Submitted to the International Journal of Computer Vision, Dec 1999
A shorter version appeared in the International Conference in Computer Vision, Corfu, Greece, Sep 1999

Contour and Texture Analysis for Image Segmentation

Jitendra Malik, Serge Belongie, Thomas Leung and Jianbo Shi*
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720-1776

Abstract. This paper provides an algorithm for partitioning grayscale images into disjoint regions of coherent brightness and texture. Natural images contain both textured and untextured regions, so the cues of contour and texture differences are exploited simultaneously. Contours are treated in the *intervening contour* framework, while texture is analyzed using *textons*. Each of these cues has a domain of applicability, so to facilitate cue combination we introduce a gating operator based on the texturedness of the neighborhood at a pixel. 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.

Keywords: segmentation, texture, grouping, cue integration, texton, normalized cut

1. Introduction

To humans, an image is not just a random collection of pixels; it is a meaningful arrangement of regions and objects. Figure 1 shows a variety of images. Despite the large variations of these images, humans have no problem interpreting them. We can agree about the different regions in the images and recognize the different objects. Human visual grouping was studied extensively by the Gestalt psychologists in the early part of the large20^{th} century (Wertheimer, 1938). They identified several factors that lead to human perceptual grouping: similarity, proximity, continuity, symmetry, parallelism, closure and familiarity. In computer vision, these factors have been used as guidelines for many grouping algorithms.

The most studied version of grouping in computer vision is image segmentation. Image segmentation techniques can be classified into two broad families–(1) region-based, and (2) contour-based approaches. Region-based approaches try to find partitions of the image pixels into sets corresponding to coherent image properties such as brightness, color and texture. Contour-based approaches usually start with a first

* Currently with the Department of Computer Science, Carnegie Mellon University

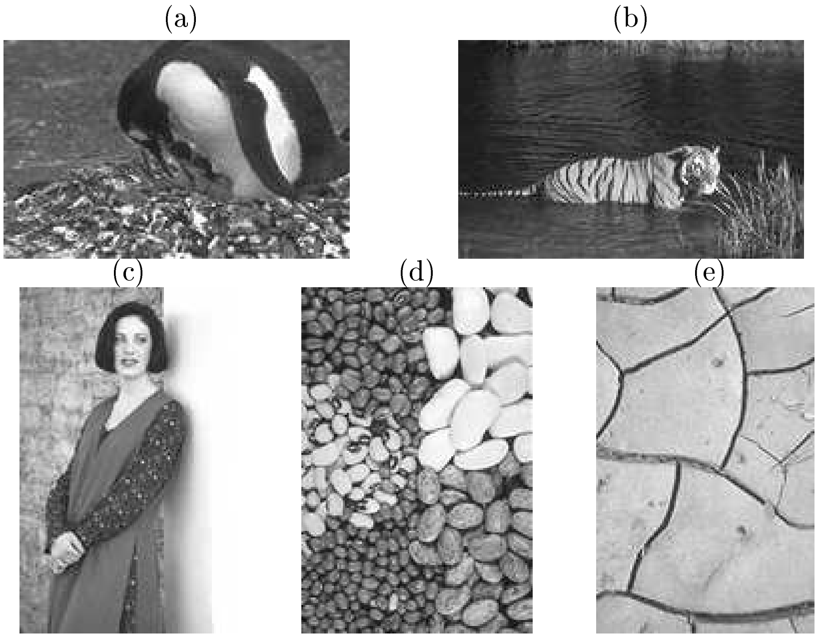

Figure 1. Some challenging images for a segmentation algorithm. Our goal is to develop a single grouping procedure which can deal with all these types of images.

stage of edge detection, followed by a linking process that seeks to exploit curvilinear continuity.

These two approaches need not be that different from each other. Boundaries of regions can be defined to be contours. If one enforces closure in a contour-based framework (Elder and Zucker, 1996; Jacobs, 1996) then one can get regions from a contour-based approach. The difference is more one of emphasis and what grouping factor is coded more naturally in a given framework.

A second dimension on which approaches can be compared is local vs. global. Early techniques, in both contour and region frameworks, made local decisions–in the contour framework this might be declaring an edge at a pixel with high gradient, in the region framework this might be making a merge/split decision based on a local, greedy strategy.

Region-based techniques lend themselves more readily to defining a global objective function (for example, Markov random fields (Geman and Geman, 1984) or variational formulations (Mumford and Shah, 1989)). The advantage of having a global objective function is that decisions are made only when information from the whole image is taken into account at the same time.

In contour-based approaches, often the first step of edge detection is done locally. Subsequently efforts are made to improve results by a global linking process that seeks to exploit curvilinear continuity. Examples include dynamic programming (Montanari, 1971), relaxation approaches (Parent and Zucker, 1989), saliency networks (Sha'ashua and Ullman, 1988), stochastic completion (Williams and Jacobs, 1995). A criticism of this approach is that the edge/no edge decision is made prematurely. To detect extended contours of very low contrast, a very low threshold has to be set for the edge detector. This will cause random edge segments to be found everywhere in the image, making the task of the curvilinear linking process unnecessarily harder than if the raw contrast information was used.

A third dimension on which various segmentation schemes can be compared is the class of images for which they are applicable. As suggested by Figure 1, we have to deal 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 and rock region in Figure 1(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. This problem is illustrated in Figure 2. 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 e.g. (Binford, 1981)), 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 3. 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

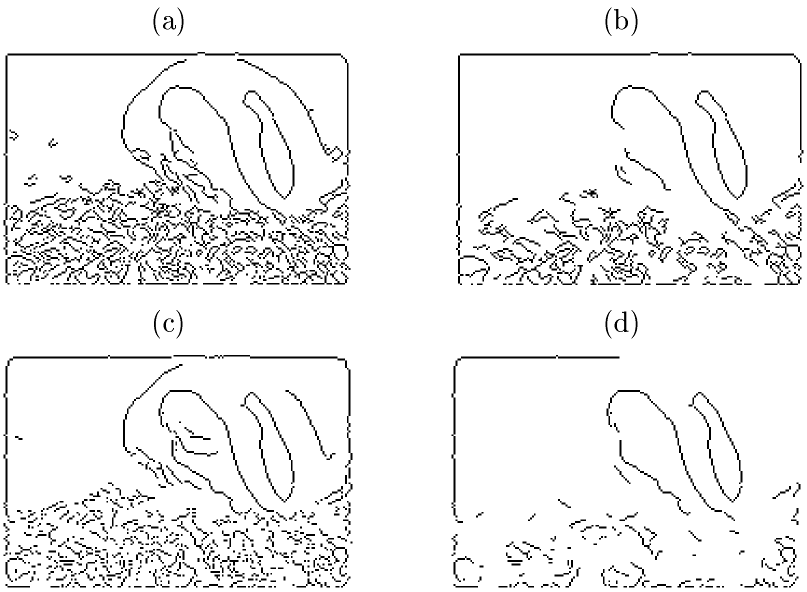

Figure 2. Demonstration of texture as a problem for the contour process. Each image shows the edges found with a Canny edge detector for the penguin image using different scales and thresholds: (a) fine scale, low threshold, (b) fine scale, high threshold, (c) coarse scale, low threshold, (d) coarse scale, high threshold. A parameter setting that preserves the correct edges while suppressing spurious detections in the textured area is not possible.

Figure 3. 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-based algorithm (Belongie et al., 1998).

filter outputs as texture descriptor. Figure 3 (b) shows the actual groups found by an EM-based algorithm using an alternative color/texture descriptor (Belongie et al., 1998).

1.1. Desiderata of a Theory of Image Segmentation

At this stage, we are ready to summarize our desired attributes for a theory of image segmentation.

1. It should deal with general images. Regions with or without texture should be processed in the same framework, so that the cues of contour and texture differences can be simultaneously exploited.

2. In terms of contour, the approach should be able to deal with boundaries defined by brightness step edges as well as lines (as in a cartoon sketch).

3. Image regions could contain texture which could be regular such as the polka dots in Figure 1(c), stochastic as in the snow and rock region in (a) or anywhere in between such as the tiger stripes in (b). A key question here is that one needs an automatic procedure for scale selection. Whatever one's choice of texture 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, stochastic, or intermediate cases — in a seamless way.

1.2. Introducing Textons

Julesz introduced the term *texton*, analogous to a phoneme in speech recognition, nearly 20 years ago (Julesz, 1981) 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 that 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 (on the order of 40 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 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 (Leung and Malik, 1999) 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.

1.3. Summary of our approach

We pursue image segmentation in the framework of Normalized Cuts introduced by (Shi and Malik, 1997). The image is considered to be a weighted graph where the nodes i and j are pixels and edge weights, W_{ij}, denote a local measure of similarity between the two pixels. Grouping is performed by finding eigenvectors of the Normalized Laplacian of this graph (§3). The fundamental issue then is that of specifying the edge weights W_{ij}; we rely on normalized cuts to go from these local measures to a globally optimal partition of the image.

The algorithm analyzes the image using the two cues of contour and texture. The local similarity measure between pixels i and j due to the contour cue, W_{ij}^{IC}, is computed in the *intervening contour* framework of (Leung and Malik, 1998) using peaks in contour orientation energy (§2 and §4.1). Texture is analysed using textons (§2.1). Appropriate local scale is estimated from the texton labels. A histogram of texton densities is used as the texture descriptor. Similarity, W_{ij}^{TX}, is measured using the χ^2 test on the histograms (§4.2). The edge weights W_{ij} combining both contour and texture information are specified by gating each of the two cues with a texturedness measure (§4.3)

In (§5), we present the practical details of going from the eigenvectors of the normalized Laplacian matrix of the graph to a partition of the image. Results from the algorithm are presented in (§6).

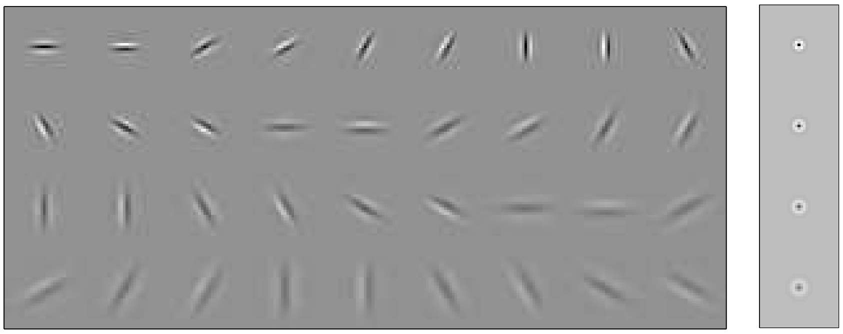

Figure 4. Left: Filter set f_i consisting of 2 phases (even and odd), 3 scales (spaced by half-octaves), and 6 orientations (equally spaced from 0 to π). The basic filter is a difference-of-Gaussian quadrature pair with 3 : 1 elongation. Right: 4 scales of center-surround filters. Each filter is L_1-normalized for scale invariance.

2. Filters, Composite Edgels, and Textons

Since the 1980s, many approaches have been proposed in the computer vision literature that start by convolving the image with a bank of linear spatial filters f_i tuned to various orientation and spatial frequencies (Knutsson and Granlund, 1983; Koenderink and van Doorn, 1987; Fogel and Sagi, 1989; Malik and Perona, 1990). (See Figure 4 for an example of such a filter set.)

These approaches were inspired by models of processing in the early stages of the primate visual system (e.g. (DeValois and DeValois, 1988)). The filter kernels f_i are models of receptive fields of simple cells in visual cortex. To a first approximation, we can classify them into three categories:

1. Cells with radially symmetric receptive fields. The usual choice of f_i is a Difference of Gaussians (DOG) with the two Gaussians having different values of σ. Alternatively, these receptive fields can also be modeled as the Laplacian of Gaussian.

2. Oriented odd-symmetric cells whose receptive fields can be modeled as rotated copies of a horizontal oddsymmetric receptive field. A suitable point spread function for such a receptive field is $f(x, y) = G'_{\sigma_1}(y)G_{\sigma_2}(x)$ where $G_\sigma(x)$ represents a Gaussian with standard deviation σ. The ratio $\sigma_2 : \sigma_1$ is a measure of the elongation of the filter.

3. Oriented even-symmetric cells whose receptive fields can be modeled as rotated copies of a horizontal evensymmetric receptive field.

A suitable point spread function for such a receptive field is

$$f(x,y) = G''_{\sigma_1}(y)G_{\sigma_2}(x)$$

The use of Gaussian derivatives (or equivalently, differences of offset Gaussians) for modeling receptive fields of simple cells is due to (Young, 1985). One could equivalently use Gabor functions. Our preference for Gaussian derivatives is based on their computational simplicity and their natural interpretation as 'blurred derivatives' (Koenderink and van Doorn, 1987; Koenderink and van Doorn, 1988).

The oriented filterbank used in this work, depicted in Figure 4, is based on rotated copies of a Gaussian derivative and its Hilbert transform. More precisely, let $f_1(x,y) = G''_{\sigma_1}(y)G_{\sigma_2}(x)$ and $f_2(x,y)$ equal the Hilbert transform of $f_1(x,y)$ along the y axis:

$$\begin{aligned} f_1(x,y) &= \frac{d^2}{dy^2}\left(\frac{1}{C}\exp\left(\frac{y^2}{\sigma^2}\right)\exp\left(\frac{x^2}{\ell^2\sigma^2}\right)\right) \\ f_2(x,y) &= Hilbert(f_1(x,y)) \end{aligned}$$

where σ is the scale, ℓ is the aspect ratio of the filter, and C is a normalization constant. (The use of the Hilbert transform instead of a first derivative makes f_1 and f_2 an exact quadrature pair.) The radially symmetric portion of the filterbank consists of Difference-of-Gaussian kernels. Each filter is zero-mean and L_1 normalized for scale invariance (Malik and Perona, 1990).

Now suppose that the image is convolved with such a bank of linear filters. We will refer to the collection of response images $I * f_i$ as the *hypercolumn transform* of the image.

Why is this useful from a computational point of view? The vector of filter outputs $I * f_i(x_0, y_0)$ characterizes the image *patch* centered at x_0, y_0 by a set of values at a *point*. This is similar to characterizing an analytic function by its derivatives at a point – one can use a Taylor series approximation to find the values of the function at neighboring points. As pointed out by (Koenderink and van Doorn, 1987), this is more than an analogy, because of the commutativity of the operations of differentiation and convolution, the receptive fields described above are in fact computing 'blurred derivatives'. We recommend (Koenderink and van Doorn, 1987; Koenderink and van Doorn, 1988; Jones and Malik, 1992; Malik and Perona, 1992) for a discussion of other advantages of such a representation.

The hypercolumn transform provides a convenient front end for contour and texture analysis:

– *Contour.* In computational vision, it is customary to model brightness edges as step edges and to detect them by marking locations corresponding to the maxima of the outputs of odd-symmetric filters (e.g. (Canny, 1986)) at appropriate scales. However, it should be noted that step edges are an inadequate model for the discontinuities in the image that result from the projection of depth or orientation discontinuities in physical scene. Mutual illumination and specularities are quite common and their effects are particularly significant in the neighborhood of convex or concave object edges. In addition, there will typically be a shading gradient on the image regions bordering the edge. As a consequence of these effects, real image edges are not step functions but more typically a combination of steps, peak and roof profiles. As was pointed out in (Perona and Malik, 1990), the oriented energy approach (Knutsson and Granlund, 1983; Morrone and Owens, 1987; Morrone and Burr, 1988) can be used to detect and localize correctly these composite edges.

The oriented energy, also known as the "quadrature energy," at angle 0° is defined as:

$$OE_{0^\circ} = (I * f_1)^2 + (I * f_2)^2$$

OE_{0° has maximum response for horizontal contours. Rotated copies of the two filter kernels are able to pick up composite edge contrast at various orientations.

Given OE_θ, we can proceed to localize the composite edge elements (edgels) using oriented nonmaximal suppression. This is done for each scale in the following way. At a generic pixel q, let $\theta^* = \arg\max OE_\theta$ denote the dominant orientation and OE^* the corresponding energy. Now look at the two neighboring values of OE^* on either side of q along the line through q perpendicular to the dominant orientation. The value OE^* is kept at the location of q only if it is greater than or equal to each of the neighboring values. Otherwise it is replaced with a value of zero.

Noting that OE^* ranges between 0 and infinity, we convert it to a probability-like number between 0 and 1 as follows:

$$p_{con} = 1 - \exp(-OE^*/\sigma_{IC}) \tag{1}$$

σ_{IC} is related to oriented energy response purely due to image noise. We use $\sigma = 0.02$ in this paper. The idea is that for any contour with $OE^* \gg \sigma_{IC}$, $p_{con} \approx 1$.

- *Texture.* As the hypercolumn transform provides a good local descriptor of image patches, the boundary between differently textured regions may be found by detecting curves across which there is a significant gradient in one or more of the components of the hypercolumn transform. For an elaboration of this approach, see (Malik and Perona, 1990).

Malik and Perona relied on averaging with large kernels to smooth away spatial variation for filter responses within regions of texture. This process loses a lot of information about the distribution of filter responses; a much better method is to represent the neighborhood around a pixel by a histogram of filter outputs (Heeger and Bergen, 1995; Puzicha et al., 1997). While this has been shown to be a powerful technique, it leaves open two important questions. Firstly, there is the matter of what size window to use for pooling the histogram – the integration scale. Secondly, these approaches only make use of marginal binning, thereby missing out on the informative characteristics that joint assemblies of filter outputs exhibit at points of interest. We address each of these questions in the following section.

2.1. Textons

Though the representation of textures using filter responses is extremely versatile, one might say that it is overly redundant (each pixel value is represented by N_{fil} real-valued filter responses, where N_{fil} is 40 for our particular filter set). 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 4. Each pixel is now transformed to a N_{fil} dimensional vector of filter responses. These vectors are clustered using K-means. 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].

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 (Jones and Malik, 1992). 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 5(b) and 6(b). These are our textons. The textons represent assemblies of filter outputs that are characteristic of the local image structure present in the image.

Looking at the polka-dot example, we find that many of 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 5(c). Each subimage shows the pixels in the image that are mapped to the corresponding texton in Figure 5(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 6. 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.

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?"

2.1.1. *Local Scale and Neighborhood 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 is 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[3]. The idea is that while textons are being associated with pixels, since they correspond to assemblies of filter outputs, it is better

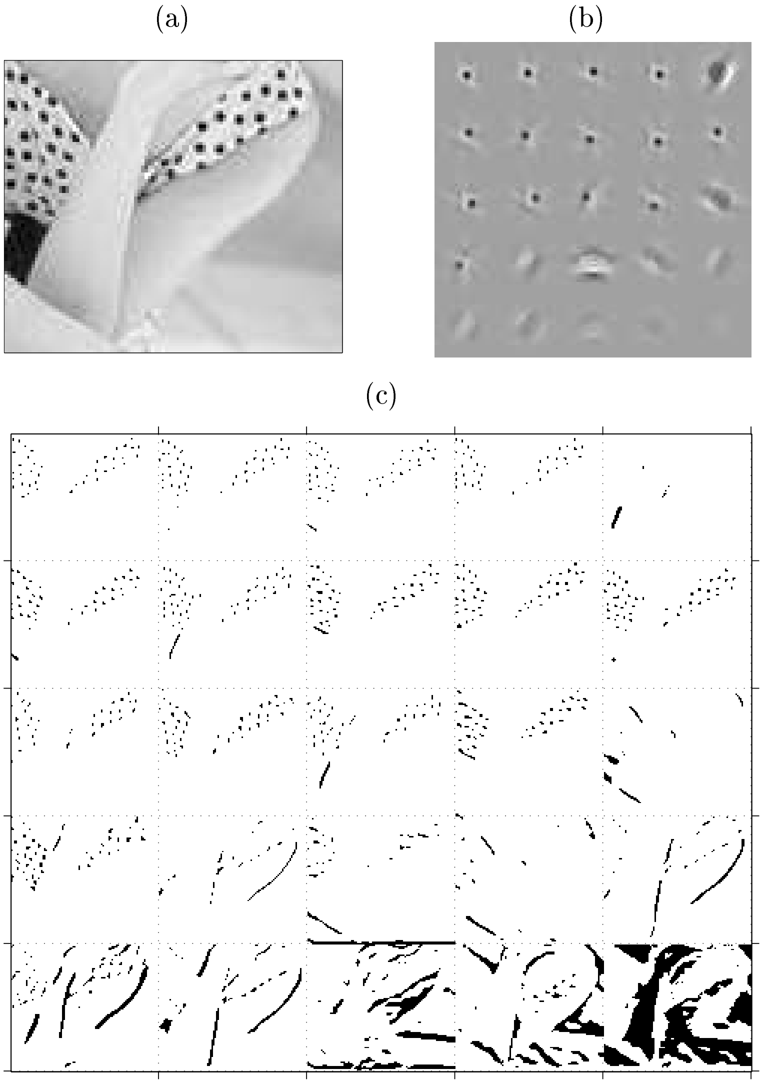

Figure 5. (a) Polka-dot image. (b) Textons found via K-means with $K = 25$, sorted in decreasing order by norm. (c) Mapping of pixels to the texton channels. The dominant structures captured by the textons are translated versions of the dark spots. We also see textons corresponding to faint oriented edge and bar elements. Notice that some channels contain activity inside a textured region or along an oriented contour and nowhere else.

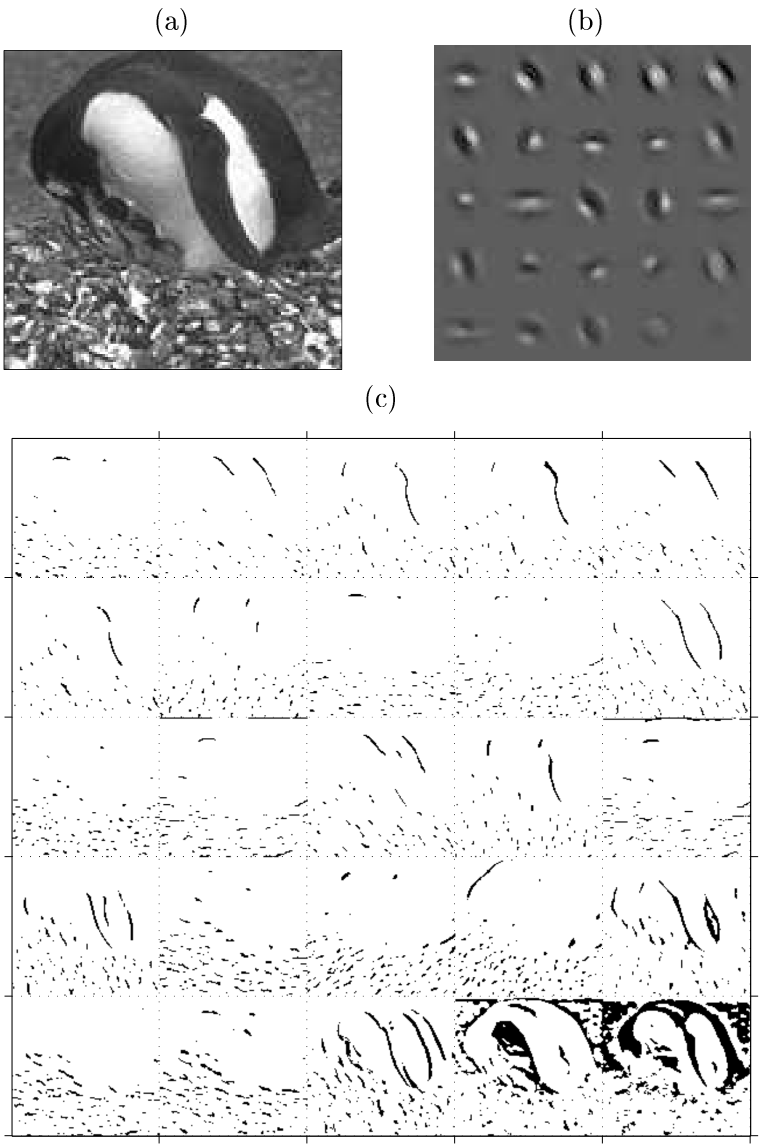

Figure 6. (a) Penguin image. (b) Textons found via K-means with $K = 25$, sorted in decreasing order by norm. (c) Mapping of pixels to the texton channels. Among the textons we see edge elements of varying orientation and contrast along with elements of the stochastic texture in the rocks.

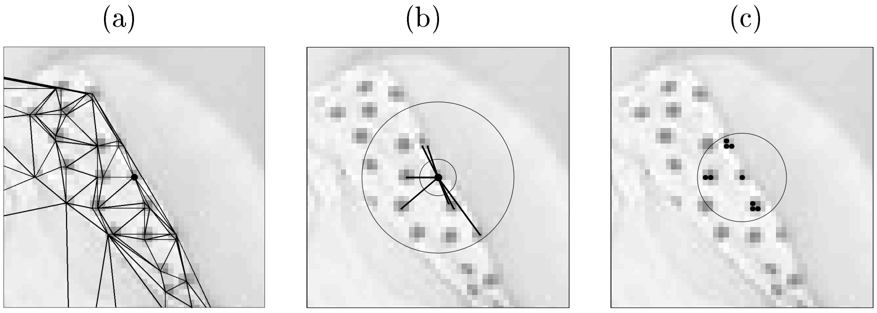

Figure 7. Illustration of scale selection. (a) Closeup of Delaunay triangulation of pixels in a particular texton channel for polka dot image. (b) Neighbors of thickened point for pixel at center. The thickened point lies within inner circle. Neighbors are restricted to lie within outer circle. (c) Selected scale based on median of neighbor edge lengths, shown by circle, with all pixels falling inside circle marked with dots.

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[4]. The intuition is that these will be pixels in spatially neighboring texels. Compute the distances of all these pixels to i; the median of these constitutes a robust local measure of inter-texel distance. We define the local scale $\alpha(i)$ to be 1.5 times this median distance.

In Figure 7(a), the Delaunay triangulation of a zoomed-in portion of one of the texton channels in the polka-dot dress of Figure 5(a) 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.

The local scale $\alpha(i)$ is based just on the texton channel for the texton at i. Since neighboring pixels should have similar scale and could be drawn from other texton channels, we can improve the estimate of scale by median filtering of the scale image.

2.1.2. *Computing windowed texton histograms*

Pairwise texture similarities will be computed by comparing windowed texton histograms. We define the window $\mathcal{W}(i)$ for a generic pixel i as the axis-aligned square of radius $\alpha(i)$ centered on pixel i.

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 in texton channel k fall inside the window $\mathcal{W}(i)$. Thus the histogram represents texton frequencies in a local neighborhood. We can write this as

$$h_i(k) = \sum_{j \in \mathcal{W}(i)} I[T(j) = k] \tag{2}$$

where $I[\cdot]$ is the indicator function and $T(j)$ returns the texton assigned to pixel j.

3. The Normalized Cut Framework

In the Normalized Cut framework (Shi and Malik, 1997), Shi and Malik formulate visual grouping as a graph partitioning problem. The nodes of the graph are the entities that we want to partition; for example, in image segmentation, they are the pixels. The edges between two nodes correspond to the *strength* with which these two nodes belong to one group; again, in image segmentation, the edges of the graph corresponds to how much two pixels agree in brightness, color, etc. Intuitively, the criterion for partitioning the graph will be to minimize the sum of weights of connections *across* the groups and maximize the sum of weights of connections *within* the groups.

Let $\boldsymbol{G} = \{\boldsymbol{V}, \boldsymbol{E}\}$ be a weighted undirected graph, where $\boldsymbol{V}$ are the nodes and $\boldsymbol{E}$ are the edges. Let $\boldsymbol{A}, \boldsymbol{B}$ be a partition of the graph: $\boldsymbol{A} \cup \boldsymbol{B} = \boldsymbol{V}, \boldsymbol{A} \cap \boldsymbol{B} = \emptyset$. In graph theoretic language, the similarity between these two groups is called the *cut*:

$$cut(\boldsymbol{A}, \boldsymbol{B}) = \sum_{i \in A, j \in B} W_{ij}$$

where W_{ij} is the weight on the edge between nodes i and j. Shi and Malik proposed to use a *normalized* similarity criterion to evaluate a partition. They call it the *normalized cut*:

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

where $assoc(\boldsymbol{A}, \boldsymbol{V}) = \sum_{i \in \boldsymbol{A}, k \in \boldsymbol{V}} W_{ik}$ is the total connection from nodes in $\boldsymbol{A}$ to all the nodes in the graph. For more discussion of this criterion, please refer to (Shi and Malik, 1997; Shi and Malik, 2000).

One key advantage of using the normalized cut is that a good approximation to the optimal partition can be computed very efficiently[5]. Let $\boldsymbol{W}$ be the association matrix, i.e. W_{ij} is the weight between nodes i and j in the graph. Let $\boldsymbol{D}$ be the diagonal matrix such that $D_{ii} = \sum_j W_{ij}$, i.e. D_{ii} is the sum of the weights of all the connections to node i. Shi and Malik showed that the optimal partition can be found by computing:

$$\begin{aligned} \boldsymbol{y} &= \arg\min Ncut \\ &= \arg\min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y}} \end{aligned} \tag{3}$$

where $\boldsymbol{y} = \{a, b\}^N$ is a binary indicator vector specifying the group identity for each pixel, i.e. $y_i = a$ if pixel i belongs to group $\boldsymbol{A}$ and $y_j = b$ if pixel j belongs to $\boldsymbol{B}$. N is the number of pixels. Notice that the above expression is a Rayleigh quotient. If we relax $\boldsymbol{y}$ to take on real values (instead of two discrete values), we can optimize Equation 3 by solving a generalized eigenvalue system. Efficient algorithms with polynomial running time are well-known for solving such problems.

The process of transforming the vector $\boldsymbol{y}$ into a discrete bipartition and the generalization to more than two groups is discussed in (§5).

4. Defining the Weights

The quality of a segmentation based on Normalized Cuts or any other algorithm based on pairwise similarities fundamentally depends on the weights – the W_{ij}'s – that are provided as input. The weights should be large for pixels that should belong together and small otherwise. We now discuss our method for computing the W_{ij}'s. Since we seek to combine evidence from two cues, we will first discuss the computation of the weights for each cue in isolation, and then describe how the two weights can be combined in a meaningful fashion.

4.1. Images without Texture

Consider for the moment the "cracked earth" image in Figure 1(e). Such an image contains no texture and may be treated in a framework based solely on contour features. The definition of the weights in this case, which we denote W_{ij}^{IC}, is adopted from the *intervening contour* method introduced in (Leung and Malik, 1998).

Figure 8 illustrates the intuition behind this idea. On the left is an image. The middle figure shows a magnified part of the original image. On the right is the orientation energy. There is an extended contour separating p_3 from p_1 and p_2. Thus, we expect p_1 to be much more strongly related to p_2 than p_3. This intuition carries over in our definition of dissimilarity between two pixels: if the orientation energy along the line between two pixels is strong, the dissimilarity between these pixels should be high (and W_{ij} should be low).

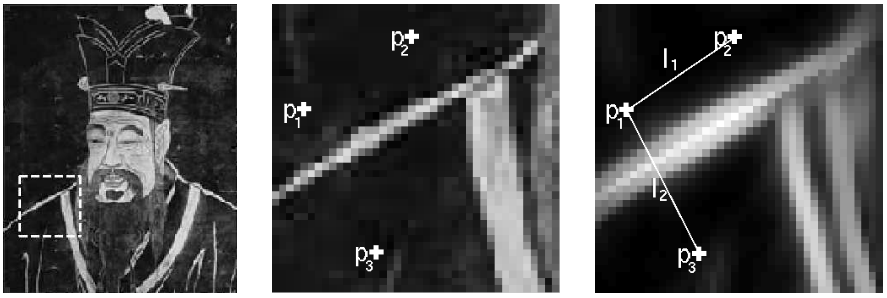

Figure 8. Left: the original image. Middle: part of the image marked by the box. The intensity values at pixels p_1, p_2 and p_3 are similar. However, there is a contour in the middle, which suggests that p_1 and p_2 belong to one group while p_3 belongs to another. Just comparing intensity values at these three locations will mistakenly suggest that they belong to the same group. Right: orientation energy. Somewhere along l_2, the orientation energy is strong which correctly proposes that p_1 and p_3 belong to two different partitions, while orientation energy along l_1 is weak throughout, which will support the hypothesis that p_1 and p_2 belong to the same group.

Contour information in an image is computed "softly" through orientation energy (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 (Perona and Malik, 1990). The orientation energy gives the confidence of this contour. W_{ij}^{IC} is then defined as follows:

$$W_{ij}^{IC} = 1 - \max_{x \in M_{ij}} p_{con}(x)$$

where M_{ij} is the set of local maxima along the line joining pixels i and j. Recall from (§2) that $p_{con}(x), 0 < p_{con} < 1$, is nearly 1 whenever the orientated energy maximum at x is sufficiently above the noise level. 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. Contours measured at different scales can be taken into account by computing the orientation energy maxima at various scales and setting p_{con} to be the maximum over all the scales at each pixel.

4.2. Images that are Texture Mosaics

Now consider the case of images wherein all of the boundaries arise from neighboring patches of different texture (e.g. Figure 1(d)). We compute pairwise texture similarities by comparing windowed texton histograms computed using the technique described previously (§2.1.2). A number of methods are available for comparing histograms. We use 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 an empirical comparison of the χ^2 test versus other texture similarity measures, see (Puzicha et al., 1997).

W_{ij}^{TX} is then defined as follows:

$$W_{ij}^{TX} \; = \; \exp(-\chi^2(h_i, h_j)/\sigma_{TX}) \tag{4}$$

If histograms h_i and h_j are very different, χ^2 is large, and the weight W_{ij}^{TX} is small.

4.3. General Images

Finally we consider the general case of images that contain boundaries of both kinds. This presents us with the problem of *cue integration.* The obvious approach to cue integration is to define the weight between pixels i and j as the product of the contribution from each cue: $W_{ij} = W_{ij}^{IC} \times W_{ij}^{TX}$. The idea is that if either of the cues suggests that i and j should be separated, the composite weight, W_{ij}, should be small. We must be careful, however, to avoid the problems listed in the Introduction (§1) by suitably gating the cues. The spirit of the gating method is to make each cue "harmless" in locations where the other cue should be operating.

4.3.1. *Estimating texturedness*

As illustrated in Figure 2, the fact that a pixel survives the non-maximum suppression step does not necessarily mean that that pixel lies on a region boundary. Consider a pixel inside a patch of uniform texture: its oriented energy is large but it does not lie on the boundary of a region. Conversely, consider a pixel lying between two uniform patches of just slightly different brightness: it does lie on a region boundary but its oriented energy is small. In order to estimate the "probability" that a pixel lies on a boundary, it is necessary to take more surrounding information into account. Clearly the true value of this probability is only determined after the final correct segmentation, which is what we seek to find. At this stage our goal is to formulate a local estimate of the texturedness of the region surrounding a pixel.

Since this is a local estimate, it will be noisy but its objective will be to bootstrap the global segmentation procedure.

Our method of computing this value is based on a simple comparison of texton distributions on either side of a pixel relative to its dominant orientation. Consider a generic pixel q at an oriented energy maxima. Let the dominant orientation be θ. Consider a circle of radius $\alpha(q)$ (the selected scale) centered on q. We first divide this circle in two along the diameter with orientation θ. Note that the contour passing through q is tangent to the diameter, which is its best straight line approximation. The pixels in the disk can be partitioned into three sets D_0, D_-, D_+ which are the pixels in the strip along the diameter, the pixels to the left of D_0, and the pixels to the right of D_0, respectively. To compute our measure of texturedness, we consider two half window comparisons with D_0 assigned to each side. Assume without loss of generality that D_0 is first assigned to the "left" half. Denote the K-bin histograms of $D_0 \cup D_-$ by h_L and D_+ by h_R respectively. Now consider the χ^2 statistic between the two histograms:

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

We repeat the test with the histograms of D_- and $D_0 \cup D_+$ and retain the maximum of the two resulting values, which we denote χ^2_{LR}. We can convert this to a probability-like value using a sigmoid as follows:

$$p_{texture} = 1 - \frac{1}{1 + \exp[-(\chi^2_{LR} - \tau)/\beta]} \tag{5}$$

This value, which ranges between 0 and 1, is small if the distributions on the two sides are very different and large otherwise. Note that in the case of untextured regions, such as a brightness step edge, the textons lying along and parallel to the boundary make the statistics of the two sides different. This is illustrated in Figure 9. Roughly, $p_{texture} \approx 1$ for oriented energy maxima in texture and $p_{texture} \approx 0$ for contours. $p_{texture}$ is defined to be 0 at pixels which are not oriented energy maxima.

4.3.2. *Gating the Contour Cue*

The contour cue is gated by means of suppressing contour energy according to the value of $p_{texture}$. The gated value, p_B, is defined as

$$p_B = (1 - p_{texture})p_{con} \tag{6}$$

In principle, this value can be computed and dealt with independently at each filter scale. For our purposes, we found it sufficient simply to

Figure 9. Illustration of half windows used for the estimation of the texturedness. The texturedness of a label is based on a χ^2 test on the textons in the two sides of a box as shown above for two sample pixels. The size and orientation of the box is determined by the selected scale and dominant orientation for the pixel at center. Within the rocky area, the texton statistics are very similar, leading to a low χ^2 value. On the edge of the wing, the χ^2 value is relatively high due to the dissimilarity of the textons that fire on either side of a step edge. Since in the case of the contour the contour itself can lie along the diameter of the circle, we consider two half-window partitions: one where the thin strip around the diameter is assigned to the left side, and one where it is assigned to the other. We consider both possibilities and retain the maximum of the two resulting χ^2 values.

keep the maximum value of p_B with respect to σ. The gated contour energy is illustrated in Figure 10, right. The corresponding weight is then given by

$$W_{ij}^{IC} = 1 - \max_{x \in M_{ij}} p_B(x)$$

4.3.3. *Gating the Texture Cue*

The texture cue is gated by computing a texton histogram at each pixel which takes into account the texturedness measure $p_{texture}$. Let h_i be the K-bin texton histogram computed using Equation 2. We define a $(K+1)$-bin histogram $\hat{h}_i$ by introducing a 0^{th} bin. The intuition is that the 0^{th} bin will keep a count of the number of pixels which do not correspond to texture. These pixels arise in two forms: (1) pixels which are not oriented energy maxima; (2) pixels which *are* oriented energy maxima, but correspond to boundaries between two regions, thus should not take part in texture processing to avoid the problems discussed in (§1) More precisely, $\hat{h}_i$ is defined as follows:

$$\hat{h}_i(k) = \sum_{j \in \mathcal{N}(i)} p_{texture}(j) \cdot I[T(j) = k] \qquad \forall k = 1 \ldots K$$

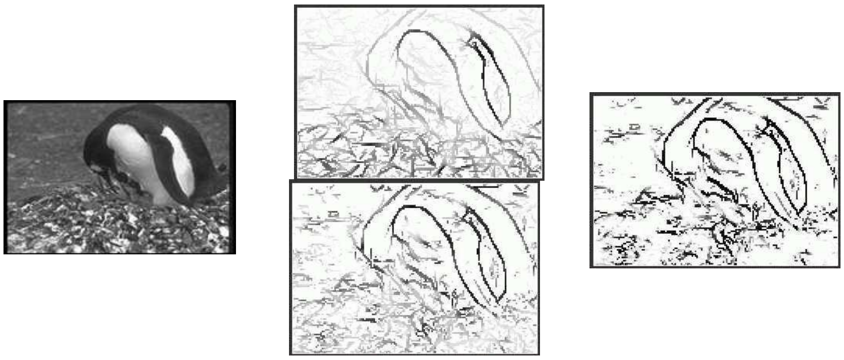

Figure 10. Gating the contour cue. Left: original image. Top: oriented energy after nonmaximal suppression, OE^*. Bottom: $1 - p_{texture}$. Right: p_B, the product of $1 - p_{texture}$ and $p_{con} = 1 - \exp(-OE^*/\sigma_{IC})$. Note that this can be thought of as a "soft" edge detector which has been modified to no longer fire on texture regions.

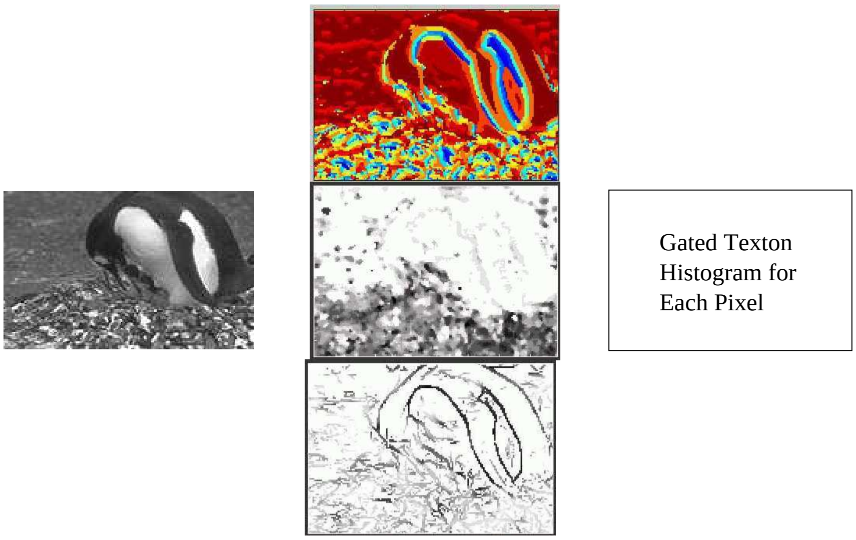

Figure 11. Gating the texture cue. Left: original image. Top: Textons label, shown in pseudocolor. Middle: local scale estimate $\alpha(i)$. Bottom: $1-p_{texture}$. Darker grayscale indicates larger values. Right: Local texton histograms at scale $\alpha(i)$ are gated using $p_{texture}$ as explained in 4.3.3.

$$\hat{h}_i(0) \;=\; N_B + \sum_{j \in \mathcal{N}(i)} (1 - p_{texture}(j))$$

where $\mathcal{N}(i)$ denotes all the oriented energy maxima lying inside the window $\mathcal{W}(i)$ and N_B is the number of pixels which are not oriented energy maxima.

4.3.4. *Combining the Weights*
After each cue has been gated by the above procedure, we are free to perform simple multiplication of the weights. More specifically, we first obtain W^{IC} using Equation 6. Then we obtain W^{TX} using Equation 4 with the gated versions of the histograms. Then we simply define the combined weight as

$$W_{ij} = W_{ij}^{IC} \times W_{ij}^{TX}$$

4.3.5. *Implementation Details*
The weight matrix is defined between any pair of pixels i and j. Naively, one might connect every pair of pixels in the image. However, this is not necessary. Pixels very far away from the image have very small likelihood of belonging to the same region. Moreover, dense connectivity means that we need to solve for the eigenvectors of a matrix of size $N_{pix} \times N_{pix}$, where N_{pix} is close to a million for a typical image. In practice, a sparse and short-ranged connection pattern does a very good job. In our experiments, all the images are of size 128×192. Each pixel is connected to pixels within a radius of 30. Furthermore, a sparse sampling is implemented such that the number of connections is approximately constant at each radius. The number of non-zero connections is 1000 in our experiments. For images of different sizes, the connection radius can be scaled appropriately.

The parameters for the various formulae are given here:

1. The image brightness lies in the range $[0, 1]$.

2. $\sigma_{IC} = 0.02$ (Equation 1).

3. The number of textons computed using K-means: $K = 36$.

4. The textons are computed following a contrast normalization step, motivated by Weber's law. Let $|F(x)|$ be the L_2 norm of the filter responses at pixel x. We normalize the filter responses by the following equation:

$$F(x) \leftarrow F(x) \times \frac{\log(1 + \frac{|F(x)|}{0.03})}{|F(x)|}$$

5. $\sigma_{TX} = 0.025$ (Equation 4).

6. $\tau = 0.3$ and $\beta = 0.04$ (Equation 5)

Note that these parameters are the same for all the results shown in (§6).

5. Computing the Segmentation

With a properly defined weight matrix, the normalized cut formulation discussed in (§3) can be used to compute the segmentation. However, the weight matrix defined in the previous section is computed using only local information, and is thus not perfect. The ideal weight should be computed in such a way that region boundaries are respected. More precisely, (1) texton histograms should be collected from pixels in a window residing exclusively in one and only one region. If instead, an isotropic window is used, pixels near a texture boundary will have a histogram computed from textons in both regions, thus "polluting" the histogram. (2) Intervening contours should only be considered at region boundaries. Any responses to the filters inside a region are either caused by texture or are simply mistakes. However, these two criteria mean that we need a segmentation of the image, which is exactly the reason why we compute the weights in the first place! This chicken-and-egg problem suggests an iterative framework for computing the segmentation. First, use the local estimation of the weights to compute a segmentation. This segmentation is done so that no region boundaries are missed, i.e. it is an over-segmentation. Next, use this intial segmentation to update the weights. Since the initial segmentation does not miss any region boundaries, we can coarsen the graph by merging all the nodes inside a region into one *super-node*. We can then use these *super-nodes* to define a much simpler segmentation problem. Of course, we can continue this iteration several times. However, we elect to stop after 1 iteration.

The procedure consists of the following 4 steps:

1. Compute an initial segmentation from the locally estimated weight matrix.

2. Update the weights using the initial segmentation.

3. Coarsen the graph with the updated weights to reduce the segmentation to a much simpler problem.

4. Compute a final segmentation using the coarsened graph.

5.1. Computing the Initial Segmentation

Computing a segmentation of the image amounts to computing the eigenvectors of the generalized eigensystem: $(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{v} = \lambda \boldsymbol{D}\boldsymbol{v}$ (Equation 3). The eigenvectors can be thought of as a transformation of the image into a new feature vector space. In other words, each pixel in the original image is now represented by a vector with the components coming from the corresponding pixel across the different eigenvectors. Finding a partition of the image is done by finding the clusters in this eigenvector representation. This is a much simpler problem because the eigenvectors have essentially put regions of coherent descriptors according to our cue of texture and contour into very tight clusters. Simple techniques such as K-means can do a very good job in finding these clusters. The following procedure is taken:

1. Compute the eigenvectors corresponding to the second smallest to the twelfth smallest eigenvalues of the generalized eigensystem $((\boldsymbol{D}-\boldsymbol{W})\boldsymbol{v}=\lambda\boldsymbol{D}\boldsymbol{v})$.[6] Call these 11 eigenvectors $v_i, i=2,\ldots,12$. The corresponding eigenvalues are $\lambda_i, i=2,\ldots,12$.

2. Weight [7] the eigenvectors according to the eigenvalues: $\hat{v}_i=\frac{1}{\sqrt{\lambda_i}}v_i$, $i=2,\ldots,12$. The eigenvalues indicate the "goodness" of the corresponding eigenvectors. Now each pixel is transformed to an 11 dimensional vector represented by the weighted eigenvectors.

3. Perform vector quantization on the 11 eigenvectors using K-means. Start with $K^*=30$ centers. Let the corresponding RMS error for the quantization be e^*. Greedily delete one center at a time such that the increase in quantization error is the smallest. Continue this process until we arrive at K centers when the error e is just greater than $1.1\times e^*$.

This partitioning strategy provides us with an initial segmentation of the image. This is usually an over-segmentation. The main goal here is simply to provide an initial guess for us to modify the weights. Call this initial segmentation of the image $\boldsymbol{S}_0$. Let the number of segments be N_0. A typical number for N_0 is $10-100$.

5.2. Updating Weights

The initial segmentation $\boldsymbol{S}_0$ found in the previous step can provide a good approximation to modify the weight as we have discussed earlier. With $\boldsymbol{S}_0$, we modify the weight matrix as follows:

- To compute the texton histograms for a pixel in R_k, textons are collected only from the intersection of R_k and the isotropic window of size determined by the scale, α.

- p_B is set to zero for pixels that are not in the region boundaries of $\boldsymbol{S}_0$.

The modified weight matrix is an improvement over the original local estimation of weights.

5.3. Coarsening the Graph

By hypothesis, since $\boldsymbol{S}_0$ is an over-segmentation of the image, there are no boundaries missed. We do not need to recompute a segmentation for the original problem of N pixels. We can coarsen the graph, where each node of the new graph is a segment in $\boldsymbol{S}_0$. The weight between two nodes in this new graph is computed as follows:

$$\hat{W}_{kl} = \sum_{i \in R_k} \sum_{j \in R_l} W_{ij} \tag{7}$$

where R_k and R_l indicate segments in $\boldsymbol{S}_0$ (k and $l \in \{1, \ldots, N_0\}$); $\hat{W}$ is the weight matrix of the coarsened graph and W is the weight matrix of the original graph. This coarsening strategy is a very standard technique in the application of graph partitioning (Metis, 1999). Now, we have reduced the original segmentation problem with an $N \times N$ weight matrix to a much simpler and faster segmentation problem of $N_0 \times N_0$ without losing in performance.

5.4. Computing the Final Segmentation

After coarsening the graph, we have turned the segmentation problem into a very simple graph partitioning problem of very small size. We compute the final segmentation using the following procedure:

1. Compute the second smallest eigenvector for the generalized eigensystem using $\hat{W}$.

2. Threshold the eigenvector to produce a bi-partitioning of the image. 30 different values uniformly spaced within the range of the eigenvector are tried as the threshold. The one producing a partition which minimizes the normalized cut value is chosen. The corresponding partition is the best way to segment the image into two regions.

3. Recursively repeat steps 1 and 2 for each of the partitions until the normalized cut value is larger than 0.1.

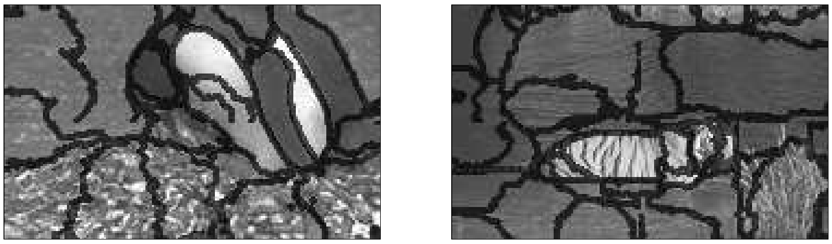

Figure 12. p_B is allowed to be non-zero only at pixels marked.

5.5. Segmentation in Windows

The above procedure performs very well in images with a small number of groups. However, in complicated images, smaller regions can be missed. This problem is intrinsic for global segmentation techniques, where the goal is find a big-picture interpretation of the image. This problem can be dealt with very easily by performing the segmentation in windows.

Consider the case of breaking up the image into quadrants. Define Q^i to be the set of pixels in the i^{th} quadrant. $Q^i \cap Q^j = \emptyset$ and $\cup_{i=1}^4 Q^i =$ Image. Extend each quadrant by including all the pixels which are less than a distance r from any pixels in Q^i, with r being the maximum texture scale, $\alpha(i)$, over the whole image. Call these enlarged windows $\hat{Q}^i$. Note that these windows now overlap each other.

Corresponding to each $\hat{Q}^i$, a weight matrix $\hat{W}^i$ is defined by pulling out from the original weight matrix W the edges whose end-points are nodes in $\hat{Q}^i$. For each $\hat{W}^i$, an initial segmentation $\hat{\boldsymbol{S}}_0^i$ is obtained, according to the procedure in (§5.1). The weights are updated as in (§5.2). The extension of each quadrant makes sure that the arbitrary boundaries created by the windowing do not affect this procedure:

Texton histogram upgrade For each pixel in Q^i, the largest possible histogram window (a $(2\alpha + 1)^2$ box) is entirely contained in $\hat{Q}^i$ by virtue of the extension. This means the texton histograms are computed from all the relevant pixels.

Contour upgrade The boundaries in Q_i are a proper subset of the boundaries in $\hat{Q}_i$. So, we can set the values of p_B at a pixel in Q_i to be zero if it lies on a region boundary in $\hat{Q}_i$. This enables the correct computation of W_{ij}^{IC}. Two example contour update maps are shown in Figure 12

Initial segmentations can be computed for each $\hat{Q}^i$ to give $\hat{\boldsymbol{S}}_0^i$. They are restricted to Q^i to produce $\boldsymbol{S}_0^i$. These segmentations are merged to form an initial segmentation $\boldsymbol{S}_0 = \cup_{i=1}^4 \boldsymbol{S}_0^i$. At this stage, fake boundaries from the windowing effect can occur. Two examples are shown in Figure 13. The graph is then coarsened and the final segmentation is computed as in (§5.3) and (§5.4).

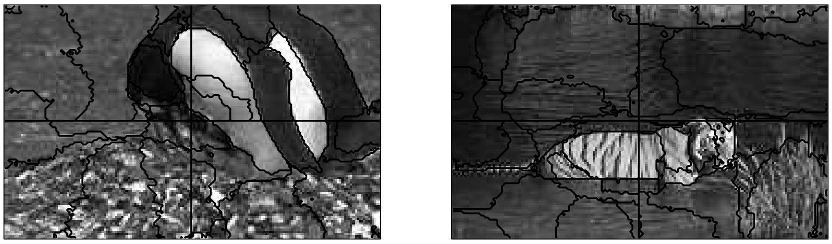

Figure 13. Initial segmentation of the image used for coarsening the graph and computing final segmentation.

6. Results

We have run our algorithm on a variety of natural images. Figures 14 to 17 show typical segmentation results. In all the cases, the regions are cleanly separated from each other using combined texture and contour cues. Notice that for all these images, a single set of parameters are used. Color is not used in any of these examples and can readily be included to further improve the performance of our algorithm [8]. Figure 14 shows results for animal images. Results for images containing people are shown in Figure 15 while natural and man-made scenes appear in Figure 16. Segmentation results for paintings are shown in Figure 17. A set of more than 1000 images from the commercially available Corel Stock Photos database have been segmented using our algorithm[9].

Acknowledgements

The authors would like to thank the Berkeley vision group, especially Chad Carson, Alyosha Efros, David Forsyth and Yair Weiss for useful discussions. 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.

Notes

[1] For more discussions and variations of the K-means algorithm, the reader is referred to (Duda and Hart, 1973; Gersho and Gray, 1992).

[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 3% of the image dimension in our experiments. This is tied to the intermediate scale of the filters in the filter set.

[4] This is set to 10% of the image dimension in our experiments.

[5] Finding the true optimal partition is an NP-complete problem.

[6] The eigenvector corresponding to the smallest eigenvalue is constant, thus useless.

[7] Since normalized cut can be interpreted as a spring-mass system (Shi and Malik, 2000), this normalization comes from the *equipartition theorem* in classical statistical mechanics which states that if a system is in equilibrium, then it has equal energy in each mode (Belongie and Malik, 1998).

[8] When color information is available, the similarity W_{ij} becomes a product of 3 terms: $W_{ij} = W_{ij}^{IC} \times W_{ij}^{TX} \times W_{ij}^{COLOR}$. Color similarity, W_{ij}^{COLOR}, is computed using χ^2 differences over color histograms, similar to texture measured using texture histograms. Moreover, color can clustered into "colorons", analogous to textons.

[9] These results are available at the following web page:
http://www.cs.berkeley.edu/projects/vision/Grouping/overview.html

References

Belongie, S., C. Carson, H. Greenspan, and J. Malik: 1998, 'Color- and texture-based image segmentation using EM and its application to content-based image retrieval'. In: *Proc. 6th Int. Conf. Computer Vision*. Bombay, India, pp. 675–82.

Belongie, S. and J. Malik: 1998, 'Finding Boundaries in Natural Images: A New Method Using Point Descriptors and Area Completion'. In: *Proc. 5th Euro. Conf. Computer Vision*. Freiburg, Germany, pp. 751–66.

Binford, T.: 1981, 'Inferring surfaces from images'. *Artificial Intelligence* **17**(1-3), 205–44.

Canny, J.: 1986, 'A computational approach to edge detection'. *IEEE Trans. Pat. Anal. Mach. Intell.* **8**(6), 679–98.

DeValois, R. and K. DeValois: 1988, *Spatial Vision*. Oxford University Press.

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

Elder, J. and S. Zucker: 1996, 'Computing contour closures'. In: *Proc. Euro. Conf. Computer Vision*, Vol. I. Cambridge, England, pp. 399–412.

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

Geman, S. and D. Geman: 1984, 'Stochastic relaxation, Gibbs distribution, and the Bayesian retoration of images'. *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 721–41.

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

Heeger, D. J. and J. R. Bergen: 1995, 'Pyramid-Based Texture Analysis/Synthesis'. In: *Proceedings of SIGGRAPH '95*. pp. 229–238.

Jacobs, D.: 1996, 'Robust and efficient detection of salient convex groups'. *IEEE Trans. Pattern Anal. Mach. Intell.* **18**(1), 23–37.

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

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

Knutsson, H. and G. Granlund: 1983, 'Texture analysis using two-dimensional quadrature filters'. In: *Workshop on Computer Architecture for Pattern Analysis and Image Database Management*. pp. 206–13.

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

Koenderink, J. and A. van Doorn: 1988, 'Operational significance of receptive field assemblies'. *Biological Cybernetics* **58**, 163–171.

Leung, T. and J. Malik: 1998, 'Contour continuity in region-based image segmentation'. In: H. Burkhardt and B. Neumann (eds.): *Proc. Euro. Conf. Computer Vision*, Vol. 1. Freiburg, Germany, pp. 544–59.

Leung, T. and J. Malik: 1999, 'Recognizing surfaces using three-dimensional textons'. In: *Proc. Int. Conf. Computer Vision*. Corfu, Greece, pp. 1010–17.

Malik, J. and P. Perona: 1990, 'Preattentive texture discrimination with early vision mechanisms'. *J. Optical Society of America* **7**(2), 923–32.

Malik, J. and P. Perona: 1992, 'Finding boundaries in images'. In: H. Wechsler (ed.): *Neural Networks for Perception*, Vol. 1. Academic Press, pp. 315–44.

Metis: 1999. http://www.cs.umn.edu/metis.

Montanari, U.: 1971, 'On the optimal detection of curves in noisy pictures'. *Comm. Ass. Comput.* **14**, 335–45.

Morrone, M. and D. Burr: 1988, 'Feature detection in human vision: a phase dependent energy model'. *Proc. R. Soc. Lond. B* **235**, 221–45.

Morrone, M. and R. Owens: 1987, 'Feature detection from local energy'. *Pattern Recognition Letters* **6**, 303–13.

Mumford, D. and J. Shah: 1989, 'Optimal approximations by piecewise smooth functions, and associated variational problems'. *Comm. Pure Math.* pp. 577–684.

Parent, P. and S. Zucker: 1989, 'Trace inference, curvature consistency, and curve detection'. *IEEE Trans. Pattern Anal. Mach. Intell.* **11**(8), 823–39.

Perona, P. and J. Malik: 1990, 'Detecting and localizing edges composed of steps, peaks and roofs'. In: *Proc. 3rd Int. Conf. Computer Vision*. Osaka, Japan, pp. 52–7.

Puzicha, J., T. Hofmann, and J. Buhmann: 1997, 'Non-parametric similarity measures for unsupervised texture segmentation and image retrieval'. In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition*. San Juan, Puerto Rico, pp. 267–72.

Sha'ashua, A. and S. Ullman: 1988, 'Structural Saliency: the detection of globally salient structures using a locally connected network'. In: *Proc. 2nd Int. Conf. Computer Vision*. Tampa, Fl., U.S.A., pp. 321–7.

Shi, J. and J. Malik: 1997, 'Normalized cuts and image segmentation'. In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition*. San Juan, Puerto Rico, pp. 731–7.

Shi, J. and J. Malik: 2000, 'Normalized cuts and image segmentation'. *IEEE Trans. Pattern Anal. Mach. Intell.* to appear.

Wertheimer, M.: 1938, 'Laws of organization in perceptual forms (partial translation)'. In: W. Ellis (ed.): *A sourcebook of Gestalt Psychology.* Harcourt Brace and Company, pp. 71–88.

Williams, L. and D. Jacobs: 1995, 'Stochastic completion fields: a neural model of illusory contour shape and salience'. In: *Proc. 5th Int. Conf. Computer Vision.* Cambridge, MA, pp. 408–15.

Young, R. A.: 1985, 'The Gaussian derivative theory of spatial vision: Analysis of cortical cell receptive field line-weighting profiles'. Technical Report GMR-4920, General Motors Research.

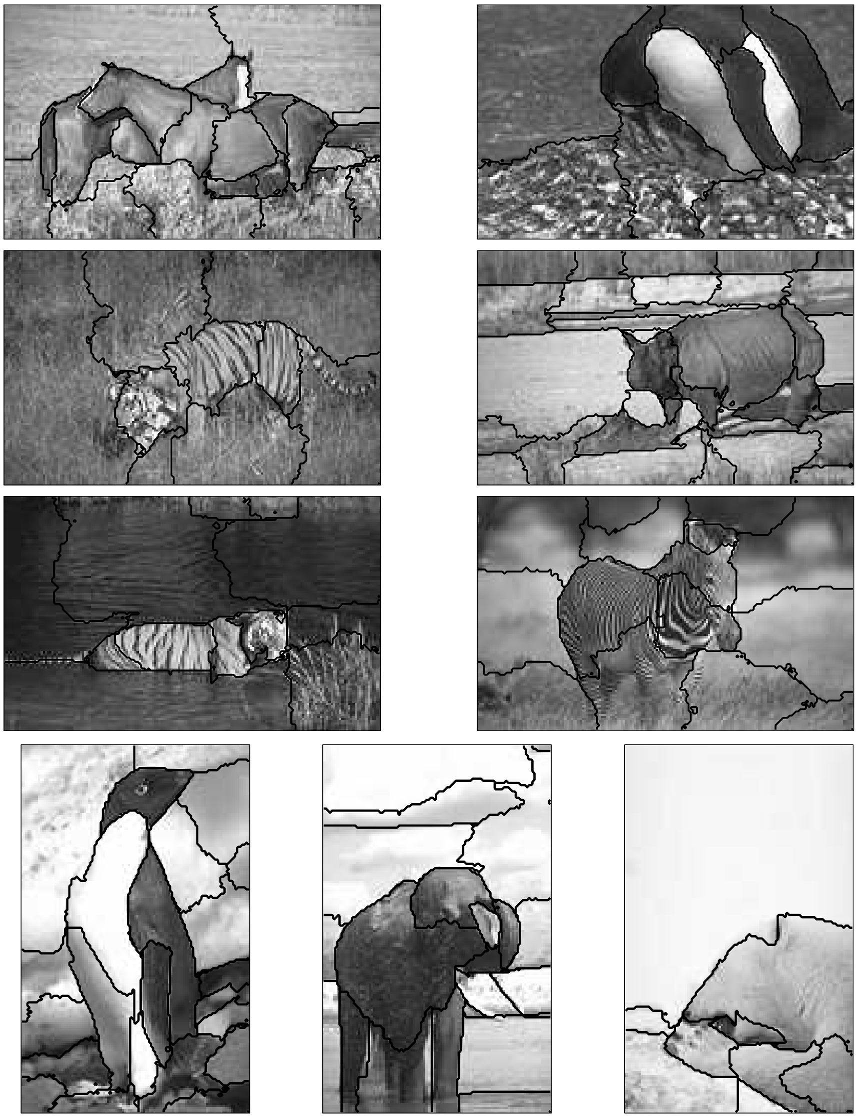

Figure 14. Segmentation of images with animals.

Figure 15. Segmentation of images with people.

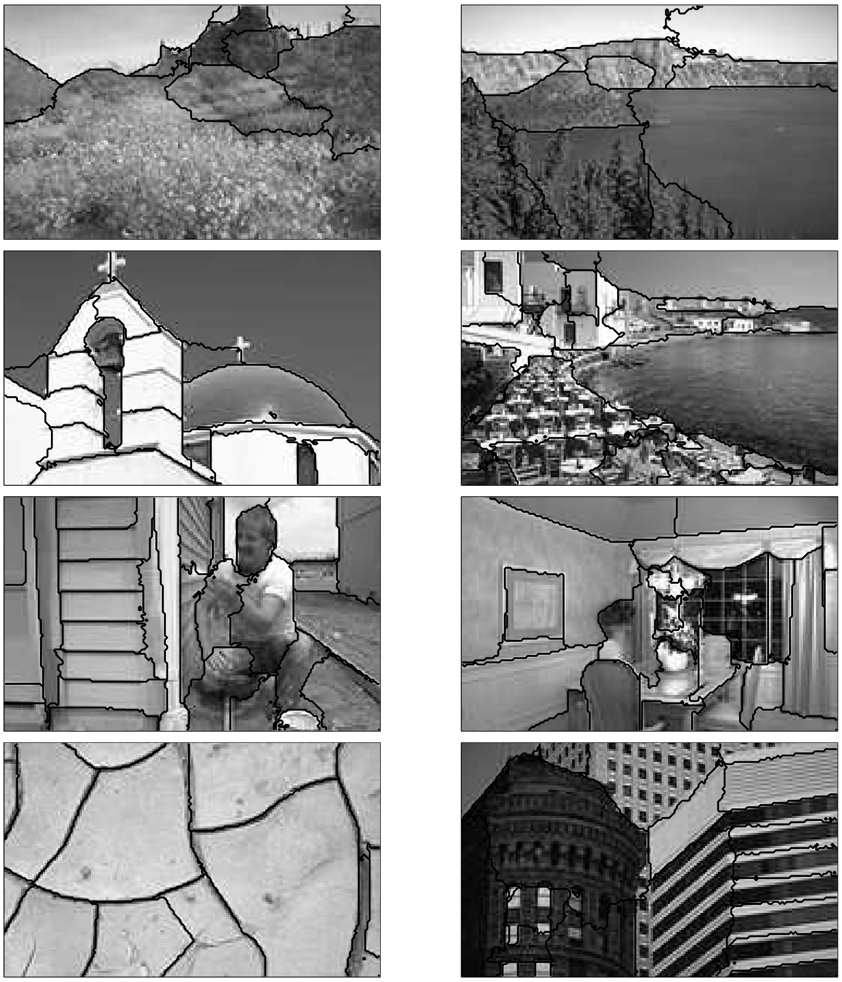

Figure 16. Segmentation of images of natural and man-made scenes.

Figure 17. Segmentation of paintings.