Image Segmentation

Introduction. The goal of image segmentation is to cluster pixels into salient image regions, i.e., regions corresponding to individual surfaces, objects, or natural parts of objects.

A segmentation could be used for object recognition, occlusion boundary estimation within motion or stereo systems, image compression, image editing, or image database look-up.

We consider **bottom-up image segmentation**. That is, we ignore (top-down) contributions from object recognition in the segmentation process.

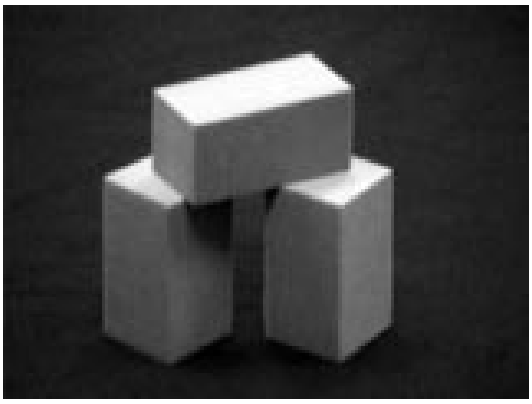
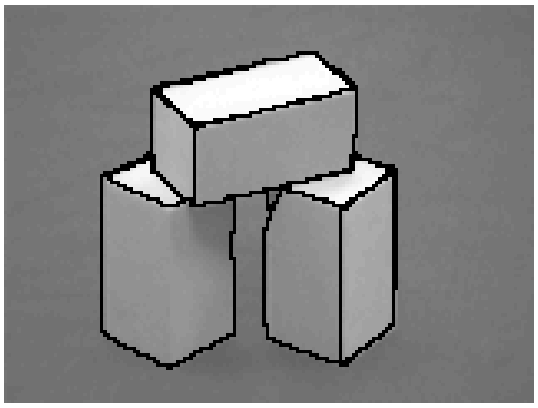
For input we primarily consider image brightness here, although similar techniques can be used with colour, motion, and/or stereo disparity information.

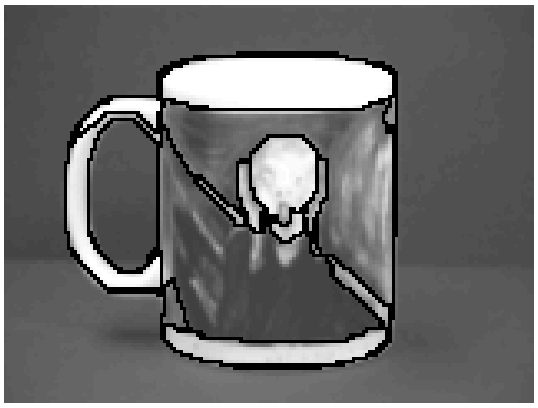
Reading on Segmentation: See Chapter 14 of the text.

Example Segmentations: Simple Scenes

Segmentations of simple gray-level images can provide useful information about the surfaces in the scene.

Original Image | Segmentation (by SMC)



Note, unlike edge images, these boundaries delimit disjoint image regions (i.e. they are **closed**).

Siren Song of Segmentation

Why would a good segmentation be useful? Imagine...

Parent to baby: “Look, there is a baby horse with its mommy!”

Baby:

Reasoning

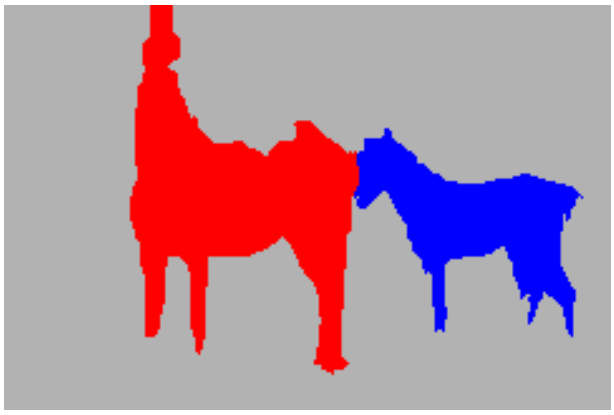
```
1.  Follow pointing gesture.
2.  Acquire image.
3.  horse isa animal
4.  animal ⇝ quadraped
5.  baby horse ⇝ small horse

Visual Task:  Seek correlates
of two similar quadrapeds in image,
one smaller than the other.
```

Image

Bottom-Up Segmentation

Parse of Two Quadrapeds



Baby: “Gaaa.” (Translation: “Eureka, I can see!”)

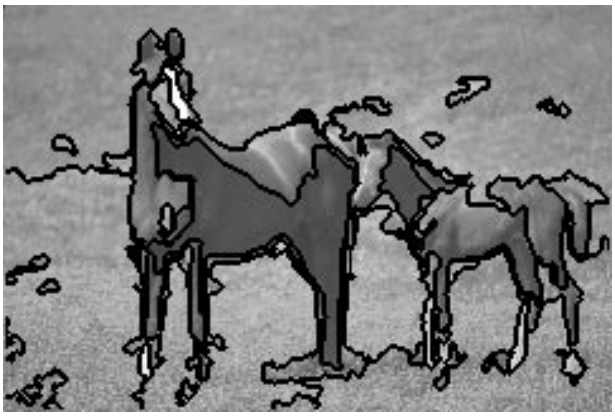
Key Questions

1. How well can we expect to segment images without recognizing objects (i.e. bottom-up segmentation)?
2. What determines a segment? How can we pose the problem mathematically?
3. How do we solve the specified problem(s)?
4. How can we evaluate the results?

Example Segmentations: Horses Image

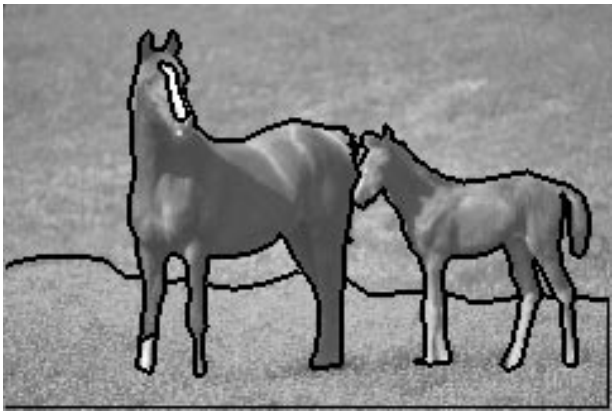
Original Image

LV

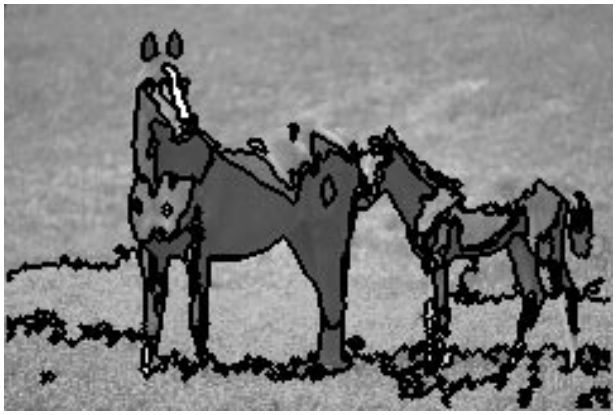


SMC

H



ED



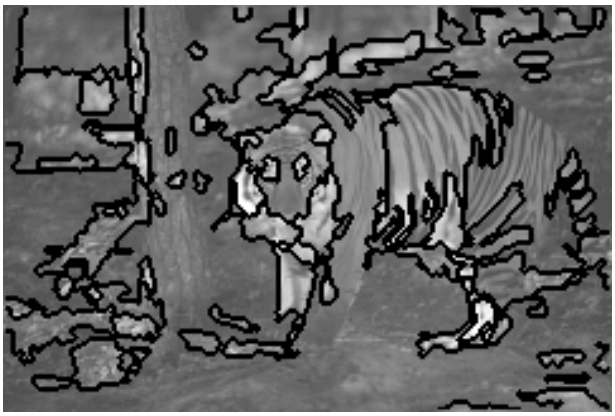
NC

Which is the best segmentation? Why?

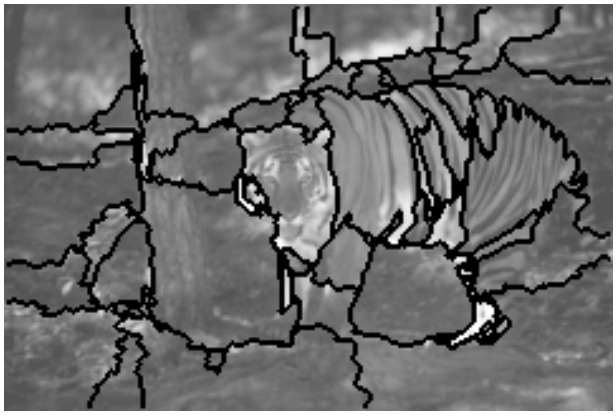
Example Segmentations: Tiger Image

Original Image

LV

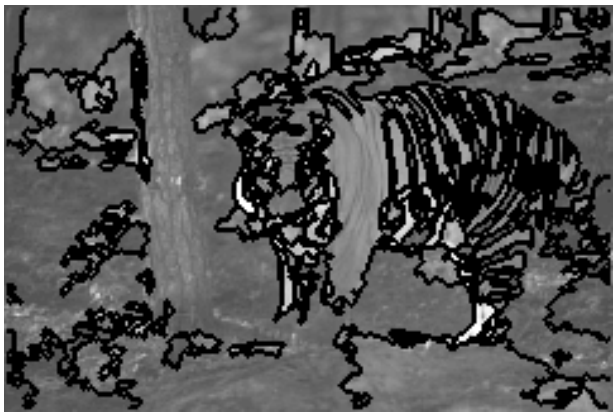


SMC

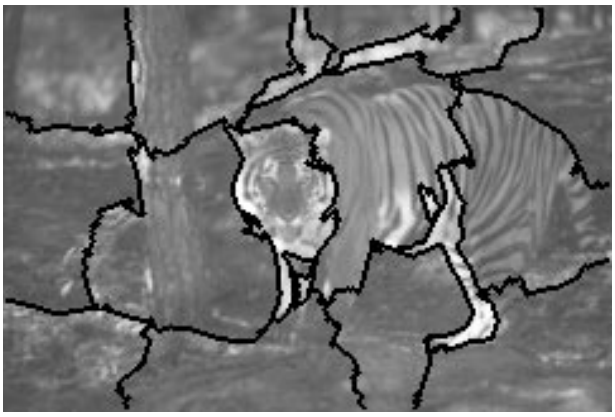


H

ED



NC



Group these into K categories based on quality. ($K = 2$?)

Observations on Example Segmentations

The previous segmentations were done by the local variation (LV) algorithm [7], spectral min-cut (SMC) [6], human (H) [11, 9], edge-augmented mean-shift (ED) [4, 3], and normalized cut (NC) [13, 5].

- The quality of the segmentation depends on the image. Smoothly shaded surfaces with clear gray-level steps between different surfaces are ideal for the above algorithms.
- Humans probably use object recognition in conjunction with segmentation, although the machine algorithms exhibited above do not.
- For relatively simple images it is plausible that machine segmentations, such as those shown on p.2, are useful for several visual tasks, including object recognition.
- For more complex images (pp. 5, 6), the machine segmentations provide a less reliable indicator for surface boundaries, and their utility for subsequent processing becomes questionable.
- While many segmentation algorithms work well with simple examples, they will all break down given examples with enough clutter and camouflage. The assessment of segmentation algorithms therefore needs to be done on standardized datasets.

Current Goals

- Provide a brief introduction to the current image segmentation literature, including:
  - Feature space clustering approaches.
  - Graph-based approaches.
- Discuss the inherent assumptions different approaches make about what constitutes a good segment.
- Emphasize general mathematical tools that are promising.
- Discuss metrics for evaluating the results.

Clustering in Feature Space

Given an image $I(\vec{x})$, consider feature vectors $\vec{F}(\vec{x})$ of the form

$$\vec{F}(\vec{x}) = \begin{pmatrix} \vec{x} \\ I(\vec{x}) \\ \vec{L}(\vec{x}) \end{pmatrix}.$$

Here $\vec{L}(\vec{x})$ is a vector of local image features, perhaps bandpass filter responses. For colour images, $\vec{F}(\vec{x})$ would also include information about the colour at pixel $\vec{x}$.

In order to segment the image we might seek a clustering of the feature vectors $\vec{F}(\vec{x})$ observed in that image. A compact region of the image having a distinct gray-level or colour will correspond to a region in the feature space with a relatively high density of sampled feature vectors.

Mixture of Gaussians Model

A natural approach is then to model the observed feature vector distribution using a mixture of Gaussians (MoG) model M,

$$p(\vec{F}|M) \;=\; \sum_{k=1}^{K} \pi_k g(\vec{F} \mid \vec{m}_k, \Sigma_k).$$

Here $\pi_k \geq 0$ are the mixing coefficients, with $\sum_{k=1}^{K} \pi_k = 1$, and $\vec{m}_k$, Σ_k are the means and covariances of the component Gaussians.

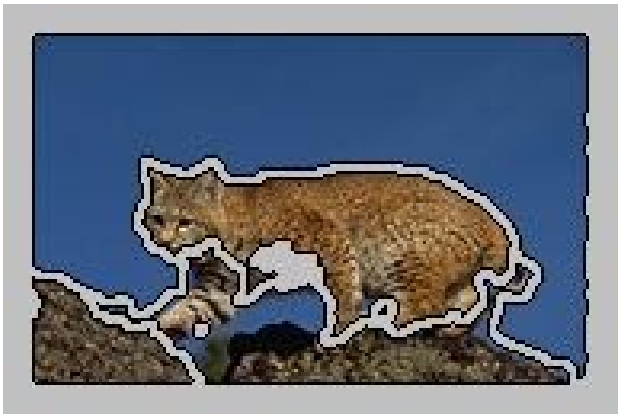
For a given K, the parameters $\{(\pi_k, \vec{m}_k, \Sigma_k)\}_{k=1}^{K}$ of the MoG model can be fit to the data $\{\vec{F}(\vec{x})\}_{\vec{x}\in X}$ using maximum-likelihood (here X denotes the set of all pixels).

Penalized likelihood (aka minimum description length (MDL)) can be used to select the number of components, K.

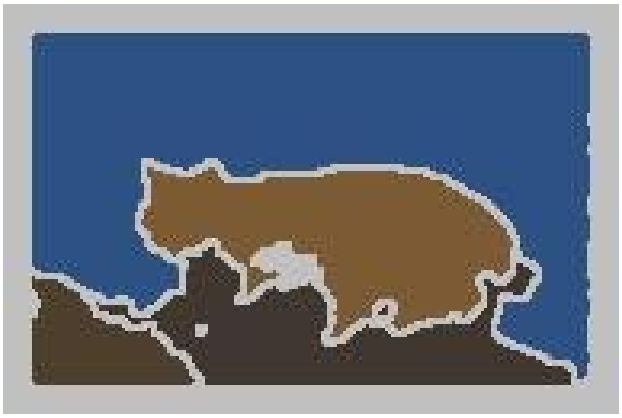
Maximum Ownership Labelling

The segment label $c(\vec{x}) = k$ for a pixel $\vec{x}$ is the k which maximizes the ownership of $\vec{F}(\vec{x})$ in the MoG model M. That is,

$$c(\vec{x}) = \arg\max_k \left[\frac{\pi_k g(\vec{F}(\vec{x}) \mid \vec{m}_k, \Sigma_k)}{p(\vec{F}(\vec{x}) \mid M)} \right].$$



Here K = 3 (above right). The max-ownership image was post-processed using connected components and small regions were discarded (gray). The average colour of the remaining large components is shown (right). The width of the segment boundaries is due to the use of a spatial texture feature.



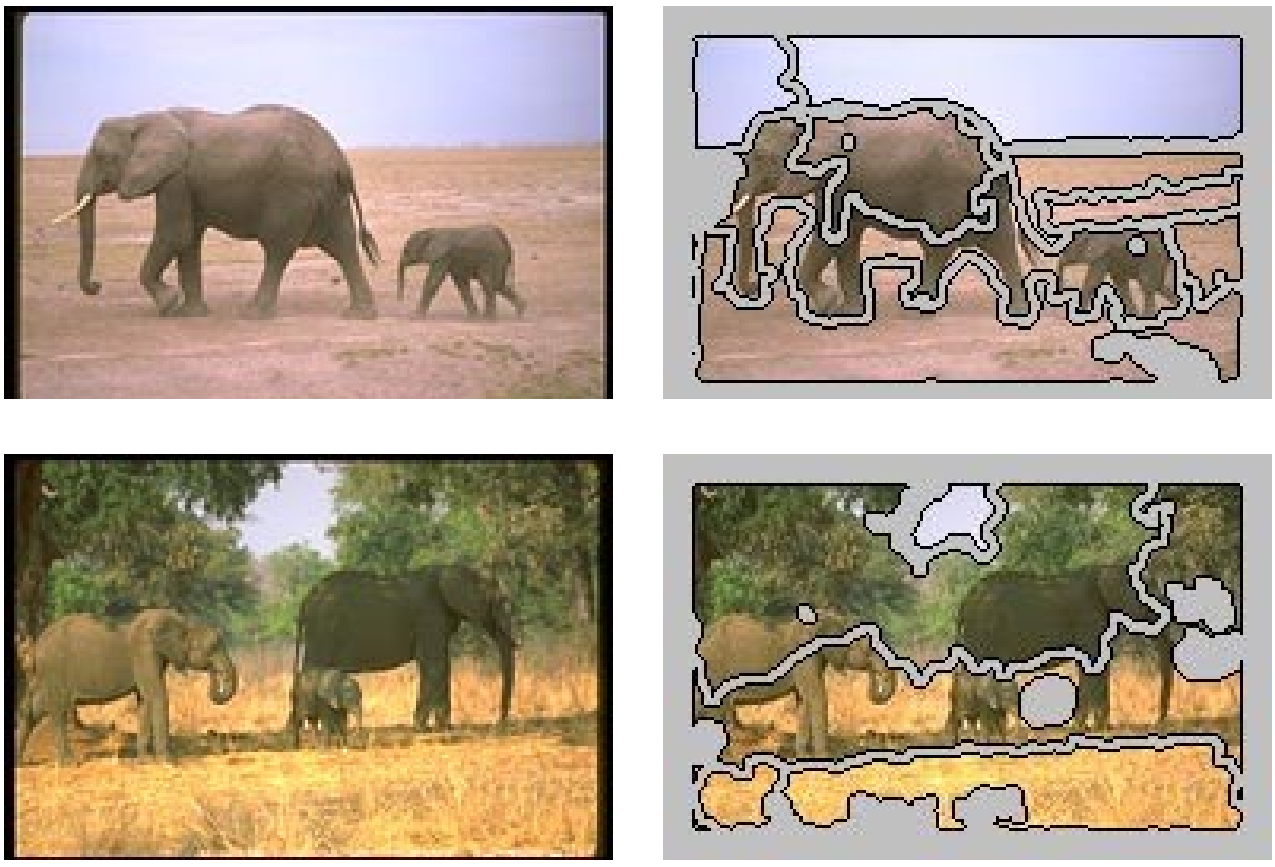
From Blobworld [2].

Variations: The MoG model can be replaced by K-means (see text), or restricted to use low-dimensional parameterizations for Σ_k (eg. block diagonal).

Assumptions Come Home to Roost

The quality of the resulting segmentation depends on the degree to which the given image matches the (implicit) assumptions we began with, namely:

1. Different segments form compact, well-separated clusters in $\vec{F}$.
2. Gaussian components in M correspond to salient regions.



From Blobworld [2].

Nevertheless, this feature space clustering can be useful for extracting rough summaries of image content suitable for querying image databases [2].

Mean-Shift Segmentation

The mean-shift segmentation algorithm [4] also considers the probability density of feature vectors $\vec{F}(\vec{x})$ obtained from a given image. However, a **non-parametric** model of the density is used instead of an MoG. In particular, a kernel-density estimate is used, with

$$p_K(\vec{f}) \equiv \frac{1}{|X|} \sum_{\vec{x} \in X} K(\vec{f} - \vec{F}(\vec{x})),$$

where X is the set of all pixels in the image, $|X|$ is the number of pixels, and $K(\vec{e})$ is a kernel.

Common choices for $K(\vec{e})$ have the form

$$K(\vec{e}) = k(\vec{e}^T \Sigma^{-1} \vec{e}), \tag{1}$$

where $k(s)$ is a convex decreasing function of the squared deviation $s \equiv \vec{e}^T \Sigma^{-1} \vec{e} \geq 0$. For example,

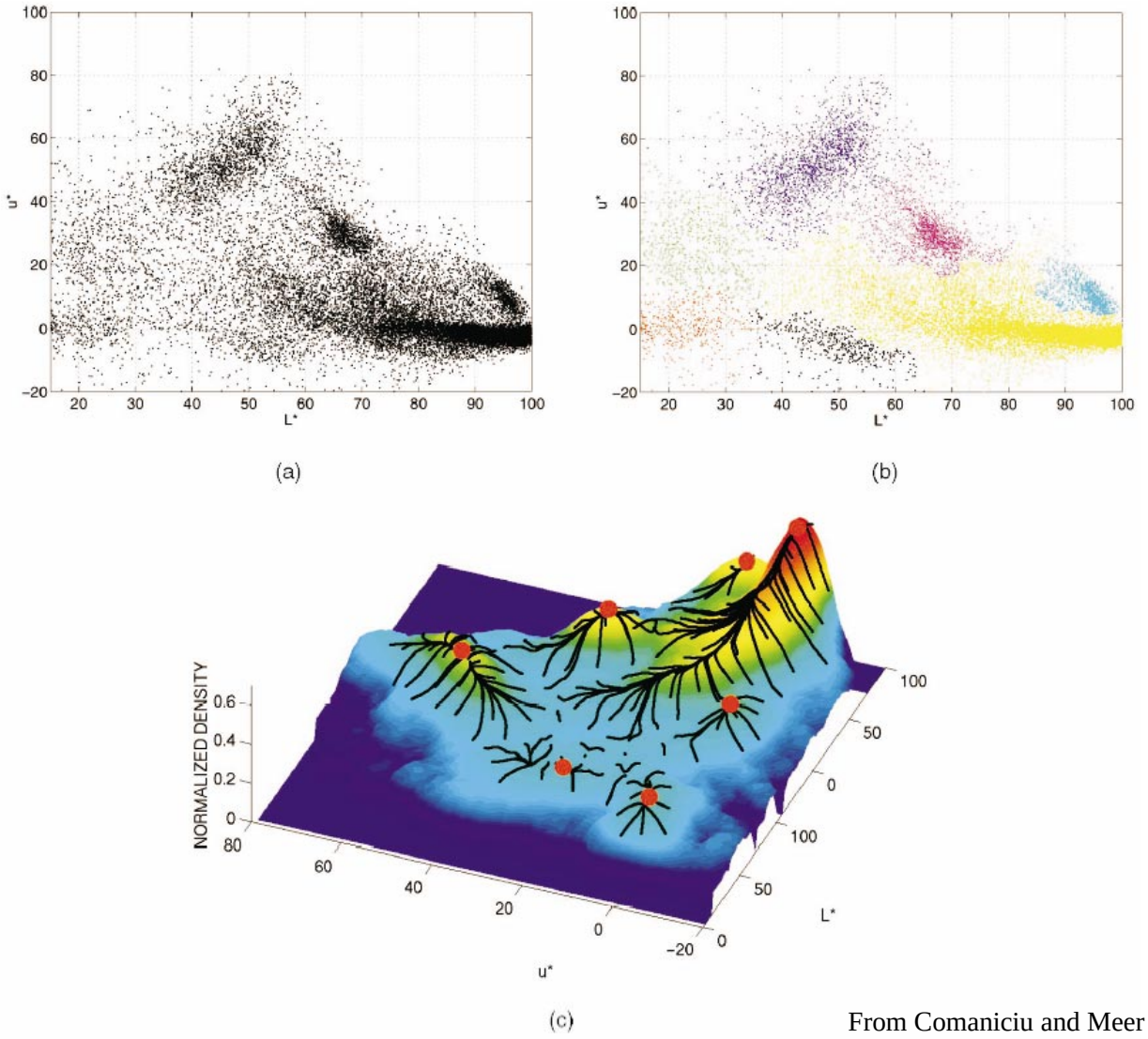
$$k(s) = ce^{-s/2}, \qquad \text{for a Gaussian kernel,} \tag{2}$$

$$k(s) = c\lfloor 1 - s \rfloor_+, \qquad \text{for an Epanechnikov kernel.} \tag{3}$$

Here $c = c(\Sigma)$ is a normalizing constant which ensures $K(\vec{e})$ integrates to one, and $\lfloor z \rfloor_+$ denotes positive rectification, i.e. $\lfloor z \rfloor_+ \equiv \max(z, 0)$.

We show an example next.

Example Feature Density



From Comaniciu and Meer [4].

The 2d feature points in (a) are interpolated using the Epanechnikov kernel (c). The covariance parameter Σ of the kernel $K(\vec{e})$ determines the smoothness of the density estimate $p_K(\vec{f})$. The trade-off is between sampling artifacts (kernel too narrow) versus loss of resolution in $p_K(\vec{f})$ (kernel too broad).

Mean-Shift Iterations

We will use the modes (i.e. peaks) of $p_K(\vec{f})$ to be segmentation labels, replacing the use of the component labels in the previous MoG model. That is, we wish to locally solve

$$\vec{f}_* = \arg\max_{\vec{f}} p_K(\vec{f}).$$

This is similar to robust M-estimation, although here we are maximizing the objective function $p_K(\vec{f})$, not minimizing it. A similar derivation to the one for M-estimation shows $\vec{f}_*$ must satisfy

$$\vec{f}_* = \left[\sum_{\vec{x} \in X} w(\vec{F}(\vec{x}) - \vec{f}_*) \vec{F}(\vec{x}) \right] / \left[\sum_{\vec{x} \in X} w(\vec{F}(\vec{x}) - \vec{f}_*) \right]$$

where $w(\vec{e}) = -k'(\vec{e}^{\,T} \Sigma^{-1} \vec{e})$ and $k'(s) = \frac{dk}{ds}(s)$. In words, $\vec{f}_*$ must be the weighted mean of $\vec{F}(\vec{x})$ using the weights $w(\vec{F}(\vec{x}) - \vec{f}_*)$ centered on $\vec{f}_*$.

The analogue of the iterative reweighting idea used in M-estimation is to solve for $\vec{f}_*$ here by iterating the **mean-shift** equation

$$\vec{f}_{j+1} = \frac{\sum_{\vec{x} \in X} w(\vec{F}(\vec{x}) - \vec{f}_j) \vec{F}(\vec{x})}{\sum_{\vec{x} \in X} w(\vec{F}(\vec{x}) - \vec{f}_j)}. \tag{4}$$

Note $\vec{f}_{j+1}$ is just the weighted mean of the feature points $\vec{F}(\vec{x})$, with the weights $w(\vec{F}(\vec{x}) - \vec{f}_j)$ centered on the previous guess $\vec{f}_j$.

Watersheds of Mean-Shift

The label for an arbitrary pixel $\vec{x}_0$ denotes the mode that the mean shift iterations (4) converge to, when started at the feature $\vec{f}_0 = \vec{F}(\vec{x}_0)$. That is, the segments produced by mean-shift are defined to be the **domains of convergence** (aka watersheds) of the mean-shift iterations.

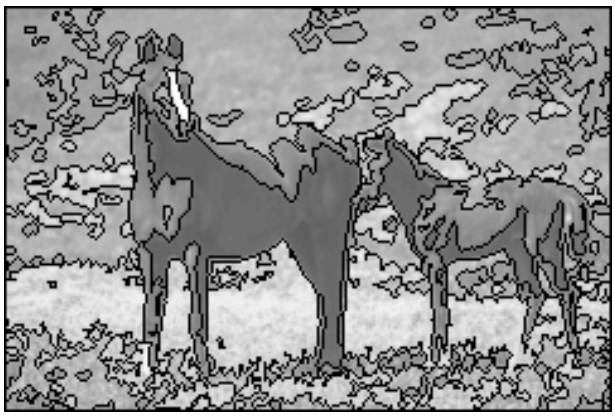
In the figure on p.14 the trajectories of mean-shift are shown in (c). The labelling resulting from the watersheds is shown by the colours in (b).

Properties:

1. **Convergence:** The mean-shift iterations converge to a stationary point of $p_K(\vec{f})$ (see [4]).
2. **Anti-edge Detection:** The mean-shift iterations are repelled from local maxima of the norm of the gradient (wrt $\vec{x}$) of $\vec{F}^{\,T}(\vec{x})\Sigma^{-1}\vec{F}(\vec{x})$. This occurs, for example, at strong edges in the image $I(\vec{x})$.
3. **Fragmentation of Constant Gradient Regions:** The density $p_K(\vec{f})$ is constant (up to discretization artifacts) in regions where the gradient of $\vec{F}^{\,T}(\vec{x})\Sigma^{-1}\vec{F}(\vec{x})$ is constant. For example, when $\vec{\nabla} I(\vec{x})$ is constant, every point of $p_K(\vec{f})$ is stationary and the mean-shift iterations stall (see figure p.14, part c). Postprocessing is required to keep only salient local maxima (see [4]).

Example Mean-Shift Segmentations

Segmentations from the basic mean-shift algorithm:



The scale of the mean-shift kernel (controlled by Σ) roughly controls the size and shape of the extracted regions. There is a trade-off between maintaining the salient boundaries but suffering over-segmentation, versus missing some of the important boundaries and under-segmenting the image. The segmentations above illustrate a typical compromise.

An enhanced system (EDISON [3]) combines the mean-shift algorithm with image edge information. An edge-saliency measure is used to modify the weight function used in the mean-shift equation (4). This eases the above trade-off, allowing weak boundaries to be kept in the segmentation without incurring as much over-segmentation. Image segmentation results using the EDISON system are shown on pp. 5-6 (labelled ED). The use of salient-edge information significantly improves the results.

Similarity Graph Based Methods

Graph-based methods provide an alternative to feature space clustering.

A weighted undirected graph $G = (V, E)$ is formed, with the set of vertices V corresponding to the pixels $\vec{x}$ in the image. Edges E in the graph are taken between any two pixels $\vec{x}_i$ and $\vec{x}_j$ within a small distance of each other.

The edge weight $w(\vec{x}_i, \vec{x}_j) \geq 0$ reflects the dissimilarity (alternatively, the similarity) between the two image neighbourhoods centered on pixels $\vec{x}_i$ and $\vec{x}_j$. A common form of the weight function is to use $w(\vec{x}_i, \vec{x}_j) = 1 - a(\vec{x}_i, \vec{x}_j)$ where the affinity $a(\vec{x}_i, \vec{x}_j)$ is given by

$$a(\vec{x}_i, \vec{x}_j) \equiv e^{-\frac{1}{2}(\vec{F}(\vec{x}_i) - \vec{F}(\vec{x}_j))^T \Sigma^{-1} (\vec{F}(\vec{x}_i) - \vec{F}(\vec{x}_j))}.$$

Here $\vec{F}(\vec{x})$ is a feature vector associated with pixel $\vec{x}$, for example:

1. $\vec{F}(\vec{x}) = I(\vec{x})$, so the affinity is determined only by the grey-level difference between neighbouring pixels,
2. $\vec{F}(\vec{x}) = \vec{I}(\vec{x})$, the RGB values for a colour image, or some mapping of the RGB values to a more uniform colour space (eg. L*u*v*).
3. $\vec{F}(\vec{x})$ includes texture primitives, such as local filter responses, along with the brightness and/or colour at pixel $\vec{x}$.

Connected Components (Not Robust)

A simple approach is to delete all edges between dissimilar pixels (i.e., with weights $w(\vec{x}_i, \vec{x}_j) > \tau$), and then seek connected components (CCs) in the remaining graph.

Note that a single edge with $w(\vec{x}_i, \vec{x}_j) \leq \tau$ would be sufficient to cause two desired regions to be merged. Therefore CCs are not robust to stray links (aka "leaks") between regions. The consequence is that there is often no suitable value of τ which gives a useful segmentation.

Kruskal's Algorithm. In passing, it is useful to point out that an efficient way to do CC clustering, with a variable τ, is to first build a minimal spanning tree (MST) of the graph. Kruskal's algorithm can be used, which is a greedy approach guaranteed to give an optimal MST. Beginning with the completely disconnected graph, edges are added one at a time in increasing order of their weights, so long as adding an edge does not introduce cycles in the current sub-graph.

The CCs of the decimated graph (with edges having $w(\vec{x}_i, \vec{x}_j) > \tau$ removed) are then efficiently computed by deleting these same edges from the MST. The trees in the resulting forest provide the desired CCs.

A modified version of Kruskal's algorithm is considered next.

Local Variation Method

Felzenszwalb and Huttenlocher [7] introduce a simple but effective modification of Kruskal's algorithm. As in Kruskal's algorithm, it begins with the completely disconnected graph, edges are added one at a time in increasing order of their weights, maintaining a forest of MSTs for the current components.

During processing, each MST C_i is associated with a threshold

$$T(C_i) = w(C_i) + k/|C_i| \tag{5}$$

where $w(C_i)$ is the maximum weight in the spanning tree C_i (i.e. the **local variation** of C_i). Also $k > 0$ is a constant, and $|C_i|$ is the number of pixels in C_i.

Suppose the edge $(\vec{x}_k, \vec{x}_l)$ is to be processed next, and its two endpoints are in two separate MSTs C_i and C_j. Then these MSTs are merged by adding the edge $(\vec{x}_k, \vec{x}_l)$ only if

$$w(\vec{x}_k, \vec{x}_l) \leq \min(T(C_i), T(C_j)). \tag{6}$$

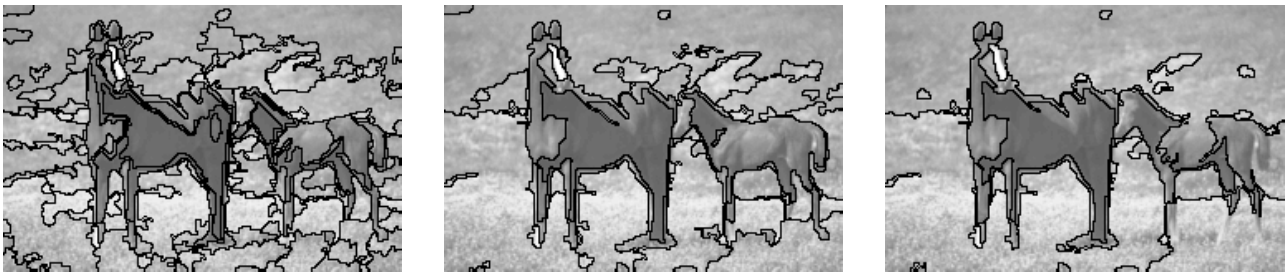
Note that, as the size of C_i increases, (5) and (6) dictate an increasingly tight upper bound $T(C_i)$ (compared to the largest weight $w(C_i)$ in C_i) for the acceptable affinity of an edge merging C_i with another region.

Examples of Local Variation Segmentation

Sorting the edges according to weight causes the algorithm to grow relatively homogeneous regions first.

The parameter k in (5) roughly controls the size of the regions in the resulting segmentation. Larger k provides a looser constraint (6), and allows more merging.

k = 50 k = 150 k = 250



The merging is sensitive to the local variation within the regions being merged. Due to the increasingly tight bound (5), a large homogeneous region C_i is only merged using edges with weights at most fractionally larger than $w(C_i)$, the largest affinity in the MST C_i. However, this bound is much looser for small regions C_i, encouraging their growth.

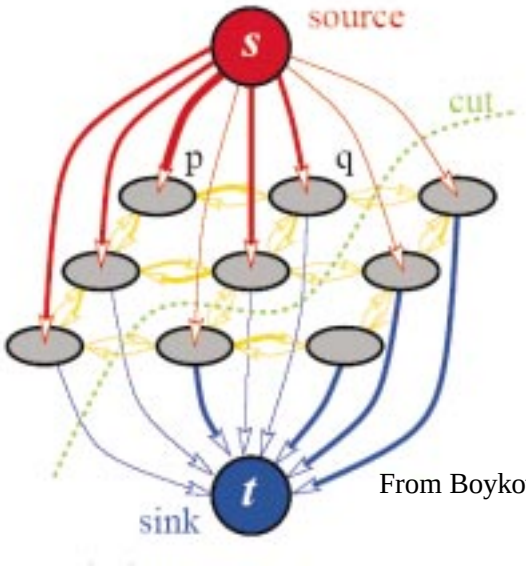
The approach has a tendency to produce narrow regions along 'true' segment boundaries (see examples above).

The approach is very efficient computationally, requiring $O(e \log(e))$ operations where e is the number of edges.

Source-Sink Minimum Cut

An alternative graph-based approach makes use of efficient solutions of the max-flow/min-cut problem between source and sink nodes in directed graphs.

S-T Min Cut



From Boykov and Kolmogorov [1].

S-T Min-Cut Problem. An S-T graph is a weighted directed graph with two identified nodes, the source s and the sink t. We seek a minimum cut separating s and t. That is, we seek a partioning of the graph into two sets of nodes F and G, with $G = V - F$, $s \in F$, and $t \in G$, such that the linkage

$$L(F, G) = \sum_{\vec{x}_i \in F, \vec{x}_j \in G} a(\vec{x}_i, \vec{x}_j). \tag{7}$$

is minimized.

Source-Sink Minimum Cut (Cont.)

Efficient algorithms have recently been developed that solve the S-T min-cut problem (see [1]).

The S-T min-cut problem is computationally much simpler than the more general **graph partitioning problem**, which is to find a (non-empty) partition F and $G = V - F$ which minimizes $L(F, G)$ (i.e., without any further constraints, such as $s \in F$ and $t \in G$.)

To take advantage of an efficient solution to the S-T min-cut problem, we need to generate an S-T graph. Given two disjoint sets of pixels S and T, we form a weighted directed graph as follows. For each edge $(\vec{x}_i, \vec{x}_j)$ in the previous undirected graph, the two directed edges $\langle \vec{x}_i, \vec{x}_j \rangle$ and the reverse edge $\langle \vec{x}_j, \vec{x}_i \rangle$ are included. Both of these edges are weighted by the affinity $a(\vec{x}_i, \vec{x}_j)$. In addition, two additional nodes s and t are created, namely the source and sink nodes, respectively. Finally, infinitely weighted directed links $\langle s, \vec{x}_i \rangle$ and $\langle \vec{x}_j, t \rangle$ are included for each $\vec{x}_i \in S$ and $\vec{x}_j \in T$.

The resulting S-T min-cut then provides a globally minimum cost cut between the sets of pixels S and T.

Seed Regions for S-T Min Cut

The sets S and T (connected to the source and sink, respectively) should satisfy:

1. Each S and T generated must be sufficiently large (otherwise the minimum cut is either S and $V - S$, or T and $V - T$),
2. Each S and T should be contained within different 'true' segments (due to the infinite weights, neither S or T will be partitioned),
3. Enough pairs S and T should be generated to identify most of the salient segments in the image.

One suitable generation process is discussed in Estrada et al [6]. It is based on spectral properties of a matrix representing the affinities. Sample results are given in segmentations labelled SMC on pp.2, 5, and 6 above.

The process is much more computationally intensive than the previous ones. Several hundred min-cut problems are typically solved for different S, T, and these alone require several minutes on relatively small images (eg. 40K pixels).

The intriguing property of this approach is that the S-T min-cut algorithm computes **globally** optimal cuts (subject to the proposals for S and T).

Normalized Cut

Finally we outline the normalized cut approach of Shi and Malik [13]. Here we seek a partition F and $G = V - F$ of the affinity weighted, undirected graph (without source and sink nodes). In order to avoid partitions where one of F or G is a tiny region, Shi and Malik propose the normalized cut criterion, namely that F and G should minimize

$$N(F,G) \equiv L(F,G)\left(\frac{1}{L(F,V)} + \frac{1}{L(G,V)}\right), \tag{8}$$

where L is the linkage defined in (7).

Unfortunately, the resulting graph partitioning problem,

$$F = \arg\min_{F \subset V} N(F, V - F), \tag{9}$$

is computationally intractable [13]. Therefore we must seek algorithms which provide approximate solutions of (9).

Note any segmentation technique can be used for generating proposals for suitable regions F, for which $N(F, V - F)$ could be evaluated. Indeed, the SMC approach above can be viewed as using S and T to provide lower bounds on the terms $L(F, V)$ and $L(G, V)$ (namely $L(S, V)$ and $L(T, V)$, respectively), and then using the S-T min cut to globally minimize $L(F, G)$ subject to $S \subset F$ and $T \subset G$.

Discrete Rayleigh Quotient

Shi and Malik [13] prove that (9) is equivalent to the discrete optimization problem

$$\arg\min_{\vec{y}} \frac{\vec{y}^{\,T}(D-A)\vec{y}}{\vec{y}^{\,T}D\vec{y}} \text{ subject to } y_i \in \{1,-b\} \text{ and } \vec{d}^{\,T}\vec{y}=0. \quad (10)$$

Here A is the $N \times N$ symmetric matrix of affinities $a(\vec{x}_i, \vec{x}_j)$, which is arranged (say) according to the raster ordering of the pixels $\vec{x}_i$, $i = 1, \ldots, N$. Also, $\vec{d} = A\vec{1}$, where $\vec{1}$ is the N-vector of all ones, $b > 0$, and D is the diagonal matrix with $D_{i,i} = d_i$.

Given a solution $\vec{y}$ of (10), the corresponding solution F of (9) is then obtained by setting $F = \{\vec{x}_i \mid y_i > 0\}$. And, vice versa, given F we set $y_i = 1$ for each $\vec{x}_i \in F$, and set the other elements of $\vec{y}$ to $-b$, where $b > 0$ is chosen such that $\vec{d}^{\,T}\vec{y} = 0$.

Spectral Approximation for Normalized Cut

Equation (10) is a discrete version of a standard eigenvector formulation, namely the Rayleigh quotient. This suggests using the tractable approximation obtained by temporarily allowing $\vec{y}$ to be a real-valued vector (instead of 2-valued). By setting $\vec{y} = D^{-1/2}\vec{u}$ we find

$$\arg\min_{\vec{u}} \frac{\vec{u}^{\,T}(I - B)\vec{u}}{\vec{u}^{\,T}\vec{u}} \text{ subject to } \vec{d}^{\,T/2}\vec{u} = 0. \tag{11}$$

with $B = D^{-1/2}AD^{-1/2}$, a symmetric matrix. This is a standard eigenvalue problem in linear algebra!

Equation (11) can be simplified further by noting that $\vec{u} = \vec{d}^{\,1/2}$ is an eigenvector of B with eigenvalue 1. It therefore must be an eigenvector of $I - B$ with eigenvalue 0. Moreover, it can be shown that all the eigenvalues of $I - B$ are in the interval $[0, 2]$. Thus (11) is the standard Rayleigh quotient form for the eigenvector u of $I - B$ with the **second smallest** eigenvalue.

Spectral Approximation (Cont.)

In the original Ncut algorithm [13], an approximation to a discrete solution of (10) is then obtained by thresholding $D^{1/2}\vec{u}$ at each of a set of values. For each threshold τ, the Ncut objective function (8) is evaluated, and the best value of τ is selected. This produces two regions F and $V - F$. Regions are then recursively partitioned using the same approach, until a user-specified number of segments is obtained. See pp.5-6 for examples.

The step of thresholding the second largest eigenvector to provide a partitioning proposal is a key limitation of the approach. In practice, the approximation only appears to be consistently reliable when there is exactly one obvious way to partition the data. More recently, Yu and Shi [14] attempt to alleviate this problem by extracting K segments from the subspace spanned by the K eigenvectors of $I - B$ having the smallest eigenvalues.

For further information, see the reading list in the CVPR 2004, graph-based segmentation tutorial [12].

Natural Image Boundaries

As we have seen, segmentation involves finding salient regions and their boundaries.

A **boundary** in an image is a contour that represents the change from one object or surface to another. This is distinct from image **edges**, which mark rapid changes in image brightness (say), but may or may not correspond to salient boundaries.

The previous segmentation techniques could be (and some have been) usefully coupled with a bottom-up boundary detector. For example:

1. The EDISON mean-shift segmentation algorithm [3] illustrated one example of this in reweighting the mean-shift iterations based on salient edge information.
2. The affinities used in the Ncut algorithm [13] use intervening edge information to reduce the affinities between pairs of pixels [8].

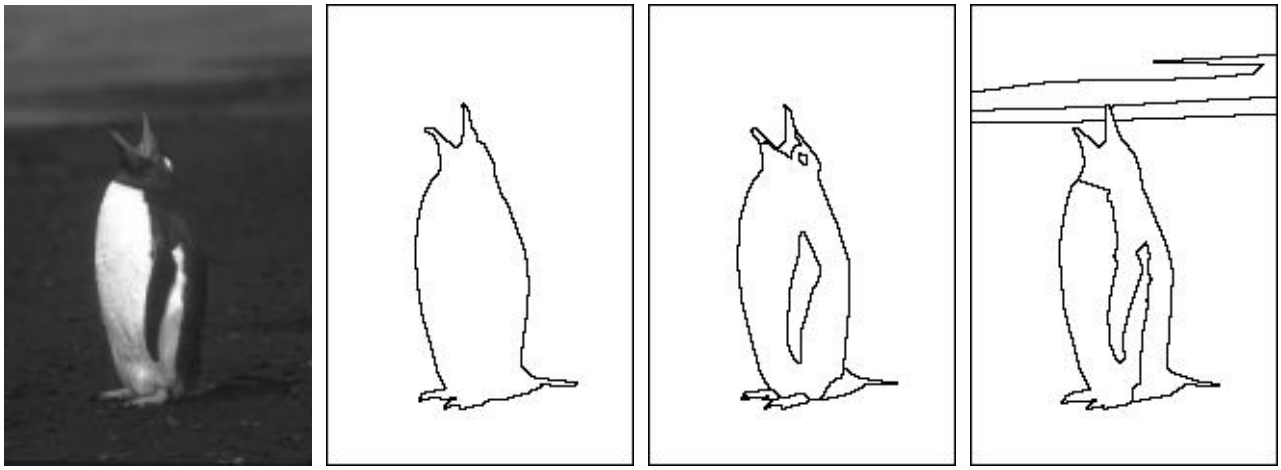
The development of local, bottom-up, boundary detectors is an important problem, complimentary to the segmentation approaches discussed here. See Martin et al [10] for recent work.

Berkeley Segmentation Database

The Berkeley Segmentation Dataset [9, 11] provides image segmentations done by humans. As stated on the dataset's webpage:

> The goal of this work is to provide an empirical and scientific basis for research on image segmentation and boundary detection.

The public portion of this dataset consists of the segmentations of 300 images by roughly 5 humans each, done separately for greylevel and colour versions of the images. Three examples from one image are shown below:



Note these segmentations appear to be consistent, except different subjects have decided to resolve particular regions into more or less detail. This variability should be taken into account in a quantitative comparison of two segmentations.

Quantitative Comparison of Segmentations

Martin et al [11] suggest a quantitative measure for comparing two segmentations, which must have roughly the same number of segments, but for which the relative resolution of each region may be different.

This measure is as follows. Let $R(S, \vec{x}_i)$ be the set of pixels having the same label as pixel $\vec{x}_i$ in the segmentation S. Given two segmentations, S_1 and S_2, define

$$E(S_1, S_2, \vec{x}_i) = \frac{|R(S_1, \vec{x}_i) - R(S_2, \vec{x}_i)|}{|R(S_1, \vec{x}_i)|}.$$

Then $E(S_1, S_2, \vec{x}_i) = 0$ when the region $R(S_2, \vec{x}_i)$ contains $R(S_1, \vec{x}_i)$, but not vice versa. Finally, the local consistency error is defined to be

$$LCE(S_1, S_2) = \frac{1}{N} \sum_{\vec{x} \in X} \min(E(S_1, S_2, \vec{x}_i), E(S_2, S_1, \vec{x}_i)).$$

The minimization within the sum allows local refinements of S_1 with respect to S_2, or S_2 wrt S_1, to both receive a LCE score of 0.

Only the Ncut algorithm has been quantitatively compared to human segmentations [11] in the literature so far. However, this type of quantitative comparison (perhaps with other measures, such as boundary detection statistics [10]) is expected to play an important role in assessing the performance of segmentation algorithms in the future.

References

[1] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *submitted IEEE Trans. Pattern Anal. and Machine Intell.*, 2004.

[2] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Trans. Pattern Anal. and Machine Intell.*, 24(8):1026–1038, 2002.

[3] C. M. Christoudias, B. Georgescu, and P. Meer. Synergism in low level vision. In *16th International Conference on Pattern Recognition., Quebec City, Canada*, volume IV, pages 150–155, 2002.

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. and Machine Intell.*, 24:603–619, 2002.

[5] T. Cour, S. Yu, and J. Shi. Normalized cuts matlab code. Computer and Information Science, Penn State University. Code available at http://www.cis.upenn.edu/˜jshi/software/.

[6] F.J. Estrada, A.D. Jepson, and C. Chennubhotla. Spectral embedding and min-cut for image segmentation. In *British Machine Vision Conference*, 2004.

[7] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. of Comp. Vis.*, 59(2):167–181, 2004.

[8] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *Int. J. of Computer Vision*, 43(1):7–27, 2001.

[9] D. Martin and C. Fowlkes. *The Berkeley Segmentation Dataset and Benchmark.* http://www.cs.berkeley.edu/projects/vision/grouping/segbench/.

[10] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. and Machine Intell.*, 26(5):530–549, 2004.

[11] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[12] J. Shi, C. Fowlkes, D. Martin, and E. Sharon. Graph based image segmentation tutorial. CVPR 2004. http://www.cis.upenn.edu/˜jshi/GraphTutorial/.

[13] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. and Machine Intell.*, 22(8):888–905, 2000.

[14] S. Yu and J. Shi. Multiclass spectral clustering. In *Proc. Int'l Conf. Computer Vision*, 2003.