IEEE International Conference on Computer Vision, Corfu, Greece, September 1999

Textons, Contours and Regions: Cue Integration in Image Segmentation

Jitendra Malik, Serge Belongie, Jianbo Shi and Thomas Leung
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720
{malik,sjb,jshi,leungt}@cs.berkeley.edu

Abstract

This paper makes two contributions. It provides (1) an operational definition of textons, the putative elementary units of texture perception, and (2) an algorithm for partitioning the image into disjoint regions of coherent brightness and texture, where boundaries of regions are defined by peaks in contour orientation energy and differences in texton densities across the contour.

Julesz introduced the term texton, analogous to a phoneme in speech recognition, but did not provide an operational definition for gray-level images. Here we re-invent textons as frequently co-occurring combinations of oriented linear filter outputs. These can be learned using a K-means approach. By mapping each pixel to its nearest texton, the image can be analyzed into texton channels, each of which is a point set where discrete techniques such as Voronoi diagrams become applicable.

Local histograms of texton frequencies can be used with a χ^2 test for significant differences to find texture boundaries. Natural images contain both textured and untextured regions, so we combine this cue with that of the presence of peaks of contour energy derived from outputs of odd- and even-symmetric oriented Gaussian derivative filters. Each of these cues has a domain of applicability, so to facilitate cue combination we introduce a gating operator based on a statistical test for isotropy of Delaunay neighbors. Having obtained a local measure of how likely two nearby pixels are to belong to the same region, we use the spectral graph theoretic framework of normalized cuts to find partitions of the image into regions of coherent texture and brightness. Experimental results on a wide range of images are shown.

1 Introduction

This paper has twin objectives. It provides (1) an operational definition of textons, the putative elementary units of image analysis, and (2) an algorithm for partitioning the image into disjoint regions based on both brightness and texture. These objectives are coupled—cue integration relies on, and thus reveals, the advantages of the texton representation.

1.1 Introducing Textons

Julesz introduced the term *texton*, analogous to a phoneme in speech recognition, more than 20 years ago [9] as the putative units of preattentive human texture perception. He described them qualitatively for simple binary line segment stimuli—oriented segments, crossings and terminators—but did not provide an operational definition for gray-level images. Subsequently, texton theory fell into disfavor as a model of human texture discrimination as accounts based on spatial filtering with orientation and scale-selective mechanisms which could be applied to arbitrary gray-level images became popular.

There is a fundamental, well recognized, problem with linear filters. Generically, they respond to any stimulus. Just because you have a response to an oriented odd-symmetric filter doesn't mean there is an edge at that location. It could be that there is a higher contrast bar at some other location in a different orientation which has caused this response. Tokens such as edges or bars or corners can not be associated with the output of a single filter. Rather it is the signature of the outputs over scales, orientations and order of the filter that is more revealing.

Here we introduce a further step by focussing on the *outputs* of these filters considered as points in a high dimensional space (typically on the order of 36 filters are used). We perform vector quantization, or clustering, in this high-dimensional space to find prototypes. Call these prototypes *textons*—we will find empirically that these do tend to correspond to oriented bars, terminators and so on. One can construct a universal texton vocabulary by processing a large number of natural images, or we could find them adaptively in windows of images. In each case the k-means technique can be used. By mapping each pixel to the texton nearest to its vector of filter responses, the image can be analyzed into texton channels, each of which is a point set.

It is our opinion that the analysis of an image into textons will prove useful for a wide variety of visual processing tasks. For instance, in [13] we use the related notion of 3D textons for recognition of textured materials. In the present paper, our objective is to develop an algorithm for the segmentation of an image into regions of coherent brightness and texture–we will find that the texton representation will enable us to address the key problems in a very natural fashion. Let's begin with a review of the outstanding issues in low level image segmentation.

1.2 Challenges in image segmentation

Scale selection in textured regions continues to be a fundamental problem–whatever one's choice of textured descriptor, it has to be computed over a local window whose size and shape need to be determined adaptively. What makes scale selection a challenge is that the technique must deal with the wide range of textures–regular such as the polka dots in Figure 8(b), stochastic in Figure 8(a), or intermediate cases such as the stripes of the tiger in Figure 8(c)–in a seamless way. Furthermore it would be desirable if in the neighborhood of boundaries, the windows over which texture descriptors are computed could be shaped to lie largely on the correct side of the boundary.

The other major issue is dealing with images which have both textured and untextured regions. Here boundaries must be found using *both* contour and texture analysis. However what we find in the literature are approaches which concentrate on one or the other.

Contour analysis (e.g. edge detection) may be adequate for untextured images, but in a textured region it results in a meaningless tangled web of contours. Think for instance of what an edge detector would return on the snow region in Figure 8(a). The traditional "solution" for this problem in edge detection is to use a high threshold so as to minimize the number of edges found in the texture area. This is obviously a non-solution–such an approach means that low-contrast extended contours will be missed as well. There is no recognition of the fact that extended contours, even weak in contrast, are perceptually significant.

While the perils of using edge detection in textured regions have been noted before (see eg. [2]), a complementary problem of contours constituting a problem for texture analysis does not seem to have been recognized before. Typical approaches are based on measuring texture descriptors over local windows, and then computing differences between window descriptors centered at different locations. Boundaries can then give rise to thin strip-like regions, as in Figure 1. For specificity, assume that the texture descriptor is a histogram of linear filter outputs computed over a window. Any histogram window near the boundary of the two regions will contain large filter responses from filters oriented along the direction of the edge. However, on both sides of the boundary, the histogram will indicate a featureless region. A segmentation algorithm based on, say, χ^2 distances between histograms, will inevitably partition the boundary as a group of its own. As is evident, the problem is not confined to the use of a histogram of filter outputs as texture descriptor. Figure 1 (b) shows the actual groups found by an EM style algorithm using an alternative color/texture descriptor [1].

(a)

(b)

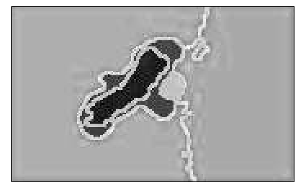

Figure 1. Demonstration of the "contour-as-a-texture" problem using a real image. (a) Original image of a bald eagle. (b) The groups found by an EM style algorithm [1].

1.3 Our approach

We pursue image segmentation in the framework of Normalized Cuts introduced by Shi and Malik [18]. The image is considered to be a weighted graph where the nodes are pixels and arc weights denote a local measure of similarity between the two pixels. Grouping is performed by finding eigenvectors of the Normalized Laplacian of this graph. The fundamental issue then is that of specifying the arc weights W_{ij}; we rely on normalized cuts to go from these local measures to a globally optimal partition of the image.

The algorithm begins by analyzing the image into textons (§2). In the next stage (§3), we determine for every pixel a suitable local neighborhood, the appropriate window for computing the local texture descriptor, and a measure of the anisotropy of this neighborhood. A histogram of texton densities is used as the texture descriptor. We use a gating operator based on a statistical test for isotropy of Delaunay neighbors. These computations critically rely on the spatial analysis of individual texton channels. The fact that each texton channel is a point set is very convenient, because it enables one to use discrete techniques such as Voronoi diagrams and Delaunay triangulations.

We are now ready (§4) to specify the arc weights W_{ij} combining both brightness and texture information. The texture cue is coded by making the arc weight dependent on χ^2 differences between local texton histograms; and the brightness cue is treated in the *intervening contour* framework of Leung and Malik [12] using peaks in contour orientation energy. The anisotropy of the local descriptor window at a pixel serves to gate between these cues so as to circumvent the problems listed in §1.2.

Results from the algorithm are presented in §5.

2 Filters and Textons

Since the early 1980s, many approaches have been proposed in the computer vision literature that employ *filter-based* descriptions of images [6, 10, 14]. By the term *filter-based* we mean that the fundamental representation for a pixel in an image includes not only its brightness or color information, but also the inner product of the neighborhood centered on that pixel with a set of filters tuned to various orientations and spatial frequencies. (See Figure 2 for an example of such a filter set.)

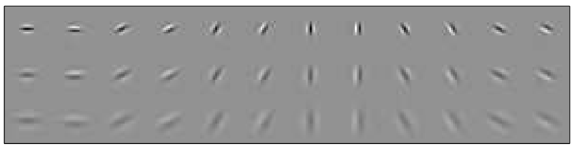

Figure 2. Gaussian derivative filter set consisting of 2 phases (even and odd), 3 scales (spaced by half-octaves), and 6 orientations (equally spaced from 0 to π). The basic filter is a difference-of-Gaussian quadrature pair with $3:1$ elongation. Each filter is divided by its L_1 norm for scale invariance.

As discussed for example in [8, 11], vectors of filter responses have many appealing properties, including relationships to physiological findings in the primate visual system [3] and to the basic mathematical notion of a Taylor series expansion.

Though the representation of textures using filter responses is extremely versatile, one might say that it is overly redundant (each pixel values is represented by N_{fil} filter responses, where N_{fil} is usually around 36). Moreover, it should be noted that we are characterizing textures, entities with some spatially repeating properties by definition. Therefore, we do not expect the filter responses to be totally different at each pixel over the texture. Thus, there should be several distinct filter response vectors and all others are noisy variations of them.

This observation leads to our proposal of clustering the filter responses into a small set of prototype response vectors. We call these prototypes *textons*. Algorithmically, each texture is analyzed using the filter bank shown in Figure 2. There are a total of 36 filters. Each pixel is now transformed to a $N_{fil} = 36$ dimensional vector of filter response These vectors are clustered using a K-means algorithm. The criterion for this algorithm is to find K "centers" such that after assigning each data vector to the nearest center, the sum of the squared distance from the centers is minimized. K-means is a greedy algorithm that finds a local minimum of this criterion[1]. In this paper, we use the `kmeans` function in the NETLAB toolbox [15].

[1]For more discussions and variations of the K-means algorithm, the reader is referred to [4, 7].

It is useful to visualize the resulting cluster centers in terms of the original filter kernels. To do this, recall that each cluster center represents a set of projections of each filter onto a particular image patch. We can solve for the image patch corresponding to each cluster center in a least squares sense by premultiplying the vectors representing the cluster centers by the pseudoinverse of the filterbank [8]. The matrix representing the filterbank is formed by concatenating the filter kernels into columns and placing these columns side by side. The set of synthesized image patches for two test images are shown in Figures 3(b) and 4(b). These are our textons. The textons represent assemblies of filter outputs that are characteristic of the local image structure present in the image.

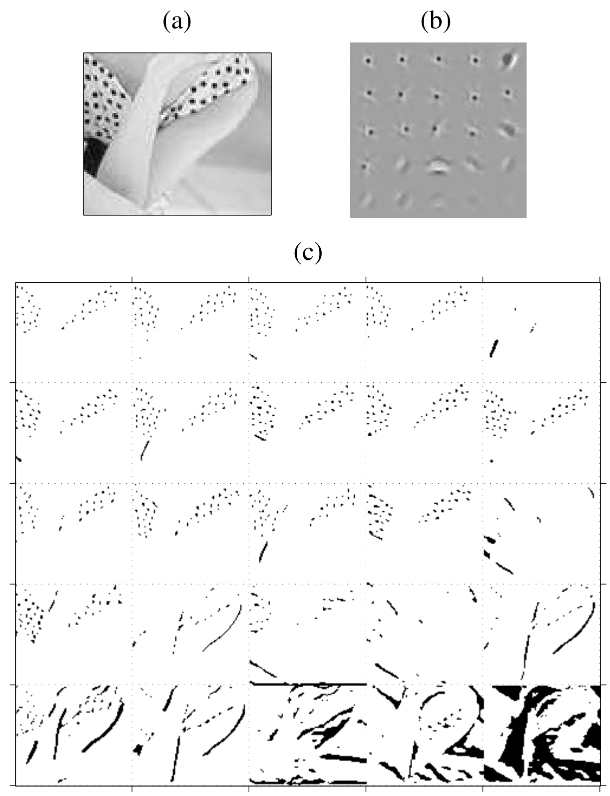

Figure 3. (a) Polka-dot image. (b) Textons found via K-means with $K = 25$, sorted in decreasing order by norm. (c) Mapping of pixels to the texton channels. The dominant structures captured by the textons are translated versions of the dark spots. We also see textons corresponding to faint oriented edge and bar elements. Notice that some channels contain activity inside a textured region or along an oriented contour and nowhere else.

Looking at the polka-dot example, we find that many of

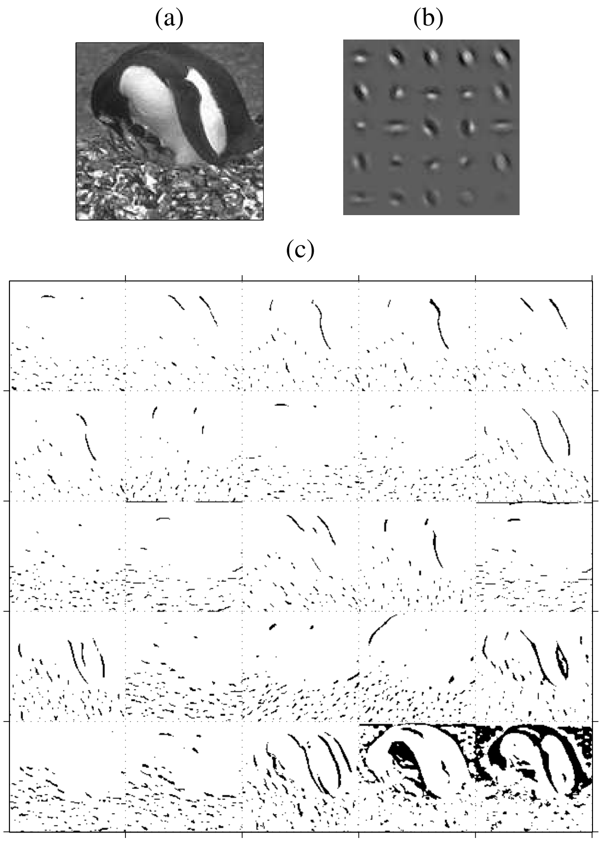

Figure 4. (a) Penguin image. (b) Textons found via K-means with $K = 25$, sorted in decreasing order by norm. (c) Mapping of pixels to the texton channels. Among the textons we see edge elements of varying orientation and contrast along with elements of the stochastic texture in the rocks.

the textons correspond to translated versions of dark spots[2]. Also included are a number of oriented edge elements of low contrast and two textons representing nearly uniform brightness. The pixel-to-texton mapping is shown in Figure 3(c). Each subimage shows the pixels in the image that are mapped to the corresponding texton in Figure 3(b). We refer to this collection of discrete point sets as the texton *channels*. Since each pixel is mapped to exactly one texton, the texton channels constitute a partition of the image.

Textons and texton channels are also shown for the penguin image in Figure 4. Notice in the two examples how much the texton set can change from one image to the next. The spatial characteristics of both the deterministic polka dot texture and the stochastic rocks texture are captured across several texton channels. In general, the texture boundaries emerge as point density changes across the different texton channels. In some cases, a texton channel contains activity inside a particular textured region and nowhere else. By comparison, vectors of filter outputs generically respond with some value at every pixel – a considerably less clean alternative.

3 Texton Channel Analysis

As discussed in the preceding section, the mapping from pixel to texton channel provides us with a number of discrete point sets where before we had continuous-valued filter vectors. Such a representation is well suited to the application of techniques from computational geometry and point process statistics. With these tools, one can approach questions such as, "what is the neighborhood of a texture element?" and "how similar are two pixels inside a textured region?"

3.1 Defining Local Scale Selection

The texton channel representation provides us a natural way to define texture scale. If the texture is composed of texels, we might want to define a notion of texel neighbors and consider the mean distance between them to be a measure of scale. Of course, many textures are stochastic and detecting texels reliably may be hard even for regular textures.

With textons we have a "soft" way to define neighbors. For a given pixel in a texton channel, first consider it as a "thickened point"— a disk centered at it. The idea is that while textons are being associated with pixels, since they correspond to assemblies of filter outputs, it is better to think of them as corresponding to a small image disk defined by the scale used in the Gaussian derivative filters. Recall Koenderink's aphorism about a point in image analysis being a Gaussian blob of small σ !

Now consider the Delaunay neighbors of all the pixels in the thickened point of a pixel i which lie closer than some outer scale.[3]. The statistics of Delaunay edge lengths provides a natural measure of scale. In passing, we note that this neighborhood tends to be in the same image region as pixel i, since all the nodes in it belong to the same texton channel and are proximal.

In Figure 5a, the Delaunay triangulation of a zoomed-in portion of one of the texton channels in the rocky region of Figure 4 is shown atop a brightened version of the image. Here the nodes represent points that are similar in the image while the edges provide proximity information.

[2]It is straightforward to develop a method for merging translated versions of the same basic texton, though we have not found it necessary. Merging in this manner decreases the number of channels needed but necessitates the use of phase-shift information.

[3]This is set to 13 pixels in our experiments.

3.2 Characterizing Isotropy

We will find it necessary later in this paper when we examine cue integration to have a statistical test for whether a pixel is in the interior of a textured region or on its boundary. The notion of Delaunay neighborhood for a thickened point defined previously can be used. We consider the orientations of the vector from pixel i to each of these points. [4]

We obtain the local estimate of isotropy by performing a simple statistical test for randomness on the neighborhood the pixels inside each texton channel. The following description will use Figure 5 to illustrate this by example; here we consider two sample pixels in the penguin image, the first on the wing boundary, the second inside the rocky ground.

The neighborhood for a point on the wing boundary is shown in Figure 5c; the filled circle marks pixel i and the open circles mark its neighbors. For this pixel, the neighborhood is clearly not isotropic. This is quantified by computing the modified Kuiper statistic V [5] from the angles of the vectors connecting pixel i to its neighbors. Denoting the sorted angles by $\theta_1, \ldots, \theta_n$, the test proceeds as follows. Let

$$x_1 = \theta_1/2\pi, \ldots, x_n = \theta_n/2\pi$$

and compute the statistics

$$D_n^+ = \text{maximum of} \quad \frac{1}{n} - x_1, \frac{2}{n} - x_2, \ldots, 1 - x_n$$

$$D_n^- = \text{maximum of} \quad x_1, x_2 - \frac{1}{n}, x_3 - \frac{2}{n}, \ldots, x_n - \frac{n-1}{n}$$

$$V_n = D_n^+ + D_n^-, \quad \text{and} \quad V = V_n(n^{\frac{1}{2}} + 0.155 + 0.24/n^{\frac{1}{2}})$$

Intuitively, this is like a Kolmogorov-Smirnov test for uniformity where the data points are angles. Tabulated values for this test are given in [5]; from this we find that when $V > 2$ the neighborhood fails the isotropy test at an upper percentage point of 0.01. The value of V for the pixel on the wing boundary is 2.3; hence it is labelled "not isotropic." By contrast, a pixel chosen in the rocky ground area gives us $V = 0.8$, and is therefore labelled "isotropic."

3.3 Computing windowed texton histogram

Pairwise texture similarities can be computed by comparing windowed texton histograms, where the windows are centered around the two pixels being compared. Each histogram has K bins, one for each texton channel. The value of the kth histogram bin for a pixel i is found by counting how many pixels fall inside a box[5] centered around pixel i in texton channel k. Thus the histogram represents texton frequencies in a local neighborhood. We can write this as

$$h_i(k) = \sum_{j \in \mathcal{W}(i)} I[T(j) = k]$$

where $\mathcal{W}(i)$ is the set of pixels in the box centered on i, $I[\cdot]$ is the indicator function, and $T(j)$ returns the texton assigned to pixel j.

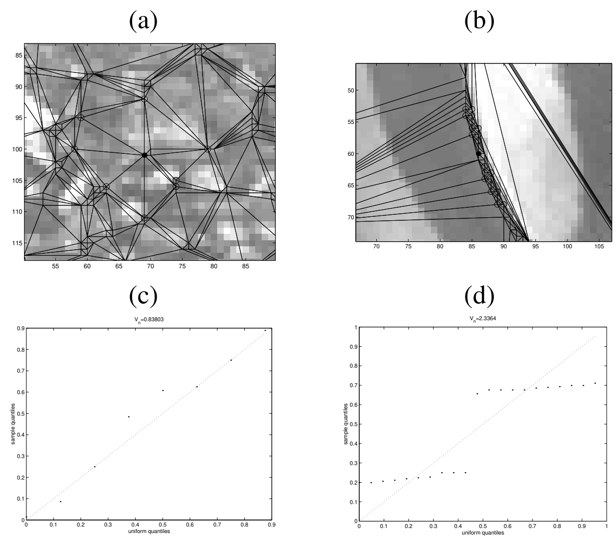

Figure 5. (a,b) Illustration of the Delaunay neighbors in zoomed in regions of two different texton channels on the penguin image. The filled circle marks pixel i and the open circles mark its neighbors. (c,d) Visualization of the computation of the statistical test for isotropy. The test value for (a) is $V = 0.8$ while that of (b) is $V = 2.3$. Using a significance level of 0.01, the isotropy hypothesis is rejected when $V > 2$.

4 Cue combination strategy

The obvious approach to cue integration (integrating information from both contours and textures) is to define the weight between pixels i and j as the product of the contribution from texture and that from contour: $W_{ij} = W_{ij}^{IC} \times W_{ij}^{TX}$. We have to be careful to avoid the problems listed in the Introduction (§1.2) by suitably gating the cues. The spirit of the gating method is to make each cue "harmless" in the vicinity of regions where one or the other cue should not be operating. This will manifest itself as suppression of oriented energy inside regions when computing the contour weights, and suppression of textons along boundaries when computing the texture weights.

Here is our way of defining the individual components and combining them:

[4] We exclude immediate 8 grid neighbors of i as they definitely do not constitute samples independent from i.

[5] At this stage we have not implemented scale selection and just use boxes of size 11×11 pixels.

4.1 Contour

The definition of W_{ij}^{IC} is adopted from that defined in [12]. Contour information in an image is computed "softly" through orientation energies (OE) from elongated quadrature filter pairs. We introduce a slight modification here to allow for exact sub-pixel localization of the contour by finding the local maxima in the orientation energy perpendicular to the contour orientation [16]. The confidence of this contour is given by the orientation energy. W_{ij}^{IC} is then defined as follows:

$$W_{ij}^{IC} = \exp(-\max_{x \in M_{ij}} OE(x)/\sigma_{IC})$$

where M_{ij} is the set of local maxima along the line joining pixels i and j. In words, two pixels will have a weak link between them if there is a strong local maximum of orientation energy along the line joining the two pixels. On the contrary, if there is little energy, for example in a constant brightness region, the link between the two pixels will be strong.

4.2 Measuring Texture Similarity

Pairwise texture similarities can be computed by comparing windowed texton histograms computed using the technique described previously (§3.3). A number of methods are available for comparing histograms; among them a simple and effective choice is the χ^2 test, defined as

$$\chi^2(h_i, h_j) = \frac{1}{2}\sum_{k=1}^{K} \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)}$$

where h_i and h_j are the two histograms. For a comparison of the χ^2 test versus other texture similarity measures, the readers are referred to [17].

W_{ij}^{TX} is defined using the χ^2 distance between texton histograms at pixels i and j:

$$W_{ij}^{TX} = \exp(-\chi^2(h_i, h_j)/\sigma_{TX})$$

4.3 Cue Combination

Cue combination is accomplished in the following steps.

1. We first compute the isotropic measure $\alpha(i)$ and unisotropic measure $\beta(i)$ at each point, i, in the image. One can think of $\alpha(i)$ as the 1D-ness measure, while $\beta(i)$ as the texture-ness measure. Define

$$\begin{aligned} \alpha(i) &= \text{sigmoid}(V(i), threshold) \\ \beta(i) &= 1 - \text{sigmoid}(V(i), threshold) \end{aligned}$$

To simplify the computaton, a discrete version of the sigmoid is used in our experiments. We select the threshold V at 2.0 at each pixel as discussed in §3.2. Figure 6 shows one such computed α and β map on a tiger image.

(a) (b)

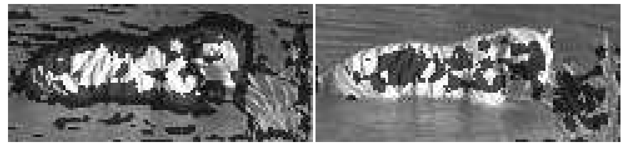

Figure 6. Subplot (a) and (b) shows the computed α and β measure in a tiger image. The α and β values are thresholded at 2.0, and masked on the original image.

2. We then compute the texture feature descriptor at each pixel of the image, gated by the function $\alpha(i)$. The main idea is to ignore any neighboring pixels which are near a region boundary in the histogram computation. Define the gated histogram as:

$$h_i^\alpha(k) = \frac{1}{Z(i)} \sum_{j \in \mathcal{W}(i)} [1 - \alpha(j)] I[T(j) = k]$$

where $Z(i) = \sum_{j \in \mathcal{W}(i)} (1 - \alpha(j))$. This definition of texture histogram avoids the problem of texture comparison near object boundaries. At intensity boundaries, the boundaries themselves can no longer be used as features, and therefore will not form groups on their own. This definition of the gated histogram also has a desirable behavior near texture boundaries: it tends to pool information from the correct side of the region. Figure 7 illustrates this point. From the gated texture

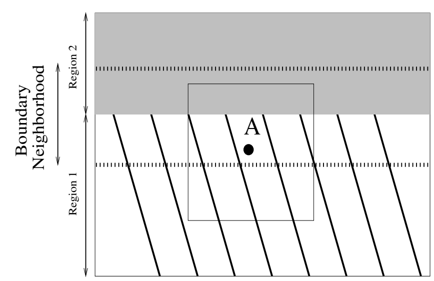

Figure 7. At the texture boundary, the proposed gated histogram tends to pool information from the "correct region". Take the point marked "A" in the boundary region between the dashed lines as an example. By masking out all features in the boundary neighborhood, the texture histogram computed for "A" will contain only the information from region 1. This avoids the problem of having corrupted texture histogram information as we get closer to the region boundary.

histogram, we can compute the pair-wise texture similarity, $W^{TX}(i,j)$ as :

$$W^{TX}(i,j) = \exp(-\chi^2(h_i^\alpha, h_j^\alpha)/\sigma_{TX})$$

As we move deeper into the boundary region, we have fewer points in the neighborhood to compute the histogram. In that case, the histogram difference becomes less reliable, and therefore should be discounted. We define the reliability measure for each histogram measure at pixel $p(i) = \text{sigmoid}(Z(i), threshold_p)$. In our experiments, the $threshold_p$ is set to $0.05 * |\mathcal{W}(i)|$.

3. In parallel to the texture computation, the intervening contour cue gated by the texture-ness can be used to group/segment pixels. The computation is same as in §4.1, except the filter energy is suppressed by textureness measure $\beta(i)$.

$$W^{IC}(i,j) = \exp(-\max_{x \in l_{ij}, \beta(x) < 1.0} OE(x)/\sigma_{IC})$$

4. Let the two pair-wise feature distance functions computed in the two previous steps be $W^{tex}(i,j)$ and $W^{IC}(i,j)$, from the texture cue and intervening contour cue respectively. Since the test for isotropy is purely a local one, one expects the α and β function to misfire sometimes. By combining the two cues, and applying global grouping algorithm to this data, we hope to "smooth out" these errors in the α and β estimates. The rule we have for combining two cues is:

$$W(i,j) = [W^{TX}(i,j)]^{p(i,j)}[W^{IC}(i,j)]$$

where $p(i,j) = \min(p(i), p(j))$ is the significance of the histogram comparison between pixels i and j.

5. Applying grouping algorithm to the combined pairwise similarity measure to obtain the final segmentation. We used the normalized cut algorithm for this step [18]. The global nature of the normalized cut algorithm help us overcome the errors in the local α and β computation.

5 Results

We have run our algorithm on a variety of natural images. Figures 8 and 9 show typical segmentation results. In all the cases, the regions are cleanly separated from each other using combined texture and contour cues.

Grouping based on each of the cues alone would result in severe artifacts: In Figure 8a, the contours on the penguin would form isolated groups using the texture cue. Similar problems would occur at the intensity boundaries in 8b and 8c. Grouping based on contour information alone would result in over-fragmentation of the pebbles in 8a and 9a, and the tiger body in 8c. On the other hand, in Figure 8b the lower arm can not be separated from the upper arm without using contour information.

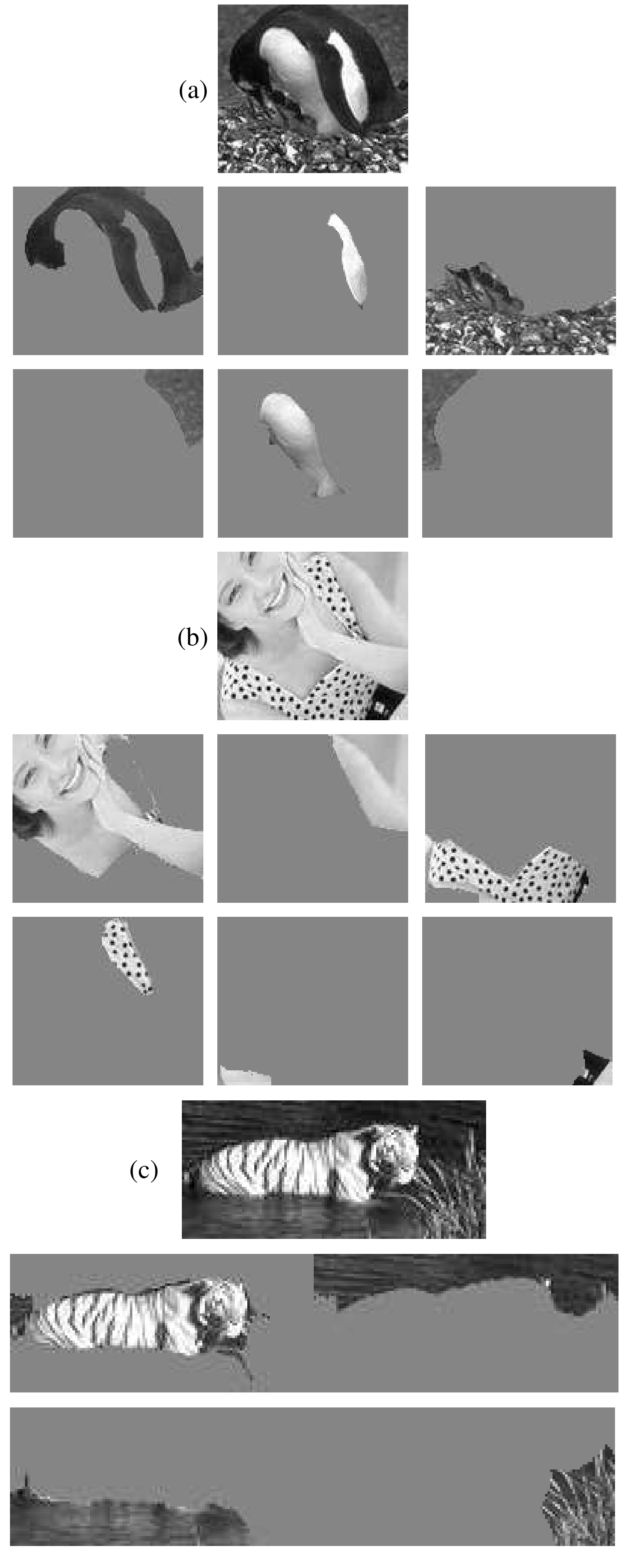

Figure 8. Segmentation results of three images using combined texture and intervening contour cue. The image regions are cleanly segmented from each other using the combined cues. In all three cases, grouping by each of the cues alone will not produce the right results.

Acknowledgement

This research was supported by (ARO) DAAH04-96-1-0341, the Digital Library Grant IRI-9411334, NSF Graduate Fellowships to SB and JS and a Berkeley Fellowship to TL.

References

[1] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color- and texture-based image segmentation using em and its application to content-based image retrieval. In *Proc. 6th Int. Conf. Computer Vision*, pages 675–82, Bombay, India, Jan. 1998.

[2] T. Binford. Inferring surfaces from images. *Artificial Intelligence*, 17(1-3):205–44, Aug. 1981.

[3] R. DeValois and K. DeValois. *Spatial Vision*. Oxford University Press, 1988.

[4] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[5] N. Fisher. *Statistical Analysis of Circular Data*. John Wiley & Sons, 1973.

[6] I. Fogel and D. Sagi. Gabor filters as texture discriminator. *Biological Cybernetics*, 61:103–13, 1989.

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

[8] D. Jones and J. Malik. Computational framework to determining stereo correspondence from a set of linear spatial filters. *Image and Vision Computing*, 10(10):699–708, Dec. 1992.

[9] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–7, March 1981.

[10] H. Knutsson and G. Granlund. Texture analysis using two-dimensional quadrature filters. In *Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pages 206–13, Oct. 1983.

[11] J. Koenderink and A. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55(6):367–75, 1987.

[12] T. Leung and J. Malik. Contour continuity in region-based image segmentation. In H. Burkhardt and B. Neumann, editors, *Proc. Euro. Conf. Computer Vision*, volume 1, pages 544–59, Freiburg, Germany, June 1998. Springer-Verlag.

[13] T. Leung and J. Malik. Recognizing surfaces using three-dimensional textons. In *Proc. Int. Conf. Computer Vision*, Corfu, Greece, Sep. 1999.

[14] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *J. Opt. Soc. America A*, 7(5):923–32, May 1990.

[15] I. Nabney and C. Bishop. Netlab neural network software. `http://www.ncrg.aston.ac.uk/netlab/`.

[16] P. Perona and J. Malik. Detecting and localizing edges composed of steps, peaks and roofs. In *Proc. 3rd Int. Conf. Computer Vision*, pages 52–7, Osaka, Japan, Dec 1990.

[17] J. Puzicha, T. Hofmann, and J. Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. In *Proc. IEEE Conf. Comp. Vis. and Pat. Rec.*, pages 267–72, Jun. 1997.

[18] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 731–7, June 1997.

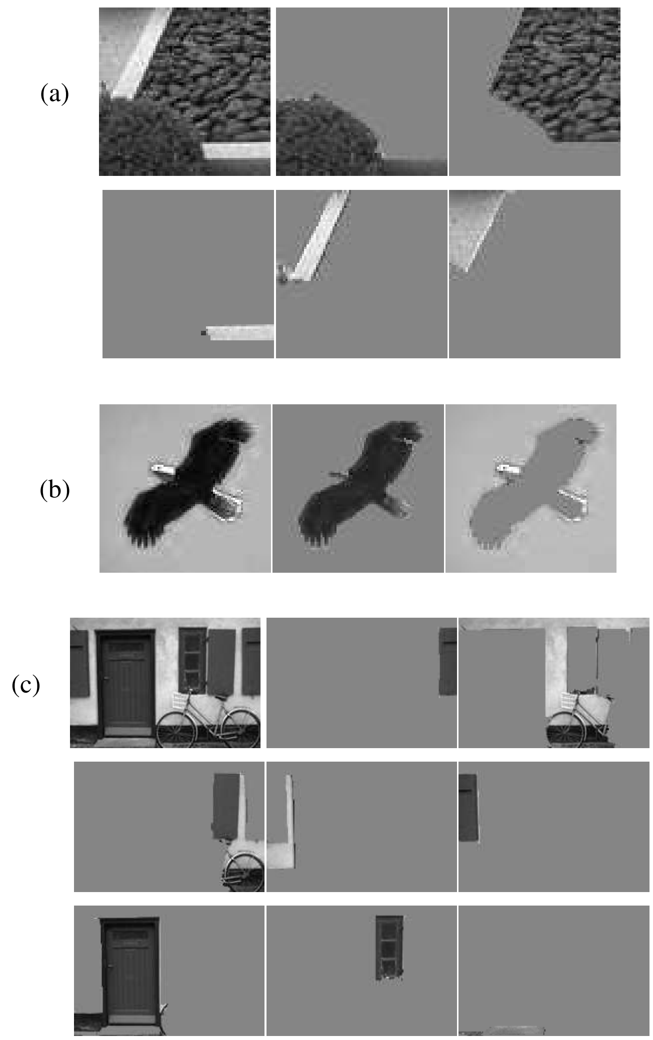

Figure 9. Segmentation results of three images using combined texture and intervening contour cue. Image (a) has regions defined by texture and contour, while images (b) and (c) only have regions defined by contour information alone. The proposed cue combination behaves correctly in both cases.