Texture Edge Detection Using the Compass Operator

Bruce A. Maxwell
Dept. of Engineering
Swarthmore College
500 College Ave.
Swarthmore, PA 19081
maxwell@swarthmore.edu

Stephanie J. Brubaker
Georgia Institute of Technology
College of Computing
801 Atlantic Drive
Atlanta, GA 30332-0280
stephanie@cc.gatech.edu

Abstract

The compass operator has proven to be a useful tool for the detection of color edges in real images. Its fundamental contribution is the comparison of oriented distributions of image features over a local area at each pixel. This paper presents extensions and modifications to the operator to make it applicable to texture edge detection in high dimensional images whose dimensions represent the output of a texture filter bank. The results show that the extended compass operator can robustly locate edges in natural scenes with complex textures. In addition, the use of a dynamic time warping distribution matching metric and jittered application of the operator improves the computational running time by a factor of over 50 while still producing comparable results. This large-scale speedup makes application of the algorithm to an entire image database computationally feasible.

1 Introduction

The capability to identify and segment texture regions in an image is critical to the analysis of natural scenes. Texture analysis encompasses texture description, texture segmentation, and segmentation's dual problem of texture edge detection. Accurate, reliable texture analysis is a hard problem that currently has no consensus solution.

All surfaces are textured at some scale, and many surfaces are textured at a scale that is between 1-10 pixels. Within this range pixel-wise or small-area operators for detecting coherence are difficult to use because the characteristics of the surface that are coherent exist at a scale larger than the operator. Therefore, we must consider operators that are designed to compare distributions of color or texture features rather than scalar values.

The compass operator, proposed by Ruzon and Tomasi [7], was designed to compare color distributions and detect edges in RGB images. At each pixel it compares the color distributions on either side of an oriented circle for a variety of orientations using the earth-movers distance, a robust histogram-matching method. The output is a vector of match values, one for each orientation. The overall pattern of the vector, and the magnitude of the largest value, are features that correlate to edges, corners, and homogeneous areas. The strengths of the compass operator are its use of larger area distributions and

BMVC 2003 doi:10.5244/C.17.56

the use of sophisticated matching functions to compare them. The major weakness is its computational running time, which is on the order of 30 minutes for one 640x480 image on a 1.8GHz Athlon.

The strengths of the compass operator make it appropriate for edge detection in texture images as well as RGB images. We define a texture image to be a multi-dimensional image where each dimension is the output of a filter. We make no assumptions about the filters used except that they be reasonably bounded in size; texture images can include both chromatic and achromatic filters, and RGB images are included as a special case.

Straight RGB images, however, do not completely capture the nature of most textures. RGB distributions–often represented as histograms–do not, in general, retain spatial patterns. Texture, on the other hand, possesses a spatial arrangement of pixels that is critical to its description. Thus, two identical RGB distributions can produce markedly different textures that have a perceptible boundary.

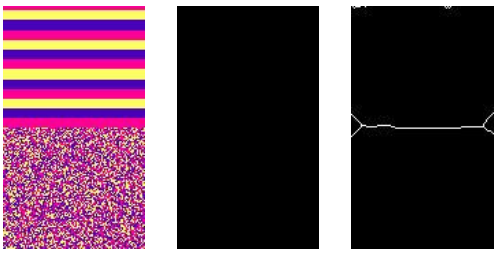

Figure 1: Example image with two textures (left), result of compass operator using RGB (middle), result of compass operator using texture (right)

For example, Figure 1 shows an image on which the compass operator fails to detect the texture boundary using RGB distributions, while applying the same operator to the output of texture filters generates a strong boundary between the textures. The lines shown in these images are the result of thresholding the magnitude of the gradient of the compass operator and executing a morphological thinning operation on the result.

The use of oriented filters in the creation of a texture image captures some of the spatial patterns present in a small image area. The distributions of these patterns should be consistent over an area of uniform texture, provided that the distribution is observed at a scale larger than the inherent texton size. Examples of oriented filters previously used in texture analysis include spatial filters such as Laws filters or Gabor filters and frequency-space filters such as Fourier, discrete cosine and wavelet transforms [4].

Comparing two distributions is not a trivial task, however. If we represent distributions as histograms, there are numerous comparison methods for them in the literature. Common ones include the L1 and L2 metrics–sum of the absolute or squared difference between corresponding buckets–and histogram intersection–sum of the minimum value of corresponding buckets in each histogram [10]. The problem with these histogram matching techniques is that they only compare corresponding buckets. In both RGB and texture image histograms it is quite possible to have two very similar histograms that nevertheless receive a poor match. In an extreme case, the two distributions could be delta functions offset by a single bucket. In this case, while the actual image regions would be nearly identical, an L1, L2 or histogram intersection match function would say the distributions were maximally different.

This is not the first paper to identify this problem. Shen and Wong first proposed a metric for comparing texture distributions that involves unfolding a 1-D histogram so that each bucket is repeated as many times as its weight [9]. Werman, Peleg and Rosenfeld extended this metric to N-dimensions [2]. However, the complexity of the metric in N dimensions is $O(M^3)$, where M is the number of elements in the histogram. Comparing two 10x10 image patches, therefore, is $O(10^6)$, which is prohibitive when applied at each pixel of an image for multiple orientations.

For the original compass operator, Ruzon and Tomasi used the Earth Mover's Distance [EMD], which defines the match distance as the minimum amount of work–in a transportation sense–to transform one histogram into another [6][7]. The EMD metric produces intuitive histogram comparisons, but is computationally demanding. Working with 3-plane images (RGB), Ruzon and Tomasi reported a 33 minute run-time for a 768 x 512 image [8].

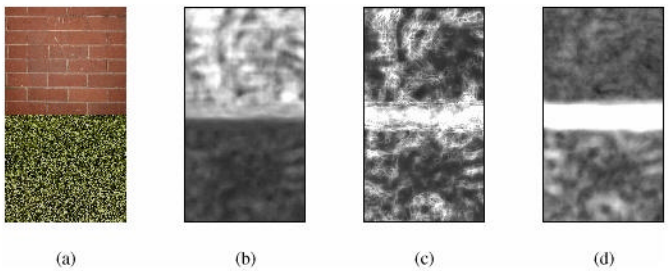

Figure 2: Comparison of metrics for comparing distributions: a) Original texture image, b) Euclidean L2 distance, c) histogram intersection, and d) DTW per plane

In order to extend the compass operator to higher-dimensional images–where the number of dimensions may be as high as 17-20–we propose a metric based on dynamic time-warping [DTW] that executes plane-wise comparisons and then combines the results across texture planes in a post-process [3]. This metric makes it computationally feasible to analyze large high-dimensional texture images while still producing comparable results. Figure 2 shows a direct comparison of Euclidean L2, histogram intersection and DTW per plane on a simple texture image, with intensity showing gradient magnitude.

In addition, we pair the DTW with a jittering approach to the compass operator application since its large size makes it an inherently low-pass operator.

These proposed modifications of the compass operator enable its application to texture edge detection in high-dimensional texture images in a reasonable amount of time (30s for a 320x240 image). The implementation does not rely upon particular texture filters, making it generally applicable to a wide variety of analysis techniques. The extended compass operator should be viewed as a post-process to standard texture analysis using filters. As such, it provides a general mechanism for detecting oriented changes in local distributions.

2 Theory and algorithms

The two major methods of describing and analyzing texture are statistical and structural, or syntactic. Statistical methods further break down into spatial and frequency-based methods. Spatial filters include Laws' texture energy filters and cooccurrence matrices. Frequency-based methods include Fourier, discrete cosine, and wavelet transforms. Syntactic methods attempt to break a texture into symbolic units and then represent the texture in terms of the relationships of the symbolic elements.

Statistical methods currently dominate the field of texture analysis. However, there is no clear consensus on which statistical methods work best. Randen and Husoy undertook a quantitative evaluation of statistical and frequency-based approaches to texture segmentation [4]. Their conclusion was that none of the methods they tested worked well on a large variety of textures. Because there is no clear consensus choice, we selected a simple, traditional approach to texture analysis and used Laws' texture energy filters as the basis for generating texture images [1]. Note that any spatial or frequency-based filter set could be used with our scheme.

Laws' texture energy measures are a set of filters designed to identify specific primitive features such as spots, edges and ripples in a local region. The origin of the Laws' filters are three vectors which correspond to an approximate Gaussian and its first two derivatives.

$$L_3 = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \tag{1}$$

$$E_3 = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \tag{2}$$

$$S_3 = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \tag{3}$$

Convolving these vectors with themselves and one another generates five 5x1 vectors with the mnemonics Level, Edge, Spot, Wave, and Ripple, which indicate which type of feature they represent. Multiplying these five vectors with themselves and one another produces a set of 25 unique 5x5 masks. Twenty of these masks are transposes of one other mask.

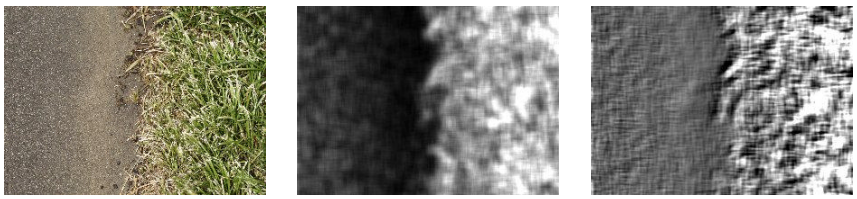

Figure 3: Two of Laws' filters applied to a pair of textures, corresponding to bands 8 (middle) and 9 (right)

Convolving these masks with an image produces a texture plane for each one. We then combine the results of the masks that are transposes of one another and divide all of the resulting planes by the LL mask, which corresponds to average intensity. The result is 14 texture planes, each of which represents the magnitude of a given texture signal. Figure 3 shows an example of two of the texture planes from a real textured image.

In order to normalize the results of each texture filter, we give each texture plane T_i equal statistical significance by normalizing by the mean $\bar{T}_i$ and standard deviation σ_i.

$$\hat{T}_i(x,y) = \frac{T_i(x,y) - \hat{T}_i}{\sigma_i} \tag{4}$$

We then restrict the range of values to 4 standard deviations and scale the values to a set range such as $[0,255]$. In addition to the 14 Laws texture measures, we also have the option of including the two rg chromaticity bands, also appropriately normalized. Chromaticity (r,g) is defined as in (4), where R, G, and B are the raw sensor values. The normalized texture image then gets passed to the compass operator.

$$\langle r,g \rangle = \frac{\langle R,G \rangle}{R+G+B} \tag{5}$$

2.1 Compass operator

The compass operator is based on comparing the two halves of an oriented circle at a variety of orientations for each pixel in an image [7]. When applying the compass operator to a texture image, however, each pixel represents an area the size of the underlying filter for a given plane. Therefore, we must take care to avoid using pixels who's underlying filters cross the central boundary of the compass operator. If the compass operator happens to be sitting on an edge, for example, then those pixels close to the edge will be representing a mixed or transitional distribution of texture features rather than one of the distributions on either side of the edge. Therefore, if the radius of the underlying filter is f, then when calculating the histograms of distributions for each side of the compass operator we ignore all pixels within f of the central line.

Once the compass operator has built the two distributions, we need to calculate the magnitude of the difference between them $d(H_1,H_2)$. Our distance measure is dynamic time warping [DTW], which is a dynamic programming approach to finding the optimal correspondence between two vectors [3]. While originally designed for comparing signals in time, it is equally applicable to comparing 1D histograms. DTW is more flexible than the unfolding method of Shen and Wong and less computationally intensive than EMD [9].

At each orientation of the compass, for each plane in the texture image, we use DTW to calculate the distance between the two halves of the compass. We then store the maximal distance and orientation for each plane. The post-processing step uses this information to identify likely edges in the image.

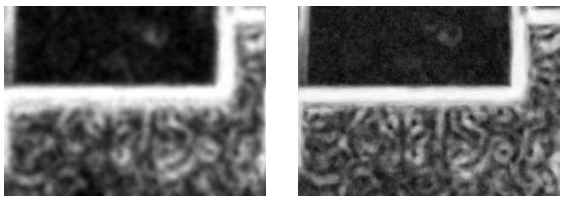

Figure 4: Raw output of the compass operator with no jitter (left) and with jittered sampling in a 4x4 pixel box (right).

Since the compass operator is expensive to compute at each pixel we take a sampling approach to estimating the texture gradient in the image. Following a practice commonly used in ray tracing, we sample the image with the compass operator using a jittering approach. For all of the images shown in this paper we apply the compass operator once on a randomly selected pixel in each 4x4 pixel block. This result is then smoothed out using a 4-connected box filter that ignores any unset pixels. The resulting dense texture gradient estimate is used for the post-processing. Figure 4 shows the raw output of the compass operator for both jittered and unjittered cases.

It is important to note that jittered sampling at a given resolution is not equivalent to applying the compass operator at a lower resolution. First, the underlying texture operators would respond differently to differently scaled textures. Second, sampling on a regular grid has the potential for both aliasing and missing edges, both of which jittering helps avoid.

2.2 Post-processing

To extract edges from the output of the compass operator we first calculate the average magnitude of the texture plane gradients. We then threshold at 75% of the maximum and thin the resulting edges, which tend to be thick at real texture boundaries. The thinning algorithm (Matlab) leaves 8-connected lines and does not break existing lines. Figure 5 shows the complete process for one of the example images.

We also tried using a covariance-weighted average, Mahalanobis distance, to generate the final magnitude image, hoping that this would catch any odd relationships between the Laws' filters. The Mahalanobis distance actually made the results worse by reducing the differences between edge and non-edge areas.

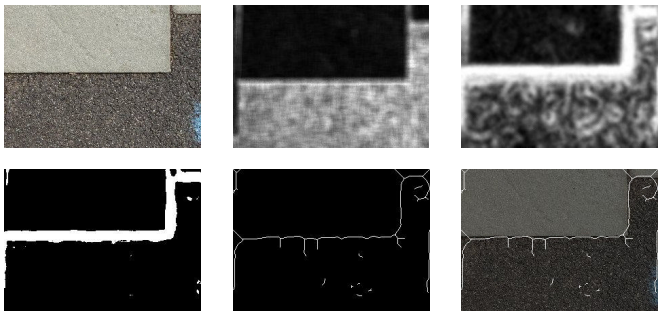

Figure 5: Complete process: original image (top-left), normalized filter output with 14 features (top-center), compass operator output (top-right), thresholded output (bottom-left), thinned result (bottom-center), edges overlaid on original (bottom-right).

3 Experiments and Results

We tested a number of different algorithms on a set of both synthetic (two) and real images (eight included herein). The real images–mostly outdoor scenes–were taken by undergraduates as part of a computer vision course. The students took the images using an

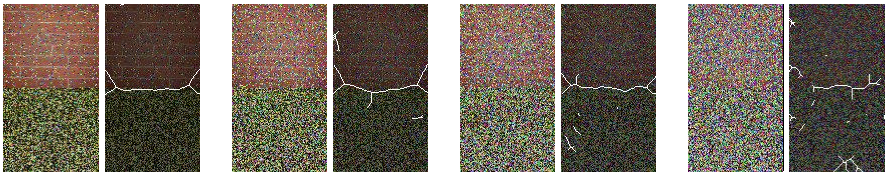

Figure 6: A series showing the effect on the compass operator of uniformly distributed random noise added to a real texture image. From left to right, the images contain 17%, 34%, 53%, and 71% noise, respectively.

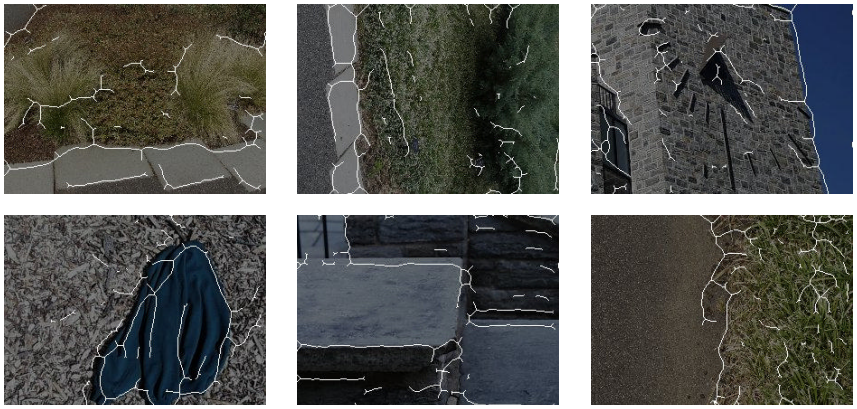

Figure 7: Results on real images using 14 Laws' texture energy filters, a radius 10 compass operator examining 4 directions and DTW per plane to compare distributions. All images were thresholded at 75% of their maximum value prior to thinning.

Olympus D600-L digital camera in standard mode (640x480 with significant jpeg compression). The outdoor images shown in the results were reduced to 320x240 prior to processing to both minimize computation and reduce the scale of the texture to be more appropriate for the Laws' filter size (5x5).

The first experiment was a simple test of the robustness of the operator to uniform random noise injected into a synthetic picture of two real textures. Figure 6 shows the results of injecting increasing amounts of noise into the image. Even the extreme case still shows a maximum difference along the boundary. This is one of the benefits of using distributions over a large area.

The second experiment looked at the performance of the operator on a variety of real images using only Laws' texture energy filters and no chromatic information. The results in Figure 7 show that the operator finds the major texture boundaries as well as some minor texture boundaries caused by shadows or variations in the texture patterns. In particular, note the strong boundaries between the stones and the plants and between the stones and the asphalt in the top two images without the use of chromatic information.

Figure 8 shows the results of a third experiment, where we included the two rg chromaticity bands along with the 14 texture filters. The covariance matrix of these images also shows a strong positive correlation between the rg chromaticity and the other texture energy bands, but it was generally less than the correlations between the texture filters

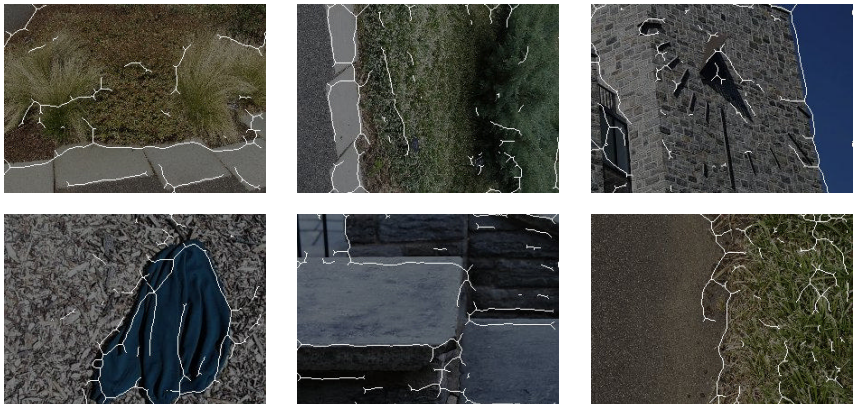

Figure 8: Results using 14 Laws texture energy filters plus rg chromaticity, a radius 10 compass operator with 4 directions and DTW per plane. The resulting were thresholded at 75% of their maximum value and then thinned.

Figure 9: The left image shows edges found using the compass operator with 14 Laws' filters, radius 10, at 4 orientations with DTW/plane. The resulting magnitude image was thresholded at 75% of its maximum and then thinned. The right image shows edges found using a standard edge operator (built into the Unix program xv) and thresholded at 50% of the maximum edge magnitude value.

themselves. This makes sense, because some of the textures involve significant chromaticity changes within texturally homogeneous regions.

While applying the compass operator is computationally expensive as compared to a virtually instantaneous standard edge detector, the vast improvement in the corresponding results justifies the extra computations. Figure 9 shows one of our test images and the resulting edges found by the compass operator and the edges found by the edge detector in xv. The result was manually-thresholded to try to balance the important edges and clutter. Clearly, the standard edge detector has good localization in edges that it finds, but the edges are largely broken up and incoherent. The texture extended compass operator, however, finds the major boundaries between textures in the image. Note, especially, the clean, strong edges along the grass on the left side and top of the image.

As a final experiment, we directly compared the earth mover's distance and dynamic time warping approaches and looked at their results on a set of texture images. Note that we did not execute the vector quantization or saturated metric used by Ruzon and Tomasi, but merely substituted EMD for DTW within our overall algorithm [8]. Arguably, these characteristics could improve both metrics similarly. We used the EMD code available from Yossi Rubner to ensure that we had a correct implementation [5].

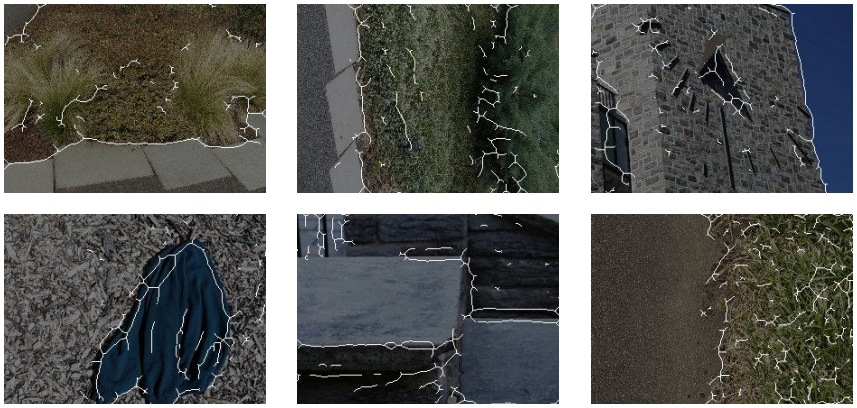

Figure 10: Results using 14 Laws' texture energy filters, a radius 10 compass operator examining 4 directions and EMD to compare distributions. All images were thresholded at 75% of their maximum value prior to thinning.

The overall results on the natural scenes are similar to the DTW per plane. Also, as can be seen in the top two images in Figure 10 the EMD results lose the boundary between the stone and asphalt. These edges are quite strong in Figure 7 and Figure 8.

The computational cost of the per-pixel EMD metric was more than 50 times greater than the jittered DTW per plane approach for a 320x240 image using 14 texture planes (30 minutes versus 30s) when run on a 1.8MHz Athlon. If we also used jittering with EMD, the DTW algorithm would still beat EMD by a factor of 6. The one drawback of the DTW per plane approach is that it scales with the number of planes, while EMD only scales with compass and image size. DTW, however, scales as $O(N^2)$ where N is the number of pixels within the compass operator, while EMD scales as $O(N^3)$. So the difference will be more pronounced for larger compass sizes.

4 Discussion and Conclusions

The results appear to support the following claims. First, the compass operator is a useful post-process for standard texture filters that highlights natural texture edges in an image. The fact that the compass operator output for each of the 14 Laws' texture planes are highly correlated suggests that it can improve texture boundary detection for a variety of statistical and frequency-based texture filters.

Second, the use of a DTW-based histogram comparison method that works on a per-plane basis provides a significant computational improvement over EMD while still producing comparable results. Since the computational complexity of the DTW method is sensitive to the number of texture planes, the computational speedup for texture images with fewer planes is actually larger than the factor of 6 found for the 14 texture plane case. For texture images with only three planes, such as an RGB image, the speedup is on the order of 30. This makes it feasible to do near real time experimentation with larger images and a variety of parameters.

There are still a number of avenues to explore in applying the compass operator to textured images. First, it would be useful to examine the use of alternative distance met-

rics within the texture histogram space. The current implementation uses the difference in normalized magnitudes to assign distances in the DTW comparison. The saturating distance metric used by Ruzon and Tomasi for color images is a possible alternative [8].

Second, experimentation with multiple image scales suggests that a pyramid scheme for detecting texture edges would be a useful approach. However, initial experiments with weighted averages of the outputs from different scales did not produce results significantly different from the single image implementation, indicating that this is not a trivial modification. Thus, while it seems appropriate, a pyramid based approach will have to be carefully thought out to significantly improve the results.

Overall, these results demonstrate the value of the compass operator in the analysis of complex images. Its fundamental characteristic, the comparison of oriented neighboring area distributions, makes it a useful operator for the detection of texture edges in natural scenes.

References

[1] K. I. Laws. Texture energy measures. In *DARPA Image Understanding Workshop*, pages 47–51, 1979.

[2] Werman M., S. Peleg, and A. Rosenfeld. A distance metric for multidimensional histograms. *Computer Vision, Graphics and Image Processing*, 32(3):328–336, 1985.

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

[4] T. Randen and J. H. Husoy. Filtering for texture classification: A comparative study. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(4):291–310, 1999.

[5] Y. Rubner. http://robotics.stanford.edu/ rubner, 2002.

[6] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of 1998 IEEE International Conference on Computer Vision*, pages 56–66, January 1998.

[7] M. A. Ruzon and C. Tomasi. Color edge detection with the compass operator. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 160–166, June 1999.

[8] M. A. Ruzon and C. Tomasi. Edge, junction, and corner detection using color distributions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(11):1281–1295, 2001.

[9] H. C. Shen and A. K. C. Wong. Generalized texture representation and metric. *Computer Vision, Graphics, and Image Processing*, 23(2):187–206, 1983.

[10] M. Swain and D. Ballard. Color indexing. *Int'l Journal of Computer Vision*, 7(1):11–32, 1991.