ARPA Image Understanding
Workshop 1996

Coalescing Texture Descriptors*

Yossi Rubner and Carlo Tomasi
Computer Science Department, Stanford University
Stanford, CA 94305

Abstract

As an alternative to texture segmentation for the description of the images, we propose a method for coalescing descriptors of adjacent image patches with similar textural content into tight clusters. We achieve this result by extending the notion of edge-preserving smoothing and anisotropic diffusion from gray-level and color images to vector-valued images that describe the textural content of each image patch. To this end, we first compute *raw texture descriptors*, that is, vectors that describe local texture appearance. We then define *texture contrast* as the maximum derivative of the texture descriptor vector, and we define *significant regions* as those in which contrast is low. We propose an efficient implementation of edge-preserving smoothing and show that it is closely related to anisotropic diffusion. Experiments on texture mosaics and real images show that clear texture edges are found even where intensity edges are weak or poorly defined, and that significant texture regions yield tight clusters of texture descriptors.

1 Introduction

Describing the textural content of images is an important task in the indexing, perceptual organization, and interpretation of pictures. For instance, in our work on image databases [Tomasi and Guibas, 1994; Guibas *et al.*, 1995] we index pictures by vectors that encode the texture content of different parts of an image. To this end, it is useful to determine areas of approximately uniform texture. In fact, each of these areas can then be analyzed separately and as a unit. Integration of texture information over an entire region, as opposed to a small patch, yields descriptor vectors that are at the same time more accurate and fewer in number. In addition, areas of mixed texture content can be excluded from analysis, thereby keeping the space of indexes clean of spurious vectors.

The traditional approach to the determination of uniform texture regions has been texture segmentation. In this approach, images are partitioned into regions of supposedly uniform texture. The insistence on partitioning, that is on the requirement to label each pixel in the image with exactly one label, although apparently a desirable outcome, conceals in reality several difficulties.

First, many parts of many images do not contain any "texture," under any definition of this term. For instance, texture segmentation would require to tag each eye in a person's portrait as a single texture, or to partition it into separate textures. This is of course possible in principle, if anything by assigning each pixel to a separate region, but it is hardly meaningful.

Second, the exact location of texture boundaries is hard to determine, and is often not required by the application. Some times, two textures blend into each other gradually, in which case the notion of a boundary is not even appropriate. However, even if a boundary does exist, whether a pixel belongs to one or the other texture is often a futile if not impossible question to answer.

Third, explicitly identifying the extent of uniform-texture regions requires complex procedures like region growing, splitting, or merging. However, it is often sufficient to know, for every patch in the image, whether that patch is similar in some sense to its neighbors. Computing regions, that is, the transitive closure of this similarity relation, is expensive and not always useful.

Fourth, and most fundamentally, the very notion of a "uniform" texture is of dubious validity. Textures are often random, and no two

*Research supported by ARPA grant DAAH04-94-G-0284.

patches are exactly equal to each other. In addition, uniform textures in the world become nonuniform in the image because of differences in distance, foreshortening, shading, and lighting. Insisting on determining in all cases whether two patches belong to the same texture requires restrictive texture models and expensive computation, and leads to questionable results.

In contrast with segmentation, we propose instead to let descriptors of similar and adjacent texture patches *coalesce* into tight clusters. In this approach, raw texture descriptors are first computed for a fixed set of overlapping image patches. If two adjacent patches contain similar textures, a smoothing process brings the corresponding descriptors even closer to each other. If the textures in two adjacent patches are very different, however, then their descriptor vectors are modified so that their distance is left unchanged or is even increased.

In other words, we propose to do edge-preserving smoothing, but in the space of texture vectors rather than for image intensities. This requires generalizing the definition of gradient to a vector function, which we do by introducing a measure of *texture contrast*. Thanks to this definition, we can run the equivalent of anisotropic diffusion processes [Perona and Malik, 1990] to achieve edge-preserving smoothing. Running these processes in a space with many dimensions brings also efficiency considerations to the forefront, and we propose an efficient approximation to anisotropic diffusion. This is particularly important in an application like retrieval from image databases, where large numbers of images must be processed when they are entered into the database, and where query images must be processed in a short time to ensure interactive behavior.

The "hard" decisions of texture segmentation are eliminated by our approach. The exact location of boundaries is unimportant, and yet boundary regions can be identified with sufficient precision that they can be tagged as "insignificant" for texture analysis. Even if only coarsely localized, strong texture edges can be found by our method, even when intensity edges are weak. Also, clusters of texture descriptor vectors can later be replaced by single vectors not by precisely labeling regions, but simply by standard vector quantization techniques [Gersho and Gray, 1992]. Now, whether vector clusters correspond exactly to image regions becomes irrelevant: clustering is done by optimizing information compression, rather than by satisfying some semantically loaded criterion of uniformity.

The next section introduces our version of raw texture descriptors. These are standard Gabor filters, placed on a log-polar grid in the frequency domain. Section 3 defines texture contrast. Section 4 discusses our efficient, edge-preserving smoothing algorithm, and section 5 concludes with some final remarks and a look at future work.

2 Texture Descriptors

A common way to describe texture is to compute the projection of the image intensity function onto a basis of functions. This is referred to as a *spectral decomposition*, because each of the different basis functions is usually concentrated in a different area of the frequency domain. In such a representation, a texture is represented by a vector of values, each value corresponding to the energy in a specified scale and orientation subband. Spectral decomposition methods include using quadrature filters [Knutsson and Granlund, 1983], Gabor filters [Farrokhnia and Jain, 1991] [Bigün and du Buf, 1994], steerable filters [Freeman and Adelson, 1991] [Simoncelli *et al.*, 1992] [Perona, 1991] [Greenspan *et al.*, 1994] [Shy and Perona, 1994] [Heeger and Bergen, 1995], and the cortex transform [Watson, 1987]. For our experiments, we implemented the Gabor filters because of their simplicity.

2.1 Gabor Filters

Gabor functions [Gabor, 1946] are Gaussians modulated by complex sinusoids. A useful property of these functions is that they are maximally compact in both space and frequency. In order to give the same emphasis to different frequency octaves, and because natural textures often have a linearly decreasing log power spectrum [Field, 1987], we use what we call *log-Gabor* filters. The centers of these filters in the frequency domain are equally spaced in a log-polar representation of the spectrum of an image. More specifically, if ω_r and ω_φ are variables along the logarithm of the magnitude and along the phase of the 2D frequency plane, we use the filters derived by Bigün and du Buf (1994) :

$$G_{ij}(\omega_r, \omega_\varphi) = G(\omega_r - \omega^0_{r_i}, \omega_\varphi - \omega^0_{\varphi_j}) \qquad (1)$$

where

$$G(\omega_r, \omega_\varphi) = \exp\left(\frac{-\omega_r^2}{2\sigma_{r_i}^2}\right) \exp\left(\frac{-\omega_\varphi^2}{2\sigma_{\varphi_j}^2}\right)$$

with $1 \leq i \leq M$ and $1 \leq j \leq N$. Here, M is the number of scales and N is the number

of orientation bands. The N orientations are taken to be equidistant:

$$\begin{aligned} \sigma_{\varphi_j} &= \pi/2N \\ \omega^0_{\varphi_j} &= 2\sigma_{\varphi_j}(j-1) \end{aligned}$$

for $1 \leq j \leq N$. The M scales are obtained by dividing the frequency range $\omega_{max} - \omega_{min}$ into the desired number of octaves: $2\sigma + 4\sigma + \ldots + 2^M\sigma = \omega_{max} - \omega_{min}$ which yields $\sigma = (\omega_{max} - \omega_{min})/2(2^M - 1)$ and

$$\begin{aligned} \sigma_{r_i} &= 2^{i-1}\sigma \\ \omega^0_{r_i} &= \omega_{min} + \{1 + 3(2^{i-1} - 1)\}\sigma \end{aligned}$$

for $1 \leq i \leq M$. Figure 1 shows the frequency response of the $M \times N$ filter bank for $M = 4$, $N = 4$.

The responses of the Gabor filters on an image are computed in the frequency domain by multiplying the Fourier-transformed image (after removing the DC component) with the filter responses in equation (1), and transforming the results back to the space domain. Working in the frequency domain avoids the image boundary problems which occur when convolving in the spatial domain. Also, we only retain the magnitudes of the responses since these encode the energy content and are independent of position within the texture region.

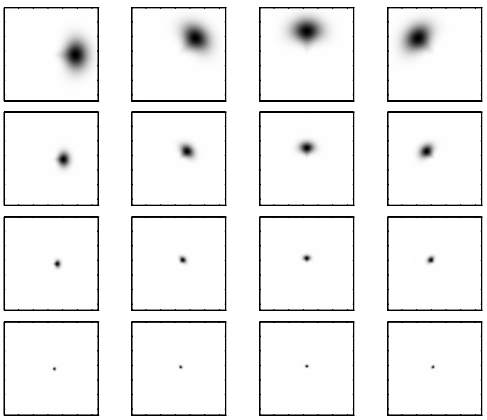


Figure 1: Frequency responses for Gabor filters for $M = 4$ and $N = 4$.

This filtering stage returns MN response images. The values from corresponding positions in these images form MN-dimensional vectors, which are our raw texture descriptors. To reduce the dependence of the responses on lighting and in order to emphasize the texture structure information, we normalize the vectors to be unit vectors. This makes our texture space a MN-dimensional hypersphere.

Figure 2 shows a mosaic of four textures from the Brodatz album [Brodatz, 1966]. Figure 3 shows the result of applying the Gabor filters from figure 1 to this image. We down-sampled each response image by a factor of four to save computations. As expected, different textures give different responses in the various scale and orientation subbands.

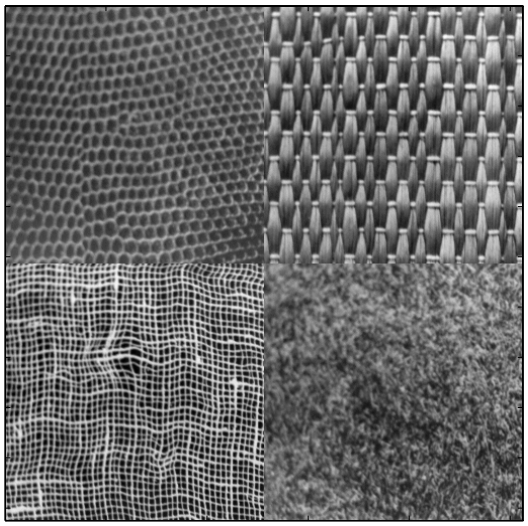


Figure 2: Mosaic of four textures from the Brodatz album.

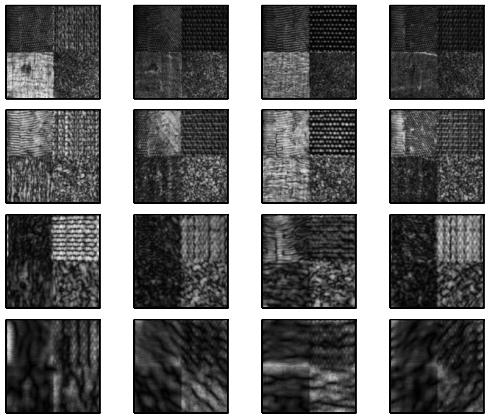


Figure 3: The result of applying the 16 Gabor filters in figure 1 to the mosaic in figure 2.

3 Texture Contrast

The raw texture vectors introduced in the previous section describe the local appearance of small image neighborhoods. The size of these neighborhoods is equal to that of the supports of the coarser resolution filters employed. On the one hand, these supports are large enough that they can straddle boundaries between different textures. In this case, they do not describe "pure" textures, and they can convey information that is hard to interpret and can be

misleading. On the other hand, the supports are often too small for them to "see" enough of a texture to yield a reliable description. In fact, the basic period of repetition of the underlying texture may be comparable to the filter support sizes, so that adjacent filters see somewhat different parts of the texture, and the description vectors differ somewhat. Also, textures exhibit variations in the world, as well as variations caused by nonuniform viewing parameters.

Before using texture vectors for indexing or interpretation, it is therefore preferable to sift and summarize the information that they convey. Specifically, it is desirable to eliminate vectors that describe mixtures of textures, and to average away some of the variations between adjacent descriptors of similar image patches. This poses a problem that is typical in the perceptual organization of images: small differences between attributes of adjacent areas should be reduced, if not obliterated, and large, extended differences should be enhanced to prevent contamination between dissimilar areas. The techniques of edge preserving smoothing [Saint-Marc *et al.*, 1991] and anisotropic diffusion [Perona and Malik, 1990] have been proposed to address this problem for gray scale images. For texture, a notion of *texture contrast* is needed to tell "small" from "large" changes, in order to apply analogous techniques. Also, because of the length of texture vectors, efficient processing is crucial, especially when several thousand images are processed as is the case in image database applications.

For texture contrast, we employ a notion of "generalized gradient" from differential geometry [Kreyszing, 1959] that has been used for color images in the past few years [Di Zenzo, 1986; Cumani, 1991; Sapiro and Ringach, 1994]. The rest of this section discusses this notion as it applies to texture.

Assume that we have a mapping $\mathbf{\Phi} : S \subseteq \Re^n \rightarrow \Re^m$. Let ϕ_i denote the ith component of $\mathbf{\Phi}$. If $\mathbf{\Phi}$ is a texture vector space, for example, then ϕ_i, $1 \leq i \leq m$ are the spectral decomposition subbands, where $m = MN$. If $\mathbf{\Phi}$ admits a Taylor expansion we can write

$$\mathbf{\Phi}(\mathbf{x}+\Delta\mathbf{x}) = \mathbf{\Phi}(\mathbf{x})+\mathbf{\Phi}'(\mathbf{x})\Delta\mathbf{x}+\|\Delta\mathbf{x}\|e(\mathbf{x},\Delta\mathbf{x}) .$$

where $\|e(\mathbf{x},\Delta\mathbf{x})\| \rightarrow 0$ as $\Delta\mathbf{x} \rightarrow 0$ and $\mathbf{\Phi}'(\mathbf{x})$ is the $m \times n$ Jacobian matrix of Φ:

$$\mathbf{\Phi}'(\mathbf{x}) = J(\mathbf{x}) = \begin{pmatrix} \frac{\partial\phi_1}{\partial x_1} & \cdots & \frac{\partial\phi_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial\phi_m}{\partial x_1} & \cdots & \frac{\partial\phi_m}{\partial x_n} \end{pmatrix} .$$

If one starts at point x and moves by a small step $\Delta\mathbf{x}$, the distance traveled in the attribute domain is approximately

$$d \cong \|\mathbf{\Phi}'(\mathbf{x})\Delta\mathbf{x}\| = \sqrt{\Delta\mathbf{x}^T J^T J \Delta\mathbf{x}} .$$

The step direction which maximizes d is the eigenvector of $J^T J$ corresponding to its largest eigenvalue. The square root of the largest eigenvalue, or, equivalently, the largest singular value of $\mathbf{\Phi}'$, corresponds to the gradient magnitude, and the corresponding eigenvector is the gradient direction.

We can give a closed form solution for the eigenvalues in the case that $n = 2$, as it is for images. In this case, the differential of $\mathbf{\Phi}$ is

$$d\mathbf{\Phi} = \sum_{i=1}^{2} \frac{\partial\mathbf{\Phi}}{\partial x_i} dx_i .$$

and so

$$\|d\mathbf{\Phi}\|^2 = \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{\partial\mathbf{\Phi}}{\partial x_i} \cdot \frac{\partial\mathbf{\Phi}}{\partial x_j} dx_i dx_j .$$

Using the notation from Riemannian geometry [Kreyszing, 1959], we have

$$\begin{aligned} \|d\mathbf{\Phi}\|^2 &= \sum_{i=1}^{2}\sum_{j=1}^{2} g_{ij} dx_i dx_j \\ &= \begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix}^T \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix} . \end{aligned}$$

where

$$g_{ij} \equiv \sum_{k=1}^{m} \frac{\partial\phi_k}{\partial x_i} \frac{\partial\phi_k}{\partial x_j} .$$

and $g_{12} = g_{21}$. Let now

$$\lambda_\pm = \frac{1}{2}\left(g_{11} + g_{22} \pm \sqrt{(g_{11}-g_{22})^2 + 4g_{12}^2}\right) .$$

be the two eigenvalues in the matrix $G = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}$. Since G is real and symmetric, its eigenvalues are real.

We choose λ_+ as the generalized gradient magnitude since it corresponds to the direction of maximum change. We can verify that this magnitude reduces to the ordinary gradient norm in the case of a gray scale image ($m = 1$):

$$\begin{aligned} \lambda_+ &= \frac{1}{2}\left(\mathbf{\Phi}_x^2 + \mathbf{\Phi}_y^2 + \sqrt{(\mathbf{\Phi}_x^2 - \mathbf{\Phi}_y^2)^2 + 4\mathbf{\Phi}_x^2\mathbf{\Phi}_y^2}\right) \\ &= \frac{1}{2}\left(\mathbf{\Phi}_x^2 + \mathbf{\Phi}_y^2 + (\mathbf{\Phi}_x^2 + \mathbf{\Phi}_y^2)\right) \\ &= \mathbf{\Phi}_x^2 + \mathbf{\Phi}_y^2 = \|\nabla\mathbf{\Phi}\|^2 \end{aligned}$$

where the subscripts x and y denote differentiation.

While the application of this definition to color is relatively straight-forward, matters are more complicated for texture. In fact, color is a point property of an image, while texture must be defined over neighborhoods. Because of the multiscale nature of texture in general, and of our raw textured descriptors in particular, different filters have in general different supports. Computing derivatives at different scales, that is, for different values of ω_r in equation (1), requires operators of appropriate magnitudes and spatial supports in order to yield properly scaled components. For instance, if differentiation is performed by convolution with the derivatives of a Gaussian, the standard deviation of each Gaussian must be proportional to scale, and the magnitudes must be properly normalized.

Figure 4(a) shows the texture contrast of the texture mosaic from figure 2. The texture boundaries are characterized by high contrast. However, even inside uniform texture regions some residue contrast appears. This is discussed in the next section.

4 Edge-Preserving Smoothing

Figure 4(a) shows some areas with relatively high texture contrast even inside regions that appear to be of the same texture. The sources of these contrast areas have been discussed at the beginning of the previous section, where it was pointed out that small, low-contrast areas should be smoothed away, while extended ridges of the contrast function should be maintained. The measure of texture contrast introduced above allows extending anisotropic diffusion [Perona and Malik, 1990] or edge-preserving smoothing [Saint-Marc *et al.*, 1991] techniques to texture descriptors. In this section, we present an efficient implementation of an edge-preserving smoothing algorithm, which is based on repeatedly convolving each of the spectral subbands of the image with a separable binomial filter weighted by a measure of the texture contrast. After describing the algorithm, we explore its relation with the method described in [Perona and Malik, 1990].

First the weighting function $g(C)$ is introduced, where C is the texture contrast defined is section 3. The function $g(C)$ should be high where the texture is uniform and low where the texture is "edgy" and can be any nonnegative monotonically decreasing function with $g(0) = 1$. We chose to use

$$g(C) = e^{-\left(\frac{|C|}{k}\right)^2} \qquad (2)$$

where k controls the decay rate of $g(\cdot)$ and, as we will see later, determines which of the edges is preserved.

Smoothing is performed by convolving the spectral subbands of the image with the binomial filter

$$\mathbf{B} = B^T B \qquad ; \qquad B = [\ 1 \quad 2 \quad 1\ ]\ .$$

after it is weighted by $g(C)$. We chose this filter because it is separable and can be applied efficiently. When applied repeatedly, the binomial filter quickly gives an approximation of a Gaussian. For brevity, introduce the following notation:

$$\begin{aligned} a *_x b &= \sum_{i=-1}^{1} a(i)b(x+i,y) \\ a *_y b &= \sum_{j=-1}^{1} a(j)b(x,y+j) \\ c * b &= \sum_{i=-1}^{1}\sum_{j=-1}^{1} c(i,j)b(x+i,y+j)\ . \end{aligned}$$

A single iteration of the smoothing procedure is computed as follows at time t:

1. Compute the texture contrast $C^{(t)}$ as described in section 3.
2. $G^{(t)} = g(C^{(t)})$.
3. For $0 \le m < M$ and $0 \le n < N$ do:
   (a) Let $I^{(t)}$ be the m-th scale and the n-th orientation subband.
   (b) Compute:

$$\begin{aligned} K^{(t)} &= B *_x G^{(t)} \\ J^{(t)} &= \frac{B *_x (G^{(t)} I^{(t)})}{K^{(t)}} \\ L^{(t)} &= B *_y K^{(t)} \\ I^{(t+1)} &= \frac{B *_y (K^{(t)} J^{(t)})}{L^{(t)}}\ . \end{aligned}$$

(This computation can be simplified. For clarity, we bring it in the above form). It is easy to see that

$$I^{(t+1)} = \frac{\mathbf{B} * (G^{(t)} I^{(t)})}{\mathbf{B} * G^{(t)}}\ .$$

Without the $\mathbf{B}(i,j)$ factor, this is similar to the adaptive smoothing proposed in [Saint-Marc *et al.*, 1991], but implemented, in addition, in a separable fashion for greater efficiency. In [Saint-Marc *et al.*, 1991], this iterative process is proven to be stable and to preserve edges. A similar proof can be applied to our case with

straight-forward modifications. That edges are preserved is proved by showing that when the contrast is large enough ($> k$), it increases as the iterations progress, thereby sharpening the edge. When the contrast is small ($< k$), the contrast decreases and the edge is smoothed. This implies that k is equivalent to a contrast threshold.

In the following, we show that the edge-preserving smoothing iteration is equivalent to an iteration of the anisotropic diffusion proposed by Perona and Malik (1990) . Define

$$c^{(t)} = \frac{G^{(t)}}{\mathbf{B} * G^{(t)}} .$$

Then we have

$$I^{(t+1)} = \mathbf{B} * (c^{(t)} I^{(t)})$$

and by using the fact that

$$\mathbf{B} * c^{(t)} = 1$$

we can write

$$I^{(t+1)} - I^{(t)} = \mathbf{B} * [c^{(t)}(I^{(t)} - I^{(t)}\delta)] \quad (3)$$

where $\delta(x, y) = 1$ if $x = y = 0$ and 0 otherwise. Equation (3) is a discretization of the anisotropic diffusion equation

$$I_t = \nabla \cdot (c(x, y, t)\nabla I) .$$

Instead of using a 4-neighbor discretization of the Laplacian as done by Perona and Malik, we use a better, 8-neighbor discretization [Jähne, 1995]:

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix} .$$

Figure 4 (b) shows the texture contrast after 20 iterations. To visualize the MN-dimensional vectors we project them onto the plane that is spanned by the two most significant principal components of all texture vectors in the image. Figures 4(c), (d) show the projections of the texture descriptors before and after the edge-preserving smoothing. Only descriptors from *significant regions* are shown in the latter. A region is significant if the contrast after smoothing is everywhere smaller than the parameter k in equation (2). We can see that the descriptors of the four textures form clear distinct clusters even in this two-dimensional projection. Notice the sparse trail of points that connects the two leftmost and the two up-most clusters in figure 4(d). These points come from a "leakage" in the boundary between two textures. This implies that we are limited in the amount of smoothing we can do before different textures start to mix. This is also a situation where most segmentation

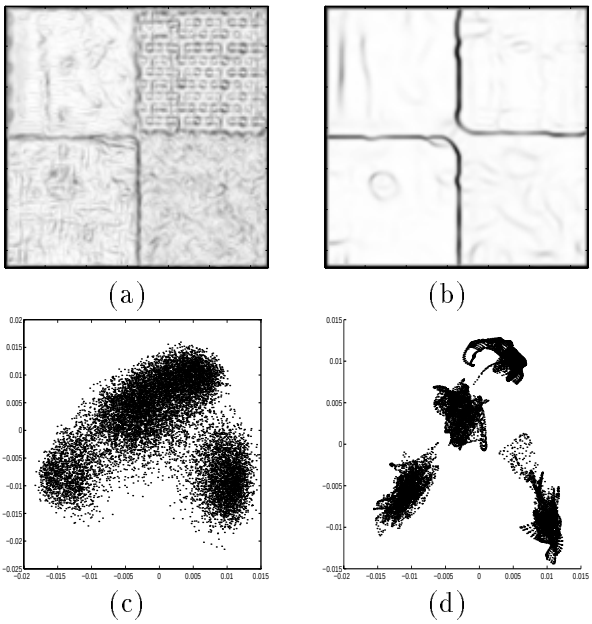


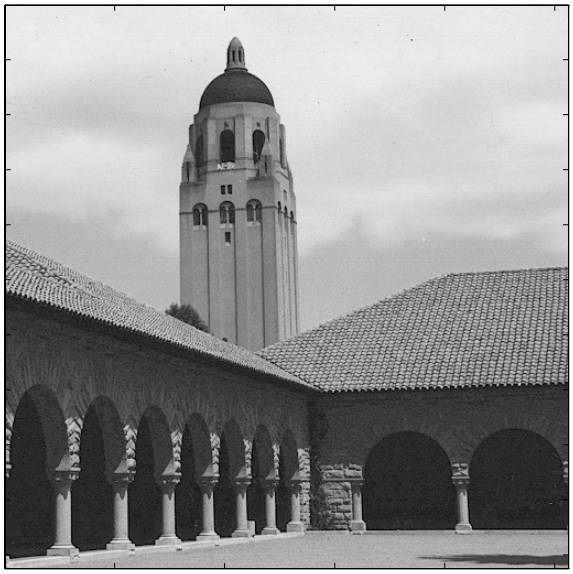
Figure 4: (a) Texture contrast of figure 2 before smoothing (dark means large) and (b) after 20 iterations. (c) Projection of the texture descriptors onto the plane of their two most significant principal components before smoothing and (d) after 20 iterations.

algorithms would have to make a classification decision, even if the two textures involved are in fact similar to each other.

In figure 5 we see a real image and the result of edge-preserving smoothing after only five iterations. We used four scales and four orientation bands. Notice the strong texture edges even where intensity edges are poorly defined or weak.

The same processing was performed on the image in figure 6. Figure 7 shows the contrast measure and the significant regions. Notice that there are many fewer edges than an intensity-edge detector would find, and yet regions of different textures are well delineated. For instance, the picket fence is essentially found to be one region, while and edge detector would have delineated each picket. Also the walls of the buildings are now nearly edge-free, in spite of substantial intensity contrast and even in the presence of relatively large variations in the attributes of the textures within each region.

Some of the texture boundaries in both figures 5 and 7 appear as double, parallel lines. This doubling is caused by the response of filters that are insensitive to both textures along the edge. While these filters give essentially no response to the regions adjacent to these edges, the edge



(a)

(b)

Figure 5: (a) A real image. (b) Texture edges after 5 iterations of the smoothing.

itself creates a peak, which is then differentiated by the contrast detector, thereby creating double ridges. We are studying ways to eliminate this effect.

5 Conclusion

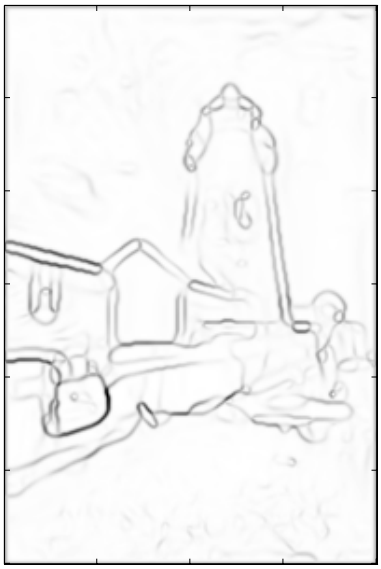
In this paper, we have proposed an alternative to texture segmentation for grouping local texture descriptors. Rather than partitioning the image into regions with "the same" texture, a poorly defined notion, we coalesce similar and adjacent texture descriptors into tight clusters by a process analogous to the edge-preserving smoothing procedures that have traditionally been applied to image intensities or to color. The main advantage of this alternative approach is that hard decisions about boundaries and "sameness" of textures need not be made. At the same time, texture descriptors are grouped in a way that allows compressing them by standard techniques like vector quantization. We have introduced a simple and powerful notion of texture contrast, and we have also defined significant regions as those where texture contrast is low. Boundary areas and complicated regions with no texture to speak of are eliminated as insignificant for texture analysis.

Figure 6: A picture of a lighthouse.

In our current and future research, we are considering variations to Gabor filters for the computation of raw texture descriptors. In particular, we are experimenting with steerable filters [Freeman and Adelson, 1991; Perona, 1991] because they can yield many different orientations with less computation. When steerable filters are computed at different scales, they can be implemented efficiently in a pyramid-like fashion. Greenspan *et al.* (1994) derive an efficient steerable pyramid which gives similar filters to the Gabor filters we are using.

Acknowledgments

We thank Brian Rogoff and Guillermo Sapiro for useful discussions.



(a)

(b)

Figure 7: (a) Texture edges after ten iterations of the smoothing. (b) Nonignificant regions are blackened out.

References

[Bigün and du Buf, 1994] J. Bigün and J. M. du Buf. N-folded symmetries by complex moments in gabor space and their application to unsupervised texture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-16(1):80–87, January 1994.

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

[Cumani, 1991] A. Cumani. Edge detection in multispectral images. *CVGIP: Graphical Models and Image Processing*, 53(1):40–51, 1991.

[Di Zenzo, 1986] S. Di Zenzo. A note on the gradient of a multi-image. *Computer Vision Graphics and Image Processing*, 33:116–125, 1986.

[Farrokhnia and Jain, 1991] F. Farrokhnia and A. K. Jain. A multi-channel filtering approach to texture segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 364–370, June 1991.

[Field, 1987] D. J. Field. Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America*, 4:2379–2394, 1987.

[Freeman and Adelson, 1991] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(9):891–906, September 1991.

[Gabor, 1946] D. Gabor. Theory of communication. *The Journal of the Institute of Electrical Engineers, Part III*, 93(21):429–457, January 1946.

[Gersho and Gray, 1992] A. Gersho and R. M. Gray. *Vector quantization and signal compression.* Kluwer Academic Publishers, Boston, MA, 1992.

[Greenspan *et al.*, 1994] H. Greenspan, S. Belongie, R. Goodman, P. Perona, S. Rakshit, and C. H. Anderson. Overcomplete steerable pyramid filters and rotation invariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 222–228, June 1994.

[Guibas *et al.*, 1995] L. Guibas, B. Rogoff, and C. Tomasi. Fixed-window image descriptors for image retrieval. In *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases*, pages 2420–31, San Jose, CA, February 1995.

[Heeger and Bergen, 1995] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *Computer Graphics*, pages 229–238. ACM SIGGRAPH, 1995.

[Jähne, 1995] B. Jähne. *Digital Image Processing.* Springer, 1995.

[Knutsson and Granlund, 1983] H. Knutsson and G. H. Granlund. Texture analysis using two-dimensional quadrature filters. *IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, 1983.

[Kreyszing, 1959] E. Kreyszing. *Differential Geometry*. University of Toronto Press, Toronto, 1959.

[Perona and Malik, 1990] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(7):629–939, July 1990.

[Perona, 1991] P. Perona. Deformable kernels for early vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 222–227, June 1991.

[Saint-Marc *et al.*, 1991] P. Saint-Marc, J. S. Chen, and G. Medioni. Adaptive smoothing: A general tool for early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(6):514–529, June 1991.

[Sapiro and Ringach, 1994] G. Sapiro and D. Ringach. Anisotropic diffusion in color space. *submitted*, 1994.

[Shy and Perona, 1994] D. Shy and P. Perona. X-y separable pyramid steerable scalable kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 237–244, June 1994.

[Simoncelli *et al.*, 1992] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. Shiftable multiscale transforms. *IEEE Transactions on Information Theory*, IT-38:587–607, 1992.

[Tomasi and Guibas, 1994] C. Tomasi and L. Guibas. Image descriptions for browsing and retrieval. In *Proceedings of the ARPA Image Understanding Workshop*, pages 165–168, November 1994.

[Watson, 1987] A. B. Watson. The cortex transform: rapid computation of simulated neural images. *Computer Vision, Graphics, and Image Processing*, 39:311–327, 1987.