3 = (1, 2, 1) \text{ to smooth}$$

$$E_3 = (-1, 0, 1) \text{ to locate edges (first difference)}$$

$$S_3 = (-1, 2, -1) \text{ to identify spots (second difference)}$$

These three vectors can be convolved with themselves and each other to produce five $5x1$ vectors. The resulting vectors are

$$L_5 = (1, 4, 6, 4, 1)$$

$$E_5 = (-1, -2, 0, 2, 1)$$

$$S_5 = (-1, 0, 2, 0, -1)$$

$$W_5 = (-1, 2, 0, -2, 1)$$

$$R_5 = (-1, -4, 6, -4, 1)$$

The names of the vectors are mneumonics for Level, Edge, Spot, Wave and Ripple, which indicate the kind of feature that each filter is designed to detect. These $5x1$ vectors can be multiplied to produce a set of $5x5$ masks that are then applied to the image.

The Laws' texture energy measures were implemented so that the proposed compass operator could be compared with the k-means clustering technique. Figure 1 shows the results of applying four different Laws' filters to an image containing two textures.

2.1.2 Cooccurrence Matrices

Cooccurrence matrices [2] are a method for identifying patterns of repeating intensity values in an image. Let m and n be the lengths of the sides of the rectangular region to be filtered. Let ϕ be the angle at which matches will be searched for, and let d be the distance at which the match is sought. Let B be the number of buckets into which the 0 to 255 pixel intensity range is divided to reduce the amount of stored information. For each mxn region in the image, a BxB matrix $P_{\phi,d}$ is created. Each location (x, y) in the matrix contains the number of times a pixel in intensity bucket x was found to have a pixel in bucket y at a distance of d and an angle ϕ in the given region. This algorithm will produce a symmetric matrix showing the frequency of the given intensity pairings at the specified direction and distance.

Once the matrix has been constructed, several statistics which describe the filtered region can be calculated from it [3]. Energy, entropy, contrast and homogeneity are found according to the following formulas:

$$Energy = \sum_{m,n} P^2_{\phi,d}(m, n)$$

$$Entropy = \sum_{m,n} P_{\phi,d}(m, n) \log_2 P_{\phi,d}(m, n)$$

$$Contrast = \sum_{m,n} (m - n)^2 P_{\phi,d}(m, n)$$

$$Homogeneity = \sum_{m,n} \frac{P_{\phi,d}(m, n)}{1 + |m - n|}$$

Figure 2 shows the results of applying cooccurrence matrices to the same image pictured in Figure 1. Matrices were calculated for angles $\theta = 0°, 90°$ and for $d = 2, 6$. Then the energy, entropy, contrast and homogeneity were calculated for each of the four combinations of d and θ. The four filtered images in Figure 2 are the energy, entropy, contrast and homogeneity calculated with $d = 2$ and $\theta = 90°$.

2.1.3 Fourier Transforms and Wavelets

The Fourier transform of an image involves decomposing the image in the frequency domain [4]. The image is represented as a sum of sinusoidal waves with different periods and amplitudes. One of the difficulties with this approach is that the domain of an image is finite, whereas sine waves are infinite. Consequently, the Fourier transform uses complex combinations of sine waves to create destructive interference that zeroes the signal outside the domain of the image. This results in a lot of extra overhead simply to zero out most of the domain. Also, the individual sine waves that the transform is composed of commonly do not correspond directly to image features in any meaningful way, since the image is unlikely to contain periodic features.

The wavelet transform does not suffer from the weaknesses of the Fourier transform. First, some wavelets possess the quality of compact support, meaning that they are non-zero only over a finite domain, like an image. Consequently, the complex pattern of destructive interference is unnecessary in this case. Second, the wavelets could conceiveably each correspond to an image feature since they each describe a local image feature. Thus, the wavelet decomposition presents valuable information about the composition of the textures in an image. In addition, wavelet solutions tend to converge more quickly than their Fourier counterparts.

2.2 Syntactic Methods

Statistical methods have dominated texture analysis because of their relative success. However, there are other methods which have shown some merit. Syntactic techniques such as shape chain grammars or graph grammars [5] are among the viable alternatives to statistical methods. Shape chain grammars involve describing each texture with a grammar similar to those found in formal language processing. Each texture primitive is described by a symbol, and the arrangement of symbols in a string represents the spatial relationships between primitives. Graph grammars label each texture primitive as a node in a graph, with the edges of the graph indicating the connectivity of the primitives. The difficulty in both

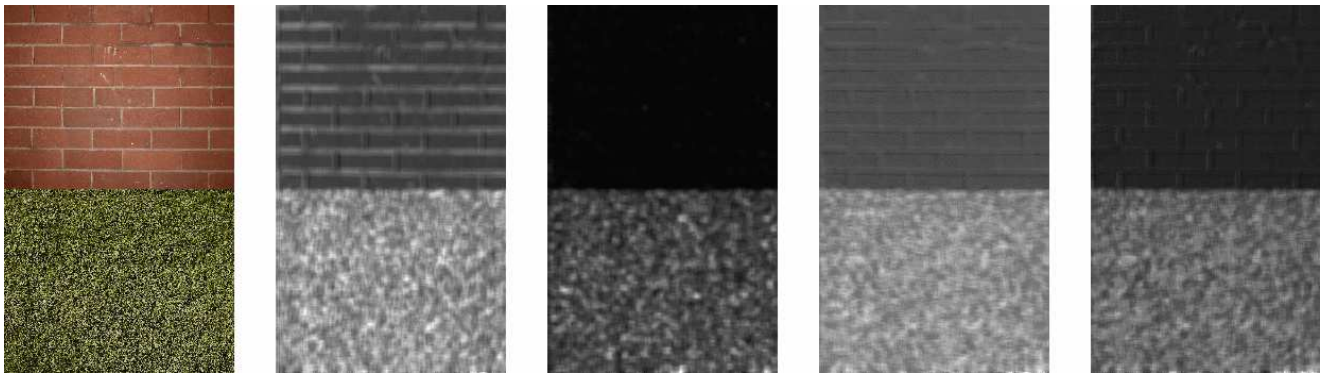

Figure 1: The image on the left has been filtered with a suite of Laws' filters. From left to right, the image has been filtered with ExE, RxR, SxW and WxW. The pixel intensity indicates the magnitude of the filter response for all images in this thesis.

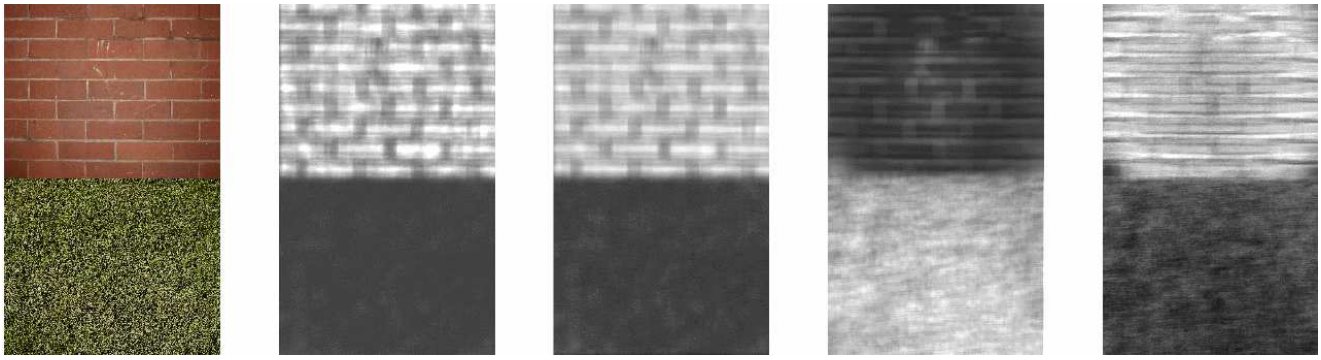

Figure 2: The original image and the corresponding energy, entropy, contrast and homogeneity, respectively. The calculations were performed for $d = 2$ and $\theta = 90°$.

of these methods is creating models of the texture primitives, particularly for pseudorandom textures or for textures where the primitives are partially occluded, such as in images of grass.

2.3 Interpreting Statistical Data

Once statistical data has been collected over an image, it must be interpreted to produce meaningful pixel groupings according to texture. Two of the most common methods for achieving this are k-mean clustering and normalized cuts. These methods cluster the data based on the statistics independently collected at each pixel and are thus highly suitable for creating non-contiguous segmentations. However, texture regions are assumed to be contiguous, providing a property that can be exploited by segmentation techniques. We therefore suggest applying Ruzon and Tomasi's color compass operator to texture segmentation since it is designed to locate prominent edges in an image. The two traditional approaches, as well as our novel approach, are described below.

2.3.1 K-Means Clustering

K-means clustering [6] is a simple way to group a collection of n-dimensional vectors into k distinct groups. Initially, k vectors are chosen at random from the space of n-dimensional vectors in the images to represent the initial cluster centers. The first data vector is then compared to each of the k centers to find the closest one, call it i. One of several distance measures could be used, including Euclidean distance and the Earth mover's distance. The vector is then added to the i-h cluster, and its position is averaged with the centroid of the cluster to form a new centroid. This process is repeated for each vector until they have all been exhausted. The result is a clustering of the vectors into k distinct sets, each of which represents a principle component of the data.

To apply this technique to texture segmentation, a texture image is first processed using a collection of filters such as Laws' filters or cooccurrence matrices. The resulting images are then used collectively to segment the image. Given n filtered images, an n-dimensional vector exists at each pixel. In other words, each pixel has n values associated with it, one for each value that it acquired after being filtered with one of the filters. These n values can be treated as a vector in

n-space that describes the texture characteristics at that particular pixel. Dividing these vectors into k clusters produces a segmentation of the image into k distinct texture regions.

This method has the advantages of being conceptually simple and computable in $O(n)$ time, where n is the number of pixels in the image. The major disadvantage of this method is its ability to produce non-contiguous segmentations. If a texture varies greatly on the local level but conforms to a global distribution, this method will cluster the texture into many small, local clusters. Also, the program must be told precisely how many clusters to look for before it calculates the clusters. This prevents the segmentation process from being entirely autonomous by requiring human intervention.

Examples of clustering based on k-means clustering can be seen in Figure 3. Each row shows the original image followed by the results of k-means in combination with cooccurrence matrices and Laws' filters, respectively. The top image is an example where k-means clustering performs very well, whereas the bottom row shows an image that the algorithm cannot accurately segment. The clustering program was asked to find 2 and 3 clusters, respectively.

2.3.2 Minimum Cut and Normalized Cut

Another way of interpreting the statistics described above is to represent the image as a graph where each pixel is a node in the set of all pixels, V. There is an edge between each pair of nodes (pixels) in the image, with the set of all edges denoted E. The weight of an edge in the graph is determined by the similarity of the two nodes that form the endpoints of the edge as well as the Euclidean distance between the nodes. Under this representation, the problem of segmenting the image is transformed into a problem of finding the optimal cut(s) in the graph. A cut is defined as a partitioning of the nodes in the graph into 2 groups, A and B, such that $A \cup B = V, A \cap B = \emptyset$. The value of the cut is calculated from the weights of the edges that have one endpoint in A and the other in B.

The typical method for quantifying a cut is to define the cost of the cut as the sum of the weights of the edges that are removed to achieve the cut:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

where $w(x, y)$ indicates the magnitude of the edge between nodes x and y. The cut that minimizes this value, called the minimum cut, could be considered the optimal cut of the graph. The minimum cut approach has the advantage of being a well-documented problem that can already be solved efficiently. However, this method proves to be too simple to accurately reflect the perceptual grouping in the image. Looking for the minimum cut favors segmenting out a small, isolated set of nodes, which may not be the best segmentation strategy. The normalized cut criterion [7] was proposed as an alternative to the minimum cut that provides a more meaningful graph partitioning. Instead of simply minimizing the weights of removed edges, the normalized cut minimizes the proportion of the cut edges to the sum of all of the edges that touch a vertex in either partition:

$$normalizedcut = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}$$

where

$$assoc(A, V) = \sum_{u \in A, t \in V} w(u, t)$$

This method encourages a partitioning along the loosest associations in the graph, as desired. It is unlikely to partition isolated nodes because the cut value for removing a few nodes is a very large portion of the total association of those nodes. Some results of using normalized cuts on images can be found in [7]. This method has the advantage that it can be solved using a generalized eigenvalue system, for which there are already efficient algorithms. However, graph cut methods like the two presented in this paragraph suffer from the same disadvantage as the k-means clustering method: both need to know the desired number of clusters in advance.

2.3.3 Compass Operator

The compass operator [8] is a recent development in color edge detection. Conceptually, a compass of fixed radius is first dropped on a pixel. As the "needle" is spun around, the difference in color between the semicircles on either side of the needle is calculated. The needle direction that produces the largest value is accepted as the edge, and the difference is the magnitude of that edge. Thus, a value like a gradient is produced with a direction and a magnitude. The Earth mover's distance is used to compute the difference between the two halves.

This operator is largely the basis for this thesis. The operator is highly successful in detecting color edges in images, even with significant textural influences. In this thesis, we desire to adapt this color edge detection technique to texture edge detection, since the operator is well-suited for both edge detection and texture analysis. The operator's structure makes it a natural edge detector, and the use of histogram analysis is suitable for texture analysis since a texture tends to exhibit a distribution of values under a filter instead of a single value.

This method has an advantage over the k-means and noralized cut approaches because it exploits the contiguous property of texture regions. Instead of performing a point-by-point clustering of pixel characteristics, it instead looks at the macro structure of the image and tries to find locations that are likely to be edges. It also does not need advanced knowledge of the number of clusters desired, making it fully autonomous.

3 The Compass Operator

The compass operator was originally designed to find edges in an image containing several regions of different colors. In this thesis, we attempt to adapt the compass operator to texture segmentation. The compass operator can be applied to an image in several ways. If G is the set of edge angles that are tested with the compass, the most obvious method for applying the compass is to create G filter images, one per angle. Each image contains a circular mask for the image that designates which half of the compass a pixel is in for the given angle. Each filter image is then compared to the target image pixle-by-pixel for filtering. This method is simple, but it is clumsy and requires alot of unnecessary storage space, particularly for high granularities. We chose to use a more elegant vector-based approach which only necessitates the storage of one compass image regardless of angle granularity. Each pixel in the compass is treated as a vector from the compass origin, and the dot product of this vector with a perpendicular to the compass needle indicates which half the pixel falls on. This information is used to calculate histograms for the two semicircles of the compass and subsequently find the edge direction of maximum magnitude.

The remainder of Section 3 explains the details of the compass operator. Section 3.1 explains the necessity of preprocessing the image and describes the kinds of preprocessing filters that may be used. Section 3.2 describes the normalization process that was applied to ensure that no single preprocessing filter would dominate the compass results. Section 3.3 describes the contents of the configuration file that is used by the system. Section 3.4 demonstrates the mathematics behind the construction of the compass operator. Section 3.5 shows how the compass operator is applied to the image after preprocessing. Finally, Section 3.6 describes how the filtered compass images are combined to produce a direction and magnitude for each edge in the image.

3.1 Filtering the Original Image

When the compass operator was applied to color images, no preprocessing was necessary. The distance in color space between adjacent pixels is an adequate measure of the similarity of two pixels, and thus can be used to determine the similarity of colors along an edge. Since the measurement of color is inherent to an image, it is unnecessary to apply a color metric before using the compass operator.

Trying to distinguish texture regions is a more complex problem. Before edges can be found, a method must be devised to quantify texture features so that their magnitudes can be compared. The problem of defining and measuring texture continues to be an area of active thought. Since our interest is in the interpretation of texture measures and not in their construction, we chose to use preexisting texture measures to preprocess the image. The two methods selected were Laws' filters and coocurrence matrices. Figure 1 and Figure 2 show examples of these filters.

3.2 Normalization

Once the image has been preprocessed, the filtered images are normalized so that no filter dominates the edge magnitude calculation. Normalization according to standard deviation was selected because it colors pixels relative to their distance from the mean value in the image. The average value of the pixels in a filtered image is first calculated. The standard deviation of the pixel values is then computed based on this mean, redistributing the values in an equitable way. The standard deviation measures the spread of values in a distribution. For a pixel (x, y) in the image, the following calculation is performed:

$$intensity(x, y) = \frac{intensity(x, y) - mean}{standarddeviation}$$

This computes how many standard deviations the value of (x, y) is from the mean. The results are then restricted to the range -2 to 2, since approximately 95% of values occur in these four standard deviations. The values are then shifted so they are all positive and stretched to the range 0 to 255. The resulting image is sent to the compass operator. This normalization procedure prevents one preprocessing filter from dominating the compass results as well as allowing for the combination of several different types of filters on equal footing.

3.3 Configuration

Before the compass can be applied to the filtered image, the framework for filtering is prepared. The compass system first reads in a configuration file. This file includes the desired compass radius, as well as the number of angles that the figure will test, called the granularity. If, for example, the user would like to test at a granularity of 4, the compass needle would be inserted at $0°, 45°, 90°$ and $135°$. The 4 angles that are each $180°$ from those above are omitted because testing them would be redundant. Figure 4 shows an example of a compass with granularity 4. The four tested angles are numbered and drawn in black, while the omitted angles are drawn as dashed grey lines.

The configuration file also contains the names and dimensions of the preprocessing filters to be run. To make the compass more accurate, all pixels that are within a preprocessing filter's radius of the compass needle are omitted from processing. This is necessary because the preprocessing of these pixels included information from pixels on the other half of the compass. Omitting these pixels gives a more accurate measure of the gradient across the two halves. Figure 5 shows a compass generated with a granularity of 5. The dark gray pixels are one half, the light gray pixels are the other half, and the black pixels are those that are omitted for a preprocessing filter of radius 5. Each circle has a radius of 30 pixels.

All of the information in the configuration file is read in, and then the image is preprocessed. The compass is then

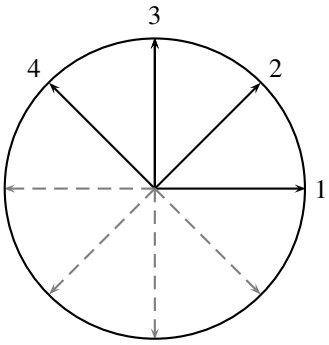

Figure 4: Example of a compass drawn at granularity 4. The four numbered vectors are those that are tested for high edge magnitudes. The four grey vectors are not included because they are redundant.

constructed from the parameters in the configuration file using the method decribed in Section 3.4.

3.4 Creating the Compass

The creation of the compass requires several steps. The important features of the compass are depicted in Figure 4. First, a circle of the given radius r is drawn in an image buffer. As before, let G be the set of angles that are tested for a given granularity. The coordinates for the endpoints of unit vectors from the origin toward each direction in G are cached in an array for rapid access.

Before the compass is applied to the image, the perpendicular distance of each point on the compass from the compass needle is calculated. The compass is prepared for these calculations as depicted in Figure 6.

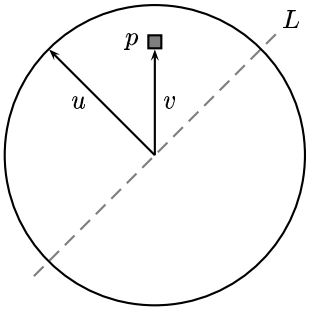

Figure 6: Diagram of the compass applied at a particular orientation, θ.

First, an imaginary line L is drawn on the circle to bisect it along the direction of the compass needle. The endpoints of a unit vector u that is perpendicular to L and starts at the origin are then calculated. This vector can be used to find the distance of a point p from L. If v is a vector from the origin to p, then

$$v \cdot u = d$$

where the magnitude d is the distance of v from L and the sign of d indicated which half of the compass p is on. d is cached for each pixel in the compass to make filtering the image faster and simpler.

3.5 Applying the Compass

There are two main components to applying the compass to a filtered image. Section 3.5.1 describes the method used to calculate the semicircle histograms of the compass operator. Section 3.5.2 addresses the issue of selecting a metric for comparing histograms.

3.5.1 Computing the Semicircle Histograms

The preprocessing stage produces a set of filtered images, one for each initial filter application. The next step is to apply the compass operator to each pixel in each image in this set. The compass is applied seperately for each angle in the given granularity. The process for a given compass angle g is as follows. Two histograms are initialized in advance to record the distribution of values in each semicircle of the compass. The compass filter is placed on top of each pixel in a filtered image. Let the current pixel be denoted (x, y). For each pixel (m, n) in the compass, the value of d is examined. If the preprocessing filter had radius f, then we compare the value of d with the value of f.

If $d > f$, then (m, n) is in the first semicircle. The absolute address of (m, n) when the compass is centered at (x, y) is calculated. The magnitude of the resulting location in the filtered image is then added to the histogram for the first semicircle.

If $d < -f$, then (m, n) is in the second semicircle. The absolute address of (m, n) is calculated in the same way, and the value of the resulting location is added to the histogram for the second semicircle. Any other values of d are ignored, either because they are outside the compass or because they are in the region between the two semicircles that we are ignoring as described in Section 3.3.

3.5.2 Comparing Histograms

Once the two histograms are calculated, the magnitude of the difference between them must be calculated so that edges can be identified. The simplest method, and the method that was attempted first, is to perform a histogram matching. The sum squared difference (L^2 norm) of the number of pixels in corresponding buckets of the two histograms is calculated. This method suffers from two large problems. First, if the histogram granularity is too fine then regions that are very close in intensity but not exactly equal will have a large histogram difference. Second, histogram matching is dominated by

outliers, whereas we are looking for the overall distribution of a texture, and the outliers will produce skewed results.

A second method for calculating the similarity of two histograms is to compute their intersection. This method is similar to calculating the intersection of two sets. For corresponding buckets in two histograms, the smaller value is added to the intersection sum. This essentially calculates the number of histogram entries that are common to both histograms. This method was also tested with the compass operator, but it did not perform any better than the histogram difference method.

It was surprising to find that both of the methods above produced good results on synthetic images and poor results on similar real texture images. However, the reason for this phenomenon became evident upon closer examination. In each synthetic image in Figure 9, the color of non-noise pixels on one side of the edge is uniform. This creates a histogram in which one particular bucket will have a very high value and the surrounding intensities will have relatively low values. Since this peak will occur at different values on different sides of the edge, the histogram difference is high and the edge easy to find. At the same time, two histograms in a location with no edge will both have the peak at the same value, giving them small histogram differences.

In a real texture image, the values on one side of an edge are not perfectly uniform. Though the values will cluster in one area of the histogram, this non-uniform property will produce a large histogram difference. Figure 7 shows two sample histograms. The distributions and means of the two curves are very similar, but a histogram difference or histogram intersection will find them to be very different because they have different values for corresponding intensities. Ideally, the distance metric used to compare histograms would not be influenced by such minor differences in intensity. Dynamic time warping was selected as a method that would minimize these effects.

Dynamic time warping [9] is a dynamic programming approach to finding a correspondence between two signals in time. Each of the two B bucket histograms can be represented as a signal that evolves over B timesteps, with the histogram magnitude determining the magnitude of the signal at a given time. Given time s from histogram #1 and time t from histogram #2, the histogram difference between all possible s and t is computed and stored in a sxt cost matrix, call it C. The goal of dynamic time warping is to find the ideal (minimum cost) matching between the two signals. This corresponds to finding the minimum cost path from $(0, 0)$ to (s, t) in C. The path is found by iterating through the entries in C. For each entry e, the three entries down and to the left of e are examined for the minimum cost path that leads from $(0, 0)$ to a spot adjacent to e. The cost of being at point e is added to this minimum value and reinserted into the matrix as the cost of the path from $(0, 0)$ to e. The process continues until (s, t) is reached. At this point, the minimum cost to get from $(0, 0)$ to (s, t) will be located in matrix position (s, t). To ensure that the points in the signals were matched only with other points in relative temporal proximity, the cost values for matrix locations that did not exhibit this quality were set to a prohibitively large value. This effectively prevents any paths from including those correspondences.

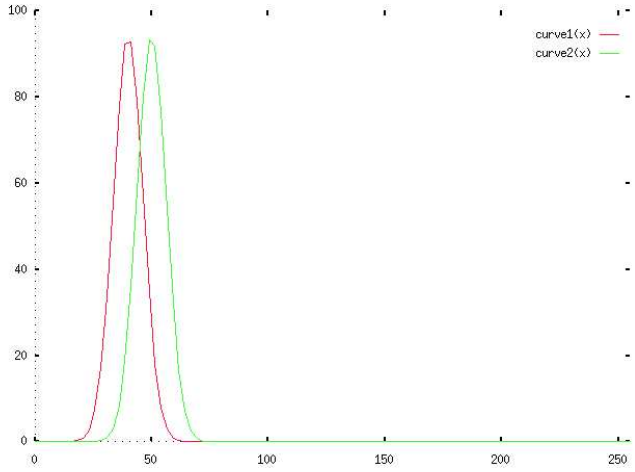

Figure 7: Example of two histogram distributions. The two are similar, but a histogram difference or histogram intersection will find them to be very different. Dynamic time warping successfully matches the two signals so their difference is minimized.

3.6 Calculating Edge Strength

Once the compass operator has been applied to each filtered image and the maximum histogram difference and angle has been calculated for each pixel, the results need to be combined into one edge image. A simple summation and thresholding technique was chosen to accomplish this task. For a given pixel, the values of that pixel in all filtered images are summed to create a single value for that pixel. The value is then compared to a threshold, with values less than the threshold set to 0 and values greater than it left alone. The resulting image is then normalized to be in the range 0 to 255.

4 Results

The compass operator was applied to the left image in Figure 8 to ensure that it was working properly. The result, shown on the right of the figure, indicates that the compass operator successfully locates ideal edges. A compass of radius 10 was used, along with a granularity of 4. As is clear in the result image, an edge of very high magnitude was found along the diagonal of the image, corresponding with the edge in the original image. The edge direction was also found to be $135°$ all along the edge, which is correct.

After testing the correctness of the compass operator, its robustness was explored. The compass operator was applied to images containing a single edge subject to varying degrees of noise. The results are shown in Figure 9. The images in

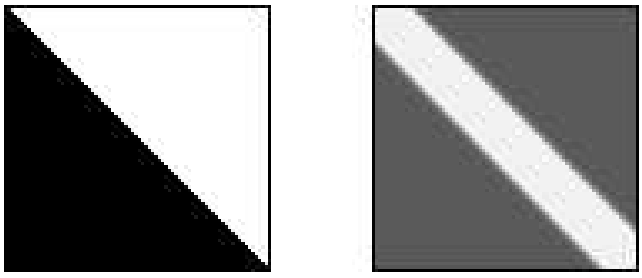

Figure 8: Left: Image used to test the compass operator. Right: The result of filtering the image at left with the compass operator. The brightness of a pixel represents the magnitude of the edge found at that point. An L^2 distance metric was used to compare histograms.

the top row contain noise percentages of 15%, 30%, 45%, 60% and 75%, from left to right. The bottom row shows the results after filtering with the compass operator. Even when the image is 75% noise, the compass does a reasonable job of finding the edge.

The system was tested on real images with varying results. Figure 10 shows the result of running the compass on an image of natural texture with a sharp edge. The original image is shown, followed by the edge magnitude results using the L^2, histogram intersection and dynamic time warping methods, from left to right. It is apparent from the image that the L^2 distance metric produced highly erroneous results. The histogram intersection performed slightly better, but the dynamic time warping technique clearly presented the best results.

Figure 11 shows the results of running the system on more complex natural images. The images were prefiltered with Laws' filters. Dynamic time warping was used as the distance metric for all images, along with a granularity of 4 and a compass radius of 10. Each column (a)-(d) has the following format: the top image is the original; the middle image is the results of filtering and edge detection, without thresholding; the bottom image is the result of edge detection with a threshold of 2400.

5 Conclusion and Future Work

There are several important results that are evident from this research. First and most significant, the choice of distance metric can make a profound difference in the success of the compass operator. Though the compass results were initially promising based on synthetic images, the effects of a naive choice in distance metric led to disappointing edge images. However, when the same system was used with dynamic time warping as the distance metric, good results were obtained.

Since the distance metric used to compare histograms appears to have a profound affect on the results of this system, a more comprehensive search for the ideal metric would be useful. Methods such as Earth movers distance or clustering might be even more successful than dynamic time warping and are worth exploring.

Another improvement upon the current system would be to filter all of the preprocessed images together. Instead of calculating edges in each image and then combining the results, an n-dimensional compass filter could be applied, where n is the number of filtered images produced by preprocessing. The result would be two n-dimensional histograms, one for each compass half. The two histograms could then be compared using Earth movers distance or vector quantization. This would produce a more unified view of an image region than individual processing can provide. Also, a more meaningful preprocessing filter technique could be employed.

Another necessary improvement to the system is the determination of a reasonable threshold to separate edges from non-edges in the result image. Since filter size is likely to have a large effect on the magnitude of edges in the image, it may be necessary to write a routine that adapts the threshold value to the current filter value. Other options include automatic threshold detection techniques and thresholding according to mean and deviation.

Finally, the edges detected in the image should be reduced to lines of pixel width to make them clearer. This could be accomplished most easily by performing a hough transform with non-maximal suppression. However, other methods such as skeletonization or line thinning may be more suitable since most natural edges are not linear.

Since the results in Figure 11 appear to be promising, this report will be submitted for publication once the edge detection has been improved and a more extensive test image set has been explored.

6 Acknowledgements

I would like to thank Dr. Bruce Maxwell of Swarthmore College for his patience, good advice, and thoughtful criticisms. I am also greatly indebted to Meg Spencer of Cornell Library at Swarthmore College for her assistance in searching for the resources to write this thesis, as well as the Department of the Army and the City Library of New York for sending me some much-needed articles through Interlibrary Loan. Finally, I would like the thank Dr. Kenneth Laws for taking the time to discuss the strengths and limitations of his texture measures with me through a series of e-mail conversations.

References

[1] K.I.Laws. "Texture Energy Measures." In *DARPA Image Understanding Workshop*, p. 47-51, 1979.

[2] R.M. Haralick, K. Shanmugam, and I. Dinstein. "Textural Features for Image Classification." In *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-3, 6:610-621, Nov. 1973.

[3] C.C. Gotlieb and H.E. Kreyszig. "Texture Descriptors Based on Co-occurrence Matrices." In *Computer Vision, Graphics and Image Processing*, vol. 51, 1:70-86, 1990.

[4] K.R. Castleman. *Digital Image Processing*, Englewood Cliffs, N.J.: Prentice Hall, 1996.

[5] M. Sonka, V. Hlavac and R. Boyle. *Image Processing, Analysis, and Machine Vision*, Pacific Grove, CA: PWS Publishing, 1999.

[6] J. MacQueen. "Some Methods for Classification and Analysis of Multivariate Observations." In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, 281-297, 1967.

[7] J. Shi and J. Malik. "Normalized Cuts and Image Segmentation." In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, 8:888-905, 2000.

[8] M.A. Ruzon and C. Tomasi. "Color Edge Detection with the Compass Operator." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 160-166, June 1999.

[9] L.R.R. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition*, Englewood Cliffs, NJ: Prentice Hall, 1993.

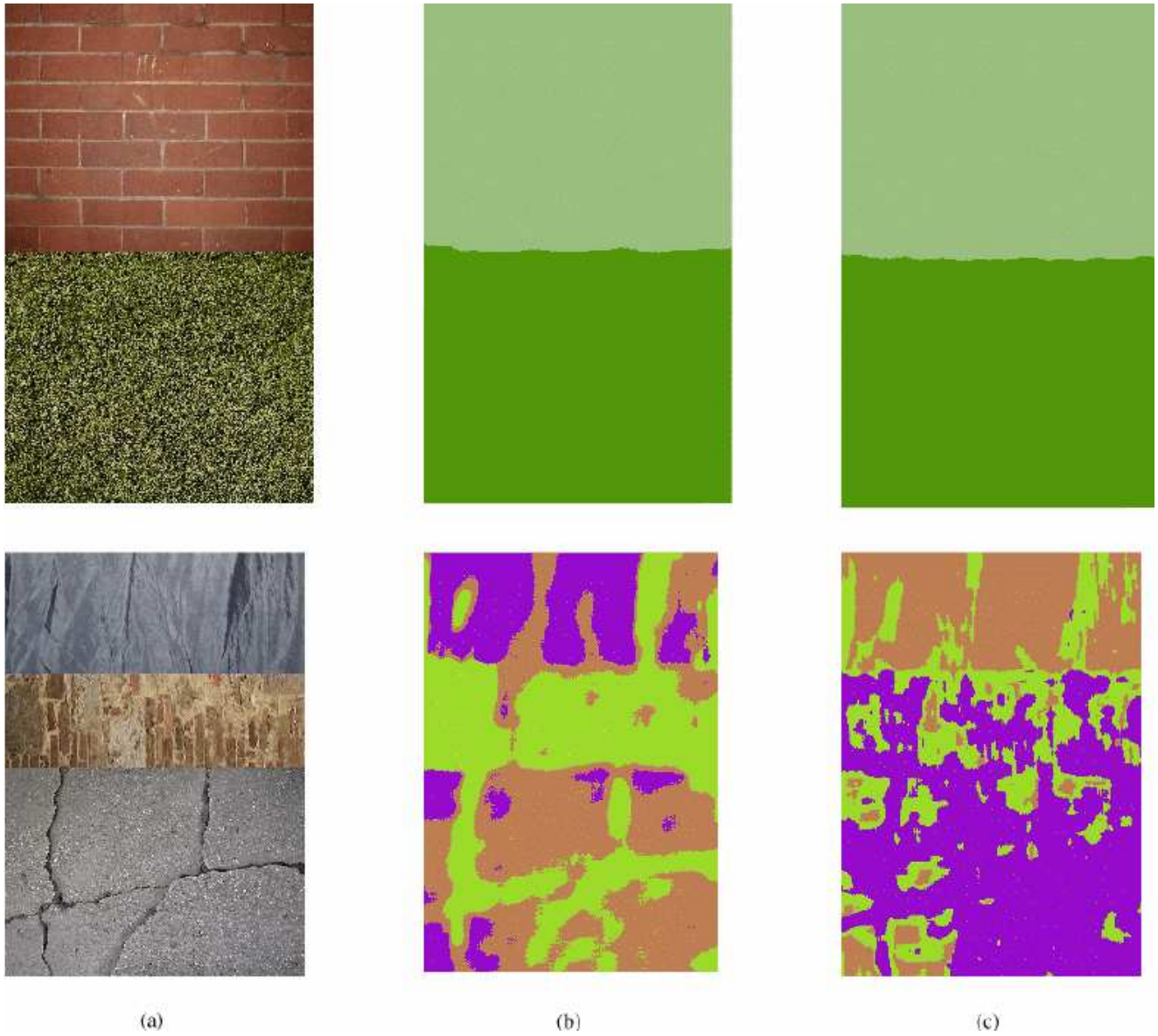

Figure 3: Each row shows the original image followed by the results of k-means clustering. The top row was processed with 2 clusters, while the bottom rowed used three clusters. (a) Original images. (b) Images processed with cooccurrence matrices and k-means clustering. (c) Images processed with Laws' filters and k-means clustering.

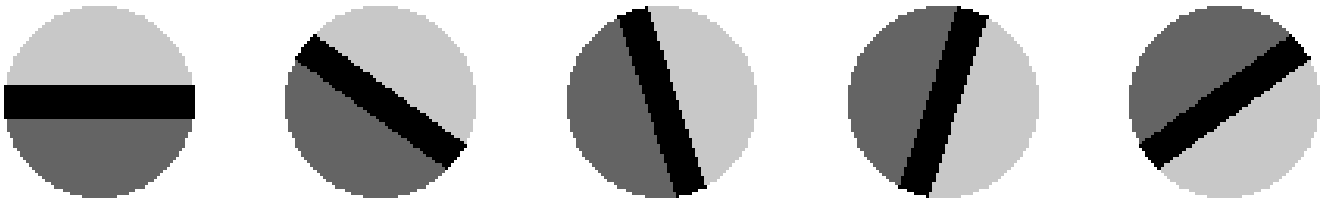

Figure 5: A compass of radius 30 with granularity 5. The dark gray pixels are one half, the light gray pixels are the other half, and the black pixels are those that are omitted for a preprocessing filter of radius 5.

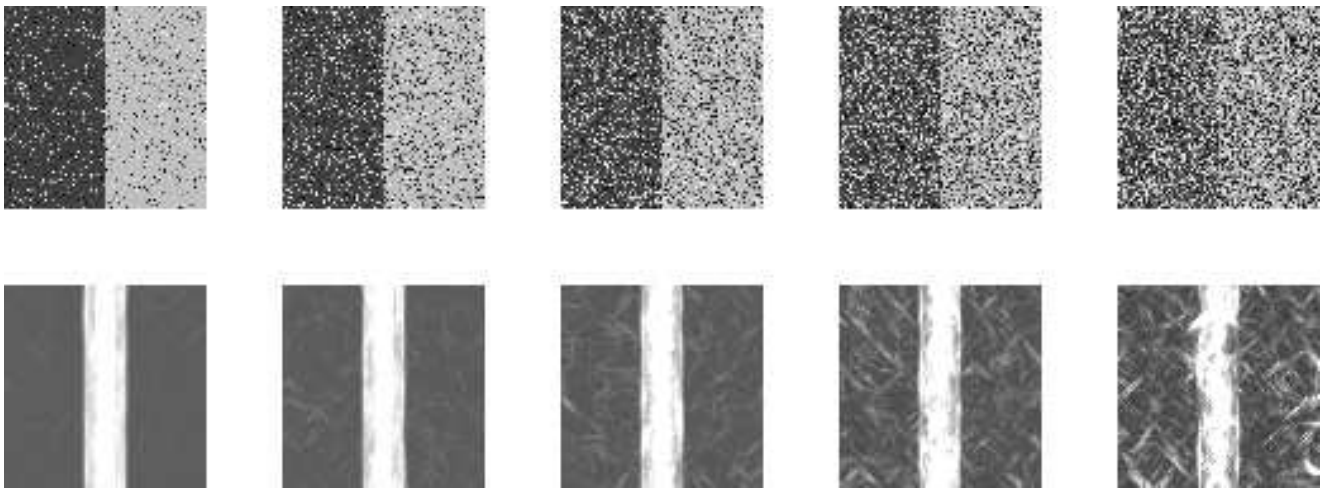

Figure 9: Top: Images of an edge prepared with noise percentages of 15%, 30%, 45%, 60% and 75%.
Bottom: Results after filtering the images above with a compass operator of radius 10. The brightness of a pixel represents the normalized magnitude of the edge found at that pixel. An L^2 distance metric was used to compare histograms.

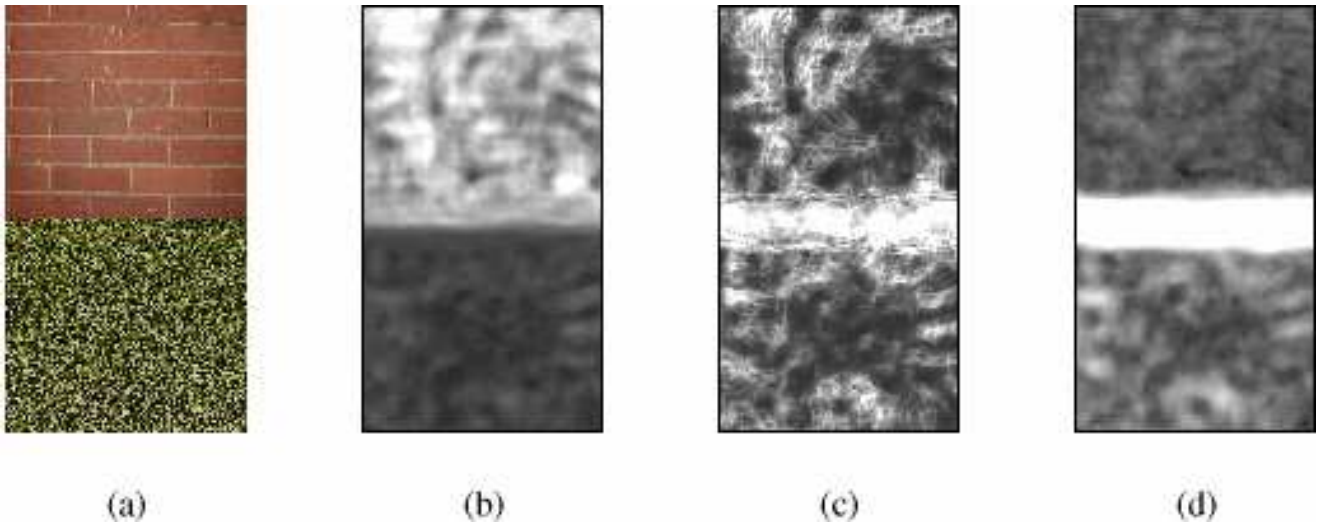

Figure 10: Results of running the compass operator on a natural image using difference distance metrics. The edge magnitude threshold was set to 0 so that all edge magnitudes would be visible for comparison. (a) The original image. (b) The filtered image using an L^2 norm. (c) The filtered image using histogram intersection. (d) The filtered image using dynamic time warping.

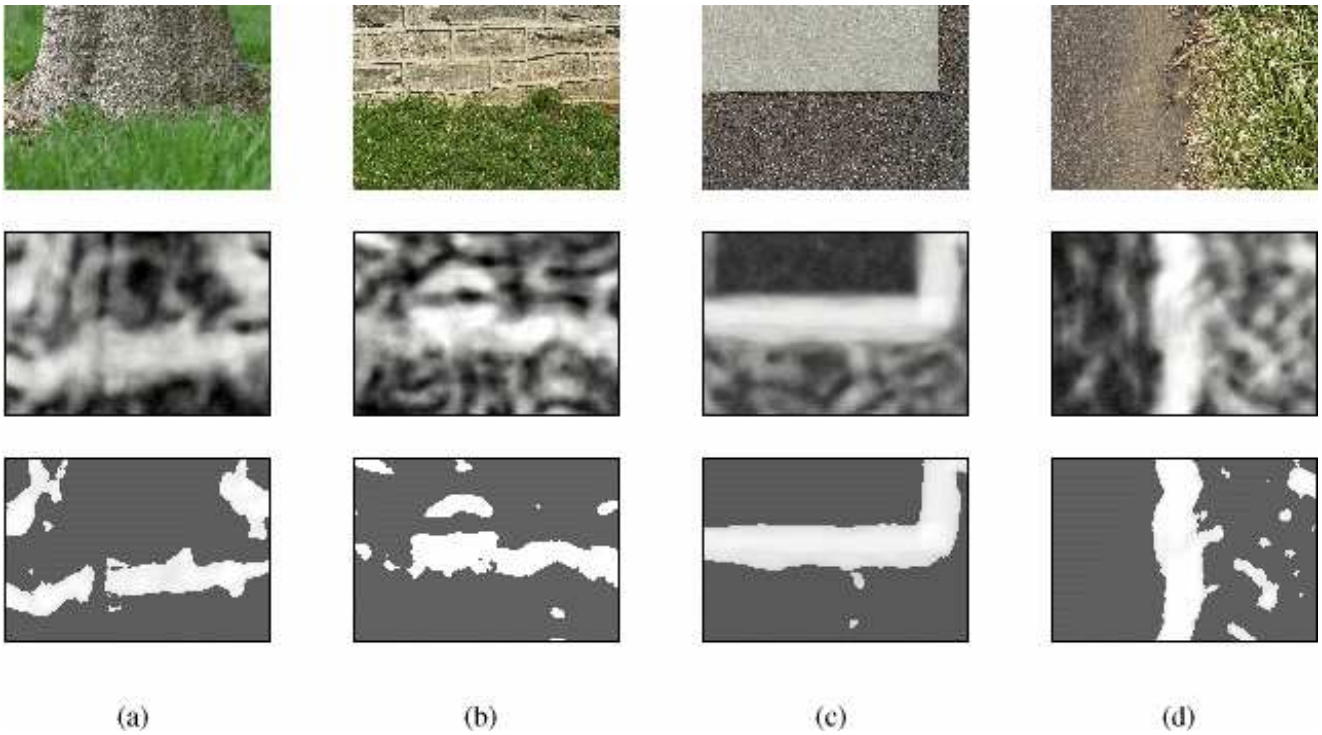

Figure 11: Results of running the system on more complex images. Each column (a)-(d) has the following format: the top image is the original; the middle image is the results of filtering and edge detection, without thresholding; the bottom image is the result of edge detection with a threshold of 2400.

Appendix 1: Compass Operator System Code