Texture Segmentation and Shape in the Same Image

John Krumm
Intelligent Systems and Robotics Center
Sandia National Laboratory
Albuquerque, NM
jckrumm@sandia.gov

Steven A. Shafer
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA
sas@cmu.edu

Abstract

Uniformly textured surfaces in 3D scenes provide important cues for image understanding. Texture can be used for both segmentation and for 3D shape inference. Unfortunately, virtually all current algorithms are based on assumptions that make it impossible to do texture segmentation and shape-from-texture in the *same* image. Texture segmentation algorithms rely on an absence of 3D effects that tend to distort the texture. Shape-from-texture algorithms depend on these effects, relying instead on the texture being already segmented. To really understand texture in images, texture segmentation and shape-from-texture must be viewed as a combined problem to be solved simultaneously. We present a solution to this problem with a region-growing algorithm that explicitly accounts for perspective distortions of otherwise uniform texture. We use the image spectrogram to compute local surface normals, which are in turn used to "frontalize" the texture. These frontalized texture patches are then subjected to a region-growing algorithm based on similarity in the local frequency domain and a minimum description length criteria. We show results of our algorithm on real texture images taken in the lab and outdoors.

1 The Problem and What We're Doing About It

We can tell a lot about a scene from a single, monocular image of it. Part of this understanding comes from uniformly textured objects in the scene. The texture can be used to segment the object and to infer its shape. Computer vision researchers have made significant progress toward exploiting both of these cues. Texture segmentation has a rich history documented in survey articles[5][14][16]. The fundamental idea is to map the image into a compact representation (*e.g.* gray-level statistics) of the texture whose parameters remain constant over the texture region. Algorithms search this 2D map of texture parameters for sharp changes (edge-finding) or uniform regions (region-growing) to accomplish a segmentation.

Shape-from-texture has an equally rich history. As in texture segmentation, the goal is to find the appropriate representation. Here, however, the representation must be sensitive in some coherent way to 3D effects. Formalizing the connection between the representation and the surface normal of the texture gives the foundation for a shape-from-texture algorithm.

In spite of the clever mathematics and high-powered algorithms focused on texture understanding, there remains an obvious gap in applying the research to realistic, uncontrived images. Past algorithms for texture segmentation were limited to textures that appear uniform in the image. While this is reasonable for aerial images, orthographic projection, and certain contrived laboratory or industrial situations, it is an unrealistic assumption for general images of the real world. Past shape-from-texture algorithms have all relied on pre-segmented images, because shape-induced texture deformations could not be distinguished from changes due to completely different objects.

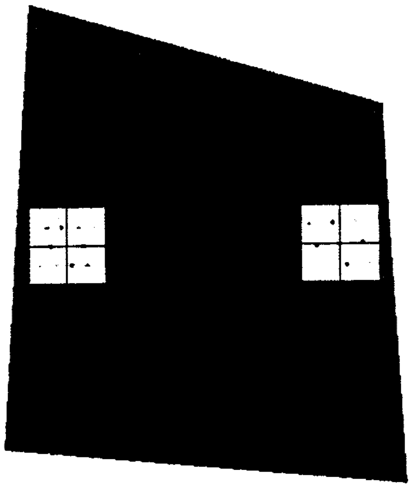


Figure 1: Part of the image spectrogram is shown as the two, light-colored patches. These are the Fourier power spectra of the underlying pixels. Since the texture is periodic, the power spectra show sharp peaks corresponding to 2D fundamental and harmonic frequencies. The frequencies in the right patch are higher (farther from frequency origin) than the left patch, because perspective effects make the texture elements appear

Our solution is to explicitly account for the texture deformation due to 3D perspective effects. Our algorithm works by first estimating the surface normal of small patches in the image. We make this estimate with our own spectrogram-based shape-from-texture algorithm. The algorithm we use works only on planar, periodic textures, so we are limited in what type of textures the image can contain. Based on these surface normal estimates, we "frontalize" the local spatial frequency content of the texture. This step undoes the effect of 3D perspective, and allows us to directly compare adjacent patches for matching. We merge adjacent patches based on a minimum description length criteria to produce the final

0-8186-7042-8/95 $4.00 © 1995 IEEE

segmentation. As a by-product of accounting for shape effects, we also get the surface normals of the textured regions.

Research in shape-from-texture using spatial frequency started with Bajcsy and Leiberman[1] in 1976. Those ideas have been mathematically formalized and tested by Jau and Chin[7], Brown and Shvaytser[3], Super and Bovik[13], Krumm and Shafer[8], and Malik and Rosenholtz[10].

We explain and evaluate our spectrogram-based shape-from-texture algorithm in Section 2. Section 3 shows how we incorporate the shape results into a region-growing algorithm, and includes results on simulated, laboratory and outdoor images.

2 Local Shape-from-Texture Using the Spectrogram

In order to undo the effects of 3D perspective, we need to find the surface normals of textured objects. Our shape-from-texture algorithm is based on the image spectrogram, which is a series of local Fourier power spectra computed at different locations in the image. Mathematically, if the image is given by $f(x, y)$, then the spectrogram is

$$S(x, y, u, v) = \left| \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} w(x', y') f(x' - x, y' - y) e^{-j2\pi(ux' + vy')} dx'dy' \right|^2 \qquad (1)$$

where $w(x, y)$ is the window function and (u, v) are the frequency coordinates in cycles/pixel. Part of a spectrogram taken on a periodically textured plate is shown in Figure 1. The two light-colored patches are local power spectra of the underlying pixels. Our particular window function is the "Blackman-Harris minimum 4-sample" window, recommended by experts[6] for Fourier analysis. Its equation is

$$w(l) = w_0 + w_1 \cos\left(\frac{2\pi}{L-1} l\right) + w_2 \cos\left(\frac{4\pi}{L-1} l\right) + w_3 \cos\left(\frac{6\pi}{L-1} l\right) \qquad (2)$$

where L is the (even integer) width of the window in pixels, $0 \le l \le (L-1)/2$, and $l = \sqrt{x^2 + y^2}$. The coefficients are $(w_0, w_1, w_2, w_3) = (0.35875, 0.48829, 0.14128, 0.01168)$. For our analysis, we let $L = 64$ pixels.

2.1 Perspectively Projected Texture

These local power spectra show frequency shifts resulting from 3D perspective effects. The two power spectra patches in Figure 1 show this shift. Since the underlying texture is periodic, the power spectra consist of delta functions. The delta functions in the right patch are farther from the origin because that part of the textured plate is farther away from the camera. We showed in [8] that the relationship between the frequencies in the two patches is given by an affine transformation:

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} \qquad (3)$$

where (u_1, v_1) and (u_2, v_2) are corresponding frequency components from the two patches, and

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} = B \begin{bmatrix} d - px_1 - qy_2 & p(y_2 - y_1) \\ q(x_2 - x_1) & d - px_2 - qy_1 \end{bmatrix} \qquad (4)$$

with

$$B = \frac{d - px_1 - qy_1}{(px_2 + qy_2 - d)^2}$$

and (x_1, y_1) and (x_2, y_2) are the two points on the image plane being compared, d is the camera's pinhole-to-sensor distance, and (p_1, q_1) and (p_2, q_2) are gradient space variables representing the 3D surface normals of the two surface patches. We derived this relationship by perspectively projecting two texture patches onto the image plane and then linearizing this projection with a Taylor series.

We conclude that the frequencies of a single sinusoid projected from the same plane to two different points in the image are approximately related by an affine transformation. The affine parameters are functions of the position of the two points on the image, the camera's pinhole-to-sensor distance, and the plane's surface normal.

2.2 Periodic Texture Representation

We compute the spectrogram of the image by computing local power spectra at equally spaced centers on the image. The windows for the power spectra are allowed to overlap. This subsection explains our compact representation of the spectrogram that we use for subsequent shape and segmentation routines.

If we assume the texture on the plane is periodic, then any physically realizable such texture can be represented by a Fourier series. Thus, we assume the frontally viewed texture brightness pattern is given by the Fourier series

$$g(s, t) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} c_{mn} \exp[j2\pi(mu_0 s + nv_0 t)] \qquad (5)$$

where we are unconcerned with the values of the fundamental frequency (u_0, v_0) and the complex Fourier series coefficients c_{mn}. Using this definition of the Fourier transform,

$$F(u, v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux + vy)} dxdy \qquad (6)$$

the Fourier transform of the frontally-viewed texture $g(s, t)$ is

$$G(u, v) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} c_{mn} \delta(u - mu_0, v - nv_0) \quad (7)$$

This is a grid of delta functions, with each delta at one component frequency. These delta's are apparent in the two power spectrum patches in Figure 1.

In order to represent the spectrogram more efficiently and to speed subsequent computations, we only store the peak frequencies from each power spectrum patch. Our spectrogram preprocessor finds the peaks in each patch in order of size. It keeps looking until the current peak is less than 20% of the magnitude of the largest peak, or until it finds six peaks, whichever comes first. It also ignores peaks below a frequency of 0.03 cycles/pixel. This helps eliminate low frequencies due to shading.

In order to track frequency shifts for computing surface normals, we need to know which peaks in one patch correspond to those in neighboring patches. Our preprocessor matches peaks between every patch and its two adjacent patches to the right and below. We do this pairwise matching by considering every possible match combination between the two sets of peaks, including leaving some peaks unmatched. We pick the combination that has simultaneously the most matches and no match errors that exceed a threshold based on the largest surface normal we expect in the scene. For a maximum (p, q) of $(1.5, 1.5)$, this threshold prevents matching peaks that are more than about 0.05 cycles/pixel apart for our imaging configuration. After this preprocessing step we do not need the original spectrogram for any of the subsequent operations. It is adequately represented by the peaks and peak matches.

2.3 Computing Local Surface Normals

We compute local surface normals by finding the (p, q) that best accounts for the observed frequency shifts between neighboring power spectrum patches. At its most basic, this computation involves just two adjacent patches centered at (x_1, y_1) and (x_2, y_2) on the image plane. The sets of m matching peaks from the two patches are (u_{11}, v_{11}), (u_{12}, v_{12}), (u_{13}, v_{13}), ... (u_{1m}, v_{1m}) and (u_{21}, v_{21}), (u_{22}, v_{22}), (u_{23}, v_{23}), ... (u_{2m}, v_{2m}). If we write the affine parameters from Equation (4) as functions of the surface normal, we have

$$e_{ssd}(p, q) = \sum_{i=1}^{m} \left\| \begin{bmatrix} u_{2i} \\ v_{2i} \end{bmatrix} - \begin{bmatrix} a_1(p, q) & b_1(p, q) \\ a_2(p, q) & b_2(p, q) \end{bmatrix} \begin{bmatrix} u_{1i} \\ v_{1i} \end{bmatrix} \right\|^2 \quad (8)$$

This will be small if we have the correct surface normal and the correct matches among the peaks. We perform an exhaustive search over a grid in (p, q) and take the surface normal that minimizes e_{ssd} as the solution. If we have more than two patches to use, we find the surface normal that minimizes the sum of the e_{ssd}'s for all unique, adjacent pairs of patches in the region. We only consider adjacent pairs of patches, that is, the patches that have had their frequency peaks matched by the preprocessor. This algorithm is similar to one developed by Super and Bovik[12]. One difference is that ours uses multiple frequency peaks from a single texture, while theirs uses a single, dominant frequency at each point.

2.4 Results of Shape-from-Texture

Two important parameters that affect the accuracy of our solution are the number of patches used to compute the surface normal and the center-to-center spacing of the power spectrum patches. For a given center-to-center spacing, we would like to use as many patches as possible, as long as they all fall on the same textured plane, in order to have more data contributing to the solution. We would also like to avoid small center-to-center distances, because the shape-induced frequency shifts could then be dominated by noise and approximation errors.

Figure 2 shows four identical plates with different Brodatz[2] textures mapped onto them using a computer graphics program. The actual surface normal is $(p, q) = (0.614, 0.364)$. We tested our algorithm on these images using different numbers of patches and different center-to-center spacing. In each trial, the center-to-center spacing of the power spectra was equal in x and y. We let this parameter vary from 5 to 50 pixels in increments of 5. For each center-to-center distance, we computed (p, q) using as many unique $n \times n$ squares of adjacent patches as would fit on the textured part of the image, starting with $n = 2$.

We computed the average errors in degrees of our surface normal estimates for different numbers of patches and different center-to-center spacings. Each average was taken over all four images and over all the $n \times n$ squares of patches that would fit on the texture. As expected, the error decreases for larger numbers of widely spaced patches, with the best estimates being in error by about six degrees. Our shape-from-texture algorithm thus succeeds in giving good results on periodic textures without the need for image feature detection.

Unfortunately the need for accuracy conflicts with the requirements of our segmentation algorithm in terms of the number of patches and center-to-center spacing. Our segmentation algorithm begins by estimating surface normals using small parts of the image. Using small support for these estimates is important, because we do not want the support to overlap texture boundaries. This means we have to keep n and the center-to-center spacing small, which tends to compromise accuracy. Fortunately, though, some of the estimates from the $n \times n$ squares are still good, even with small support and small n. We computed the average minimum error in surface normal, where the minimum is taken over all the $n \times n$ squares and the average is taken over the four images. In almost every case, at least one of the $n \times n$ squares gave a fairly accurate surface normal (to within about five degrees). Since we start our segmentation with many possible seed regions, we are likely to have some that are "good", even with small support. For the segmentation algorithm discussed in the next section, we chose a center-to-center spacing of 15 pixels, and we start our local shape computations with only two patches. Since we do not allow interleaved regions, we computed the spectrogram with the same center-to-center spacing.

3 Segmenting Textured 3D Surfaces

Our segmentation procedure is a region-growing algorithm that merges regions based on similarities in their local power spectra. The problem with applying such an algorithm naively to an image of 3D textured surfaces is that the power spectra on identically textured surfaces will be different due to 3D effects. And while a generous tolerance may still allow such regions to be merged, this may well allow different textures to be merged also. Thus, we need to explicitly account for the 3D effects. We do this by computing the surface normal of each region (using the algorithm in the previous section) and then "frontalizing" the frequencies to show what the power spectra of the texture would be if viewed from the front. If adjacent regions have similar frontalized frequency content, they are merged. A detailed description of the segmentation algorithm follows.

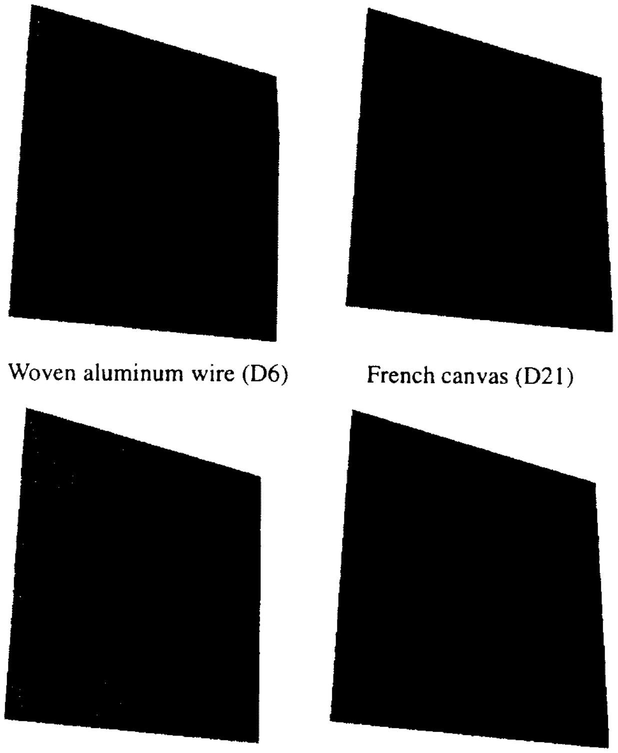


Figure 2: Images used for testing surface normal computation. These are all from the Brodatz[4] book of textures, and the book's designations are given in parentheses.

3.1 The Data Structures

The smallest elements of our image representation are the power spectrum patches, represented by their peaks. Since we segment based on 4-connectedness, each patch has a list of its 4-connected neighbors. Each patch also contains the indices of the matched peaks in the patches to the right and the patch below.

Sets of merged patches are called hypotheses. In the beginning, there is one hypothesis for each patch. Each hypothesis contains the usual records needed for region growing, *i.e.* the constituent patches and neighboring hypothesis. We also use the constituent patches to compute the surface normal using our shape-from-texture algorithm. This surface normal is used to compute a frontalized version of the frequency peaks for each constituent patch. The surface normal is also used to compute frontalized versions of the patches in the four-connected neighboring hypotheses. If two hypotheses are from the same texture on the same plane, they will have similar frontalized hypothesis.

3.2 Frontalization of Frequency Peaks

In order to assess the similarity of two sets of frequency peaks, we need to compare them as if they had the same depth and surface normal. We use our local surface normal estimates to "frontalize" the frequency peaks from power spectrum patches that we want to compare.

Mathematically, we know that perspective projection causes an approximately affine transformation on the frequency peaks[8]. This transformation is a function of the textured surface's position in space and its surface normal. Since frontalization is essentially an undoing of the perspective effects, then frontalizing the peaks is also an affine transformation based on the same variables. We know everything for the inverse transformation except for the depth. We work around this by doing a relative frontalization on a a patch with respect to another patch. For a group of patches that we want to frontalize, we pick one as the reference patch and frontalize the others to have the same depth as the reference. If (u, v) is a frequency point on a non-frontalized patch, the frontalized version is

$$\begin{bmatrix} u_{frontal} \\ v_{frontal} \end{bmatrix} = F \begin{bmatrix} u \\ v \end{bmatrix} \quad (9)$$

The elements of F are

$$\begin{aligned} f_{11} &= D\,[d\,(p^2 + rq^2) - px\,(p^2 + q^2)\,] \\ f_{12} &= Dp\,[dq\,(1 - r) - y\,(p^2 + q^2)\,] \\ f_{21} &= Dq\,[dp\,(1 - r) - x\,(p^2 + q^2)\,] \\ f_{22} &= D\,[d\,(rp^2 + q^2) - qy\,(p^2 + q^2)\,] \end{aligned} \qquad (10)$$

where

$$D = \frac{(px + qy - d)}{dr\,(p^2 + q^2)\,(px_{ref} + qy_{ref} - d)}$$

The reference patch is centered at coordinates (x_{ref}, y_{ref}) in the image, and the patch to be frontalized is at (x, y). The two patches are assumed to have the same surface normal (p, q), and $r = \sqrt{p^2 + q^2 + 1}$. A detailed derivation of these equations is in [8].

The frontalization step works this way: For a group of patches hypothesized to be on the same plane, we arbitrarily pick one patch as the reference patch. In our case we pick the first in the list. The affine frontalization transformation is then computed for each patch according to Equation (9), and each peak frequency is transformed accordingly. This does not tell us what the true frontalized frequencies are, but it tells us what the frequencies would be if all the patches had the same depth as the reference patch, which is good enough for segmentation.

3.3 Creating a Dendrogram of Hypotheses

Our segmentation algorithm consists of building a dendrogram of the image. A dendrogram, defined by Duda and Hart[4], is a hierarchical clustering represented by a tree with individual feature vectors as the leaves. In our situation, each power spectrum patch is a feature vector. At the beginning, each feature vector is its own cluster. We can build a dendrogram by successively merging the two clusters that are most similar (by some measure) into a new cluster. Different levels of the dendrogram represent different clusterings. In our algorithm, clusters correspond to hypotheses. After each merge, the number of hypotheses is reduced by one. We only allow merges between four-connected hypotheses, and at all times each patch is a member of only one hypothesis. Different levels of our dendrogram correspond to different segmentations of the images. We show selected levels of a dendrogram in the middle left of in Figure 3, where each of the 25 subimages is a little version of the region map with merged patches shaded the same. The first level has every patch as a separate region, while the last level has every patch in the same region. For Figure 3, the best segmentation occurs when there are four regions.

We build our dendrograms by successively merging the adjacent hypotheses that give the maximum decrease in description length. The overall description length for a hypothesis consists of two parts that are added together: the description length to describe the hypothesis' region in the image and the description length to describe the hypothesis' texture. The description length for the region is from Leclerc's[9] work on gray-level image segmentation. He argues that the description length of an image region is equal to the number of bits it takes to describe the chain code of unit-length line segments of the region's boundary. He shows that this is approximately equal to a constant b times the length of the boundary. For four-connected regions, Leclerc recommends that b be between $\log_2 3$ and two. We found that $b = 2$ works fine for us. This part of the description length, then, favors merging hypotheses with longer common boundaries.

The second part of the description length of a hypothesis concerns the texture itself. The texture of each hypothesis is represented by the frontalized frequency peaks of the region's constituent patches. If these peaks fall into a few, tight clusters, they probably came from the same texture. Given the frequency peaks from all the patches in a region, we take their description length as the description length of the best possible clustering of the peaks. We find this clustering by building another dendrogram of the peaks, whose merge criterion is simple Euclidean distance in frequency. Each level of the dendrogram represents a possible clustering of the peaks, and we take the level with the minimum description length. This is based on an idea of R. Wallace's [15]. The formula that we use to compute this description length is based on the assumption that the peaks are Gaussian distributed around the cluster centers. It is

$$n_p \log_2(n_c) + 2\sum_{i=1}^{n_c} \log_2(n_i) + \frac{1}{2}\log_2(e) \sum_{i=1}^{n_c}\sum_{j=1}^{n_i}\left[\left(\frac{u_{ij} - \bar{u}_i}{\sigma}\right)^2 + \left(\frac{v_{ij} - \bar{v}_i}{\sigma}\right)^2\right] \qquad (11)$$

where

n_p = number of peaks

n_c = number of clusters

n_i = number of peaks in cluster i

(u_{ij}, v_{ij}) = peak j of cluster i

$\left(\bar{u}_i, \bar{v}_i\right)$ = mean of peaks in cluster i

σ = standard deviation of peaks

The first addend of this formula gives the number of bits it takes to label each peak with an integer giving its cluster number. The second addend is Rissanen's[11] estimate of the number of bits to describe the mean and standard deviation vectors of each cluster. The third addend is the number of bits it takes to describe the deviation of the peaks from their cluster centers. It is based on a Gaussian probability distribution, and is essentially $-\log_2 P(\text{peaks}|\mu, \sigma)$. The standard deviation for this term could be computed from the clusters themselves, however small clusters will give unstable values. We set the standard deviation to 0.01 for our experiments. This value is reasonable, since the Nyquist-limited range of frequencies is [-0.5,0.5].

The dendrogram of hypotheses is built this way: At the beginning, each patch is its own hypothesis. To compute the next level in the dendrogram, all pairs of 4-connected hypotheses are temporarily merged into new, trial hypotheses. Each trial merge involves computing the surface normal of the merged region, frontalizing the peaks, and building a dendrogram of the frontalized peaks. We take the merge that gives the largest reduction in description length. The "magic numbers" are Leclerc's value of b for the chain-code description length, the standard deviation of the peaks, and the level in the dendrogram of hypotheses to choose as the final segmentation. The first two parameters are easy to choose. The final dendrogram level is more difficult. As our algorithm stands now, we must choose this manually, although some measure based on description length would be appropriate here.

After picking the dendrogram level, we refine the edges by shifting the region affiliation of power spectrum patches along the region borders. If the switch lowers the total description length, we reassign the patch, otherwise we keep it as it was.

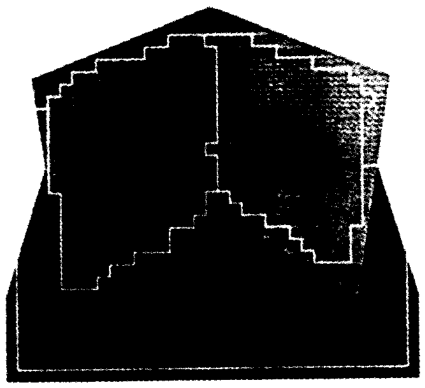
3.4 Results

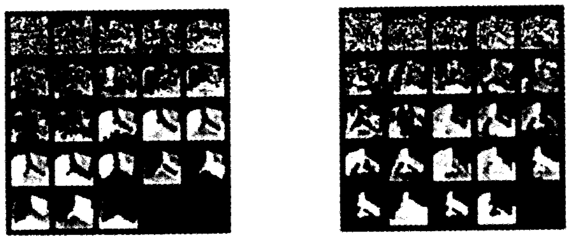
The results of our segmentation algorithm on three different images are shown in Figure 3 - Figure 5. Figure 3 shows the segmentation of three Brodatz[2] textures mapped to three planes. Selected levels of the dendrogram are shown below the image, where each subimage of the dendrogram is shaded to show the regions. The bottom of the figure is a needle diagram of the surface normals computed in our segmentation algorithm. In this case, the average error was $6.71°$. Next to the left dendrogram is the dendrogram we got by ignoring the surface normals. We created this dendrogram by changing our surface normal subroutine so that it would always return $(p, q) = (0, 0)$. This meant that the frontalization step would not alter the peak positions. There are no good segmentations in this dendrogram, which highlights the importance of considering the surface normals. We got similarly bad results from ignoring surface normals on all the images we tested.

A textured screen guard and some untextured objects are shown in Figure 4. This image was taken in our lab. Our algorithm groups the untextured parts together, since they are the same from a texture point of view. We measured the actual angle of the screen guard with an electronic level. The average surface normal error for the screen guard was $6.21°$. Figure 5 shows a large apartment building whose windows make a rectangular grid.

4 Conclusions

Our algorithm fills a gap in automatic image understanding. Until now, combining texture segmentation and shape-from-texture in the same image was impossible, due to conflicting assumptions for the two types of algorithms. Our algorithm explicitly accounts for shape effects during segmentation. We compute local surface normals with our own shape-from-texture algorithm that requires no texel detec-


Refined edges based on four-region level of dendrogram


Dendrograms with (left) and without (right) considering surface normals. Ignoring surface normals gives no good segmentations in the dendrogram.

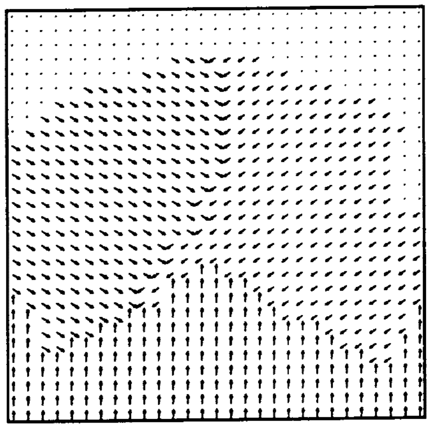
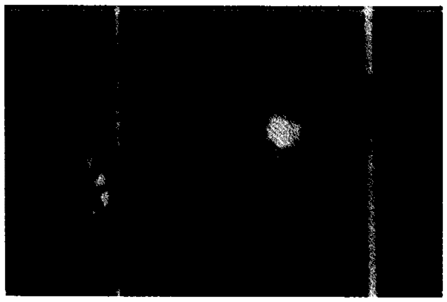

Computed surface normals, average error is 6.71 degrees

Figure 3: Synthetic image with Brodatz textures "woven aluminum wire" (D6), "netting" (D34), and "cotton canvas" (D77). Edge refinement gives a noticeable improvement. Ignoring the surface normals gives a dendrogram with no good segmentations.


Edges from three-region level of dendrogram

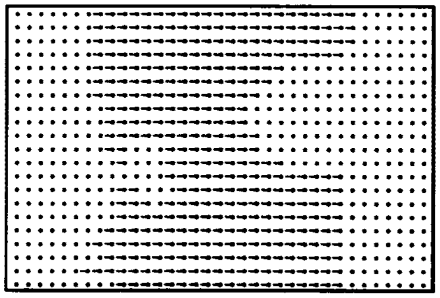
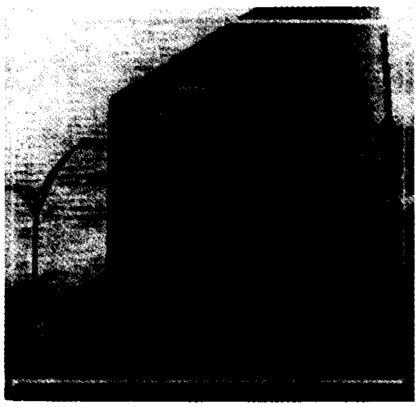

Computed surface normals, average error is 6.21 degrees

Figure 4: Laboratory image of screen guard and other objects. Our program can distinguish the textured and untextured regions of an image.


Refined edges based on four-region level of dendrogram

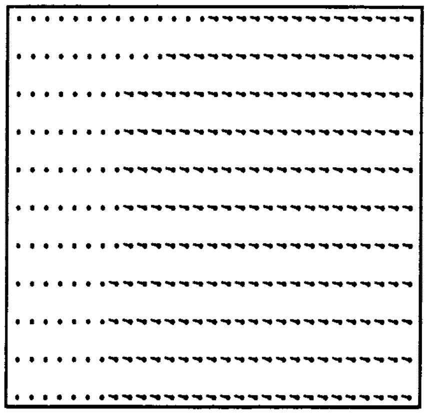

Computed surface normals

Figure 5: This is a large apartment building. Our program separates the texture from the non-texture.

tion. These surface normals allow us to undo the effects of shape. We apply a region-growing algorithm based on description length to the "frontalized" texture. The description length merge criteria provides a fairly robust and simple way of arbitrating among possible merges. To our knowledge, this is the first algorithm that can segment nonfrontal textures by explicitly accounting for shape.

References

[1] Bajcsy, Ruzena and Lawrence Lieberman. "Texture Gradient as a Depth Cue." *Computer Graphics and Image Processing* 5 (1976): 52-67.

[2] Brodatz, Phil. *Textures: A Photographic Album for Artists and Designers*, New York: Dover, 1966.

[3] Brown, Lisa Gottesfeld and Haim Shvaytser. "Surface Orientation from Projective Foreshortening of Isotropic Texture Autocorrelation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (June 1990): 584-588.

[4] Duda, Richard O. and Peter E. Hart. Pattern Classification and Scene Analysis. New York: John Wiley & Sons, 1973, p. 228.

[5] Haralick, Robert M. "Statistical and Structural Approaches to Texture." Proceedings of the IEEE 67 (no. 5, May 1979): 786-804.

[6] Harris, Fredric, J. "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform." *Proceedings of the IEEE* 66 (January 1978): 51-83.

[7] Jau, Jack Y. and Roland T. Chin. "Shape from Texture Using the Wigner Distribution." *Computer Vision, Graphics, and Image Processing* 52 (1990): 248-263.

[8] Krumm, John. "Space/Frequency Shape Inference and Segmentation of 3D Textured Surfaces" (Ph.D. Thesis). Carnegie Mellon University Robotics Institute Technical Report No. CMU-RI-TR-93-32, December 1993.

[9] Leclerc, Yvan G. "Constructing Simple Stable Descriptions for Image Partitioning." *International Journal of Computer Vision* 3 (1989): 73-102.

[10] Malik, Jitendra and Ruth Rosenholtz. "A Differential Method for Computing Local Shape-From-Texture for Planar and Curved Surfaces." Computer Vision and Pattern Recognition Conference, June 1993, 267-273.

[11] Rissanen, Jorma. "A Universal Prior for Integers and Estimation by Minimum Description Length." *The Annals of Statistics* 11 (no. 2, 1983): 416-431.

[12] Super, Boaz J. and Alan C. Bovik. "Three-Dimensional Orientation from Texture Using Gabor Wavelets." *SPIE Conference on Visual Communications and Image Processing*, November 1991.

[13] Super, Boaz J. and Alan C. Bovik. "Shape-from-Texture by Wavelet-Based Measurement of Local Spectral Moments." *IEEE Conference on Computer Vision and Pattern Recognition*, 296-301, June 1992.

[14] Van Gool, L., P. Dewaele, and A. Oosterlinck. "Texture Analysis Anno 1983." *Computer Vision, Graphics, and Image Processing* 29 (1985): 336-357.

[15] Wallace, Richard S. "Finding Natural Clusters through Entropy Minimization" (Ph.D. Thesis). Carnegie Mellon University Computer Science Technical Report No. CMU-CS-89-183, June 1989.

[16] Wechsler, Harry. "Texture Analysis - A Survey." *Signal Processing* 2 (1980): 271-282.