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.

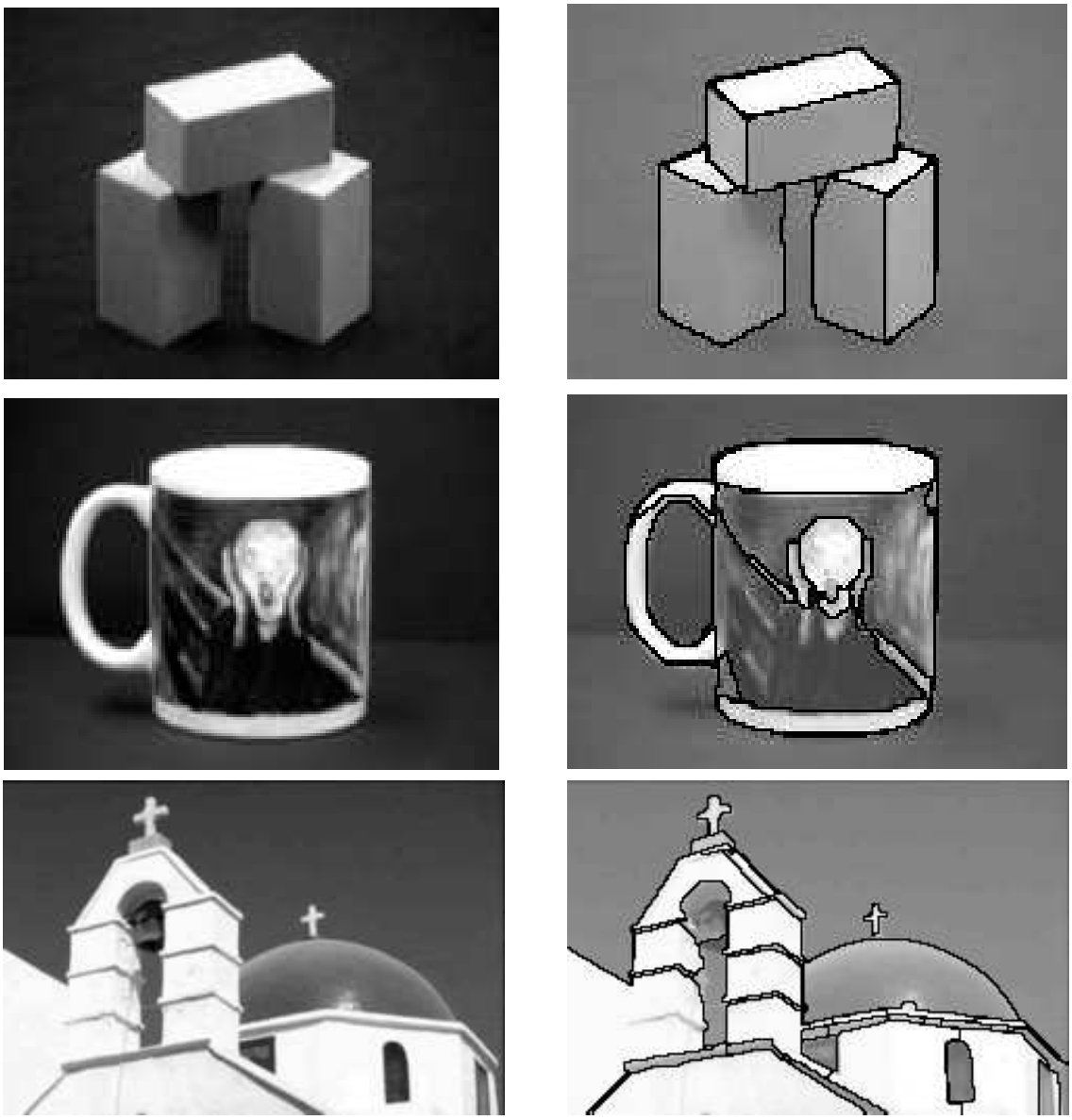
For input we primarily consider image brightness here, although similar techniques can be used with colour, motion, and/or stereo disparity information.

Readings: See Chapter 14 of Forsyth and Ponce.

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**). Do they correspond to surfaces?

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

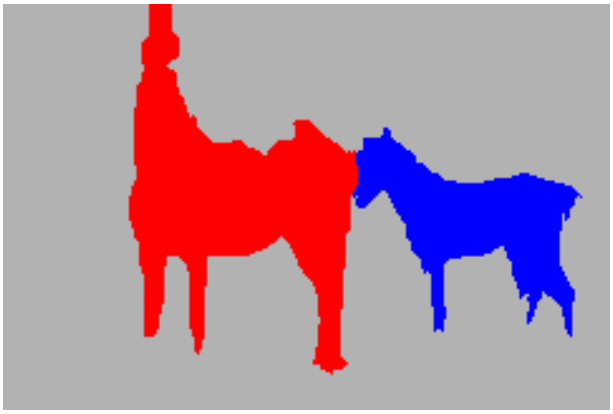
1. Follow pointing gesture.
2. Acquire image.
3. horse is an animal
4. animal ⇝ quadruped
5. baby horse ⇝ small horse

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

Image

Bottom-Up Segmentation

Parse of Two Quadrupeds



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

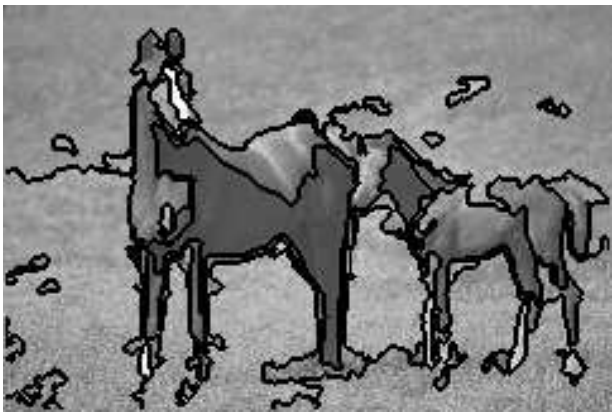
Key Questions

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

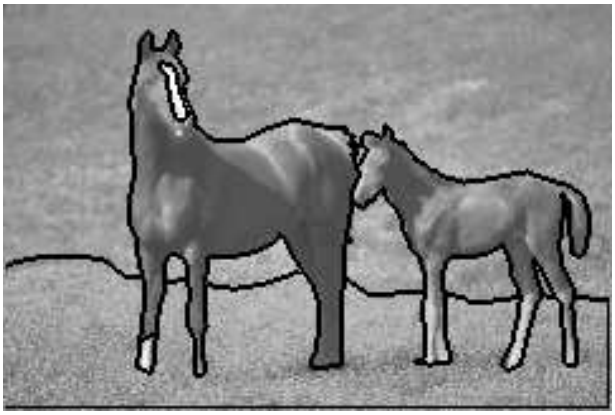
Original Image

LV

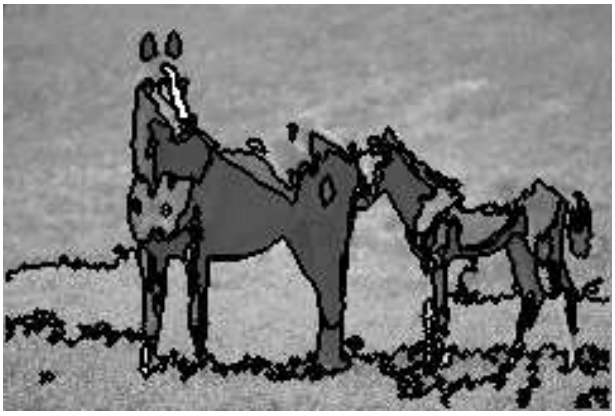


SMC

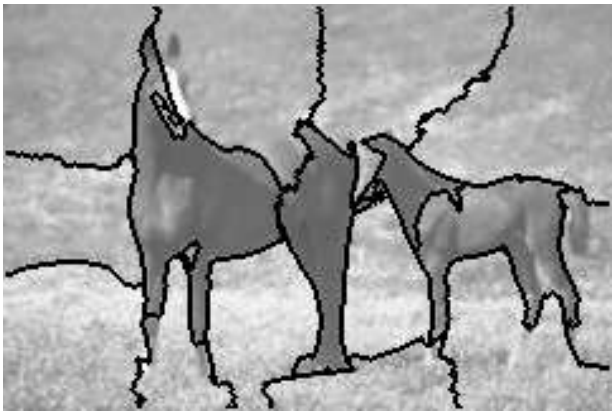
H



ED



NC



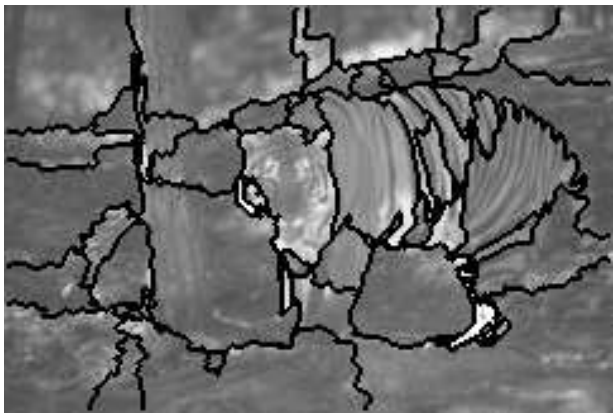
Which is the best segmentation? Why?

Example Segmentations: Tiger Image

Original Image

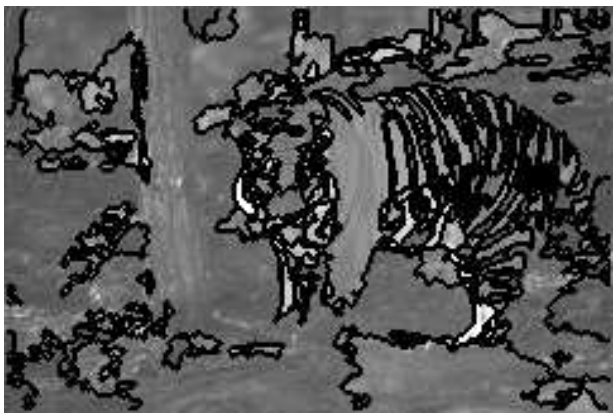
LV

SMC

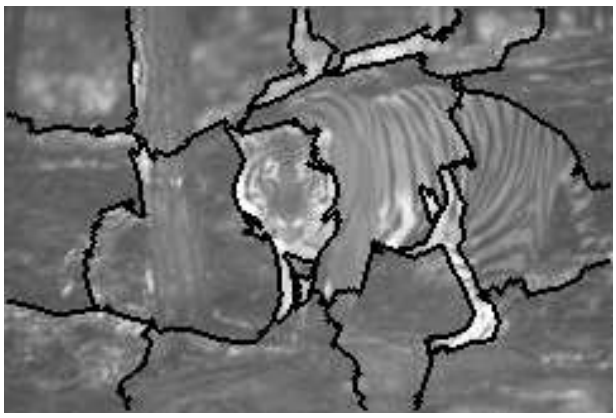


H

ED



NC



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

Observations on Example Segmentations

The previous segmentations were done by

LV: local variation algorithm [8],

SMC: spectral min-cut [7],

H: human [10, 9],

ED: edge-augmented mean-shift [5, 4],

NC: normalized cut [12, 6].

Remarks:

- 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.
- For simple images (p. 2) it is plausible that machine segmentations like those above are useful for visual tasks, e.g., object recognition.
- For more complex images (pp. 5, 6), machine segmentations provide a less reliable indicator for surface boundaries, and their utility for subsequent processing is questionable.
- While many algorithms work well with simple images, they break down with clutter and camouflage. The assessment of segmentation algorithms therefore needs to be done on standardized datasets.
- Humans probably use object recognition in conjunction with segmentation, while the machine algorithms above do not.

Current Goals

- Provide a brief introduction to the current image segmentation literature, including:
  - Feature space clustering approaches.
    - EDISON
  - Graph-based approaches.
    - Total Variation
    - Ncut
    - Spectral Min Cut
- 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

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

$\vec{L}(\vec{x})$ 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 $\vec{x}$.

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,

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

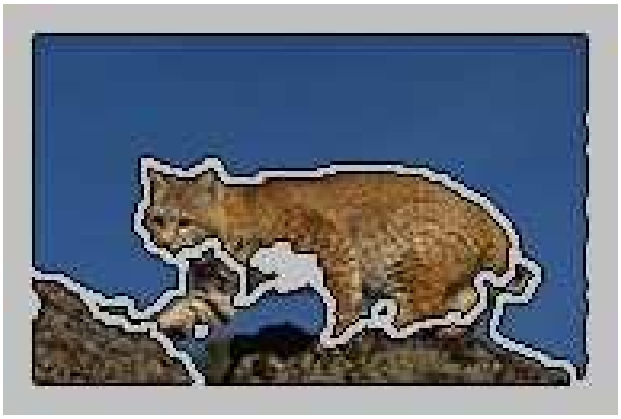
Here, $\pi_k \geq 0$ are the mixing coefficients, with $\sum_{k=1}^{K} \pi_k = 1$. Further, $\vec{m}_k$ and Σ_k are the means and covariances of the component Gaussians.

- For a given K, the parameters of the MoG model $\{(\pi_k, \vec{m}_k, \Sigma_k)\}_{k=1}^{K}$ can be fit to the features $\{\vec{F}(\vec{x})\}_{\vec{x}\in X}$ using maximum-likelihood (X denotes the set of all pixels).
- Penalized likelihood (aka minimum description length (MDL)) can be used to select the number of components, K.

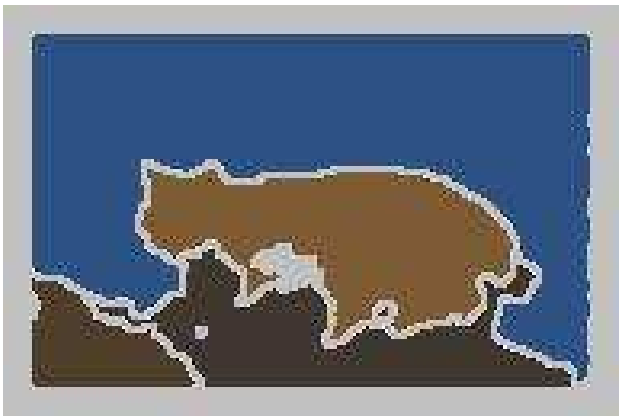
Maximum Ownership Labelling

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,

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



Above right: result for $K = 3$. The max-ownership image was post-processed using connected components. Small regions were discarded (gray). Right image: average colour for each remaining component. The width of the segment boundaries is due to the use of a spatial texture feature.



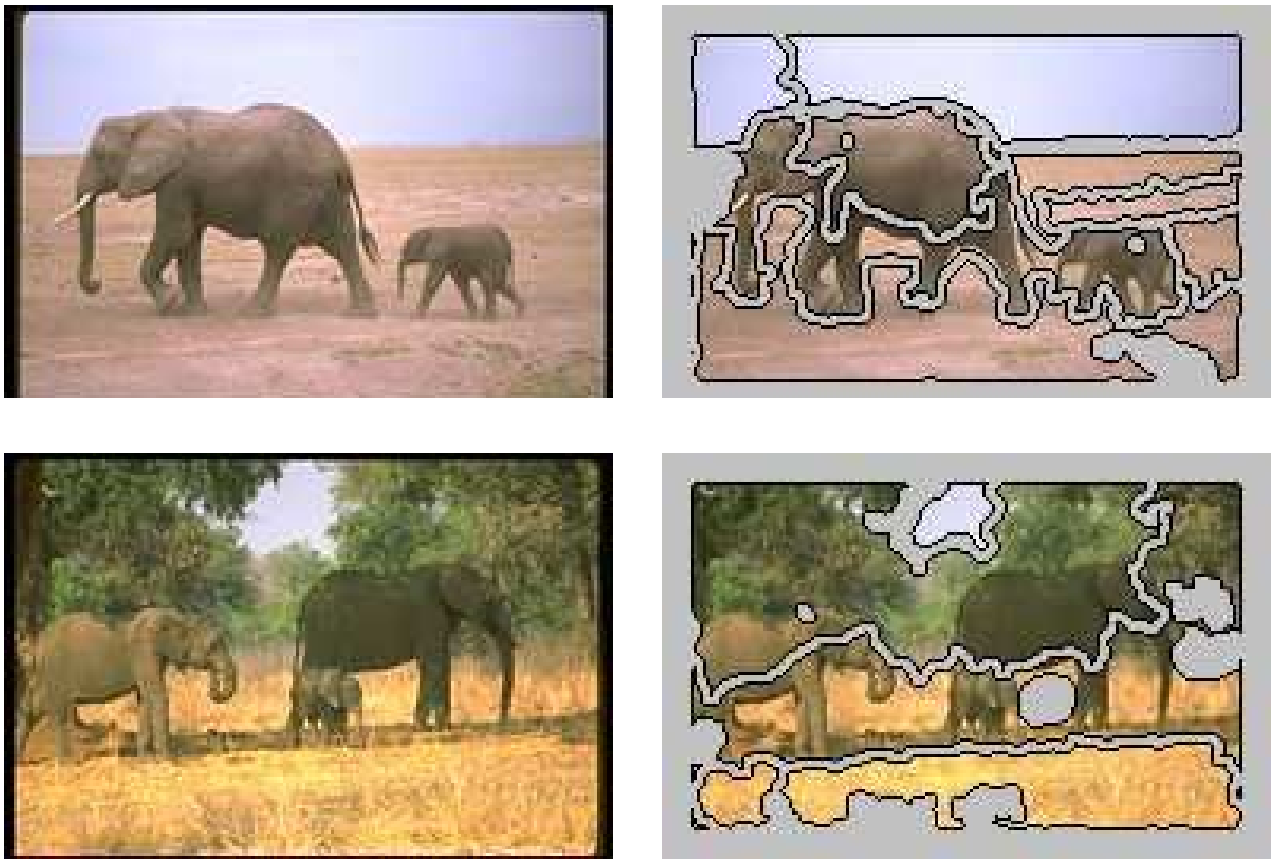
From Blobworld [3].

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 (and which led to the ML formulation); that is,

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



From Blobworld [3].

Nevertheless, this feature space clustering can be useful for extracting rough summaries of image content suitable for image retrieval [3].

Mean-Shift Segmentation

The mean-shift segmentation algorithm [5] also considers the probability density of feature vectors $\vec{F}(\vec{x})$ obtained from a given image. However, a **non-parametric** density model is used instead of an MoG.

Given features $\vec{F}_j \equiv \vec{F}(\vec{x}_j)$, the kernel-density estimate is

$$p_K(\vec{F}) \equiv \frac{1}{|X|} \sum_{j=1}^{|X|} K(\vec{F}_j - \vec{F}) , \quad \text{with } \vec{F} \in R^D ,$$

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

Common choices for $K(\vec{e})$ are functions of a (covariance) matrix Σ and the squared deviation $s \equiv \vec{e}^T \Sigma^{-1} \vec{e} \geq 0$. That is,

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

where $k(s)$ is a concave decreasing function of s. For example,

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

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

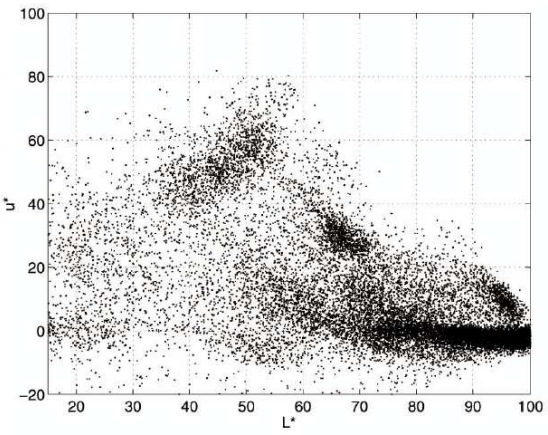
Here, $c = c(\Sigma)$ is a normalizing constant, and $\lfloor z \rfloor_+$ denotes positive rectification, i.e. $\lfloor z \rfloor_+ \equiv \max(z, 0)$.

We show an example next.

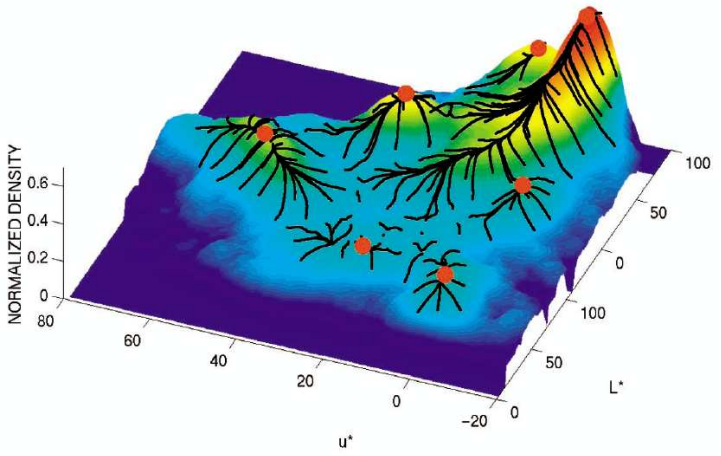
Example Feature Density

From Comaniciu and Meer [5].

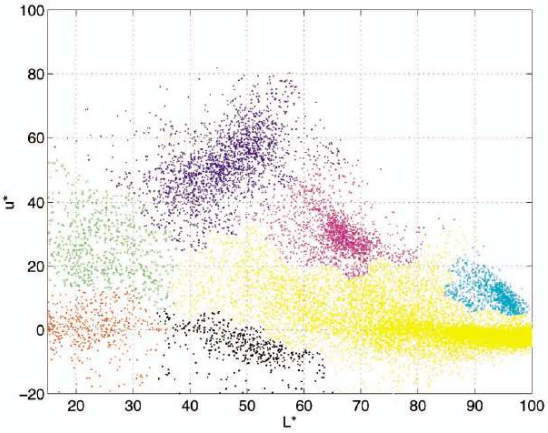
Image



Scatter plot (2D L*u*)



Mean Shift Trajectories



Mean Shift Clusters

The kernel density estimate using the Epanechnikov kernel on the 2d feature points (top right). 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, but 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

$$\vec{F}_* \;=\; \frac{\sum_j w(\vec{F}_j - \vec{F}_*) \; \vec{F}_j}{\sum_j w(\vec{F}_j - \vec{F}_*)} \, ,$$

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 $\vec{F}_j$ using the weights $w(\vec{F}_j - \vec{F}_*)$ 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

$$\vec{F}^{(t+1)} \;=\; \frac{\sum_j w(\vec{F}_j - \vec{F}^{(t)}) \; \vec{F}_j}{\sum_j w(\vec{F}_j - \vec{F}^{(t)})} \, . \tag{4}$$

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

Details of Mean-Shift Iterations

Let $\vec{F}_j \equiv \vec{F}(\vec{x}_j)$ be the feature vector at pixel $\vec{x}_j$. We define a kernel density estimate for the distribution of the image features $\vec{F}$ as follows:

$$p(\vec{F}) \;=\; \frac{1}{|X|} \sum_{j=1}^{|X|} k(s_j(\vec{F}))\,, \tag{5}$$

where $|X|$ is the number of pixels in the image, k be a non-negative kernel function, and, given a positive definite matrix C,

$$s_j(\vec{F}) \equiv (\vec{F} - \vec{F}_j)^T C^{-1} (\vec{F} - \vec{F}_j) \tag{6}$$

measures the scaled deviation between $\vec{F}$ and $\vec{F}_j$. We further assume that the kernel integrates to one in order for (5) to be a valid density function. And let $w(s) = \frac{dk}{ds}(s) \equiv k'(s)$ be the kernel derivative.

The goal of the mean-shift iterations is to find a trajectory to local maxima of $p(\vec{F})$, beginning at an arbitrary point in the feature space. A necessary condition for a local maxima of the kernel density estimate is that the gradient be zero. That is, for $\vec{F}^*$ to be a critical point it must satisfy

$$\frac{\partial p_k}{\partial \vec{F}}(\vec{F}^*) \;=\; \vec{0}\,. \tag{7}$$

Thus, to find critical points let's look at the form of the derivative of the feature distribution:

$$\begin{aligned}
\frac{\partial p_k}{\partial \vec{F}}(\vec{F}) &= \frac{1}{|X|} \sum_j \frac{\partial k}{\partial \vec{F}}(s_j(\vec{F})) \\
&= \frac{1}{|X|} \sum_j \frac{\partial k}{\partial s}\Big|_{s=s_j} \frac{\partial s_j}{\partial \vec{F}} \\
&= \frac{1}{|X|} \sum_j w(s_j(\vec{F})) \left[2C^{-1}\vec{F} - 2C^{-1}\vec{F}_j\right]
\end{aligned} \tag{8}$$

With some algebraic simplification, we find that that the critical points satisfy

$$\left[\sum_j w(s_j(\vec{F}^*))\right] \vec{F}^* \;=\; \sum_j w(s_j(\vec{F}^*))\, \vec{F}_j\,. \tag{9}$$

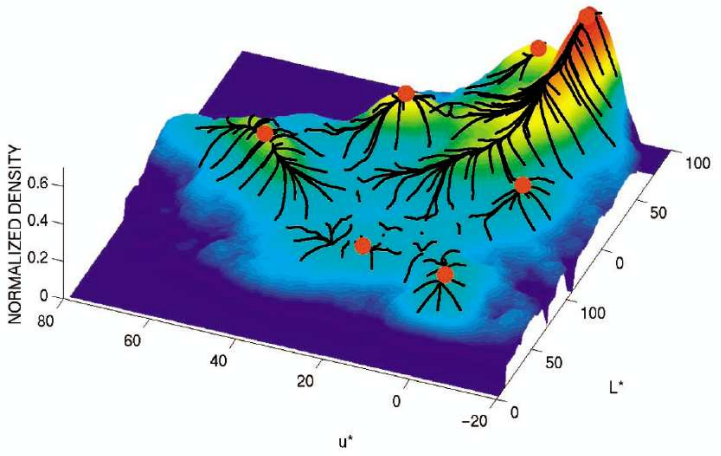
To find a critical point, given an initial guess, $\vec{F}^{(t)}$, we compute the weights using $\vec{F}^{(t)}$. Then, holding the weights fixed we solve for the next point on the trajectory toward the critical point; i.e.,

$$\left[\sum_j w(s_j(\vec{F}^{(t)}))\right] \vec{F}^{(t+1)} \;=\; \sum_j w(s_j(\vec{F}^{(t)}))\, \vec{F}_j, \tag{10}$$

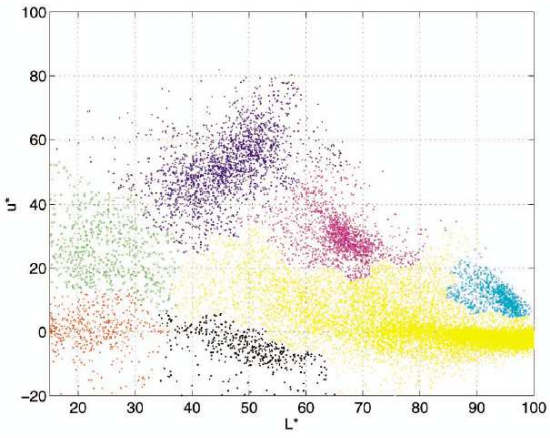
from which we obtain (4).

Watersheds of Mean-Shift

The label at pixel $\vec{x}_0$ is defined to be the mode to which mean shift iterations (4) converge when started at $\vec{F}^{(0)} = \vec{F}(\vec{x}_0)$. That is, the *segments* are the **domains of convergence** (aka watersheds) of the mean-shift iterations (right plot).



Mean Shift Trajectories



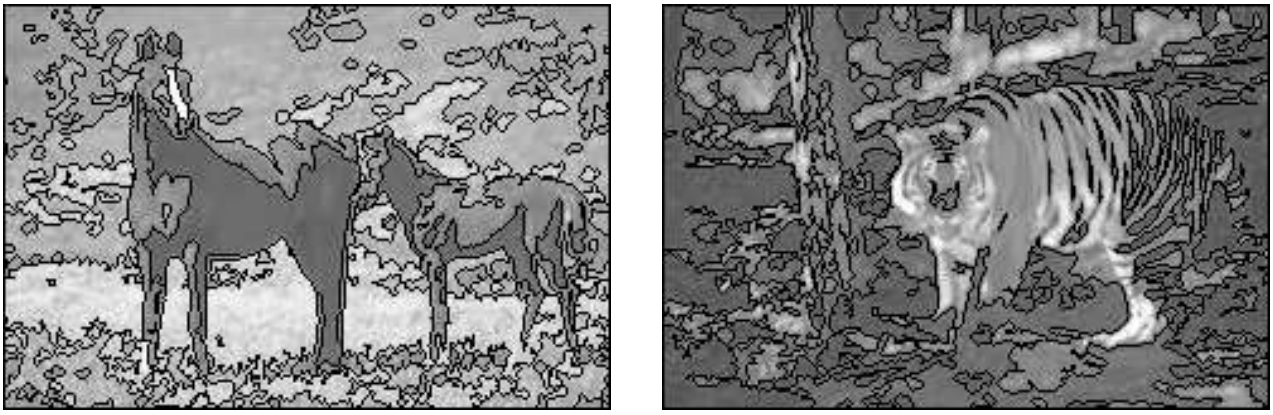
Mean Shift Clusters

Properties:

1. **Convergence:** Mean-shift iterations converge to a stationary point of $p_K(\vec{F})$ (see [5]).
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) where the gradient of $\vec{F}^{\,T}(\vec{x})\Sigma^{-1}\vec{F}(\vec{x})$ is constant. Where $\vec{\nabla} I(\vec{x})$ is constant, points of $p_K(\vec{F})$ are stationary and mean-shift iterations stall (see left fig). Postprocessing is used to select salient local maxima (see [5]).

Example Mean-Shift Segmentations

Segmentations from the basic mean-shift algorithm:



The scale of the mean-shift kernel (controlled by Σ) roughly determines 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 [4]) 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 $\mathcal{G} = (V, E)$ is formed, with the set of vertices V corresponding to the pixels $\vec{x}$ in the image. Edges E in the graph occur 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

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

As above, $\vec{F}(\vec{x})$ 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.

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** is a greedy approach that is guaranteed to give an optimal MST:

- Begin with the completely disconnected graph.
- Add edges one at a time in increasing order of their weights, so long as adding an edge does not introduce cycles in the 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.

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 that gives a useful segmentation.

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

Local Variation Method

Felzenszwalb and Huttenlocher [8] 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,
- it maintains a forest of MSTs for its current components.

During processing, each MST C_i is associated with a threshold

$$T(C_i) \;=\; w(C_i) + \frac{k}{|C_i|}\,, \tag{11}$$

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

$$w(\vec{x}_k, \vec{x}_l) \;\leq\; \min(T(C_i), T(C_j))\,. \tag{12}$$

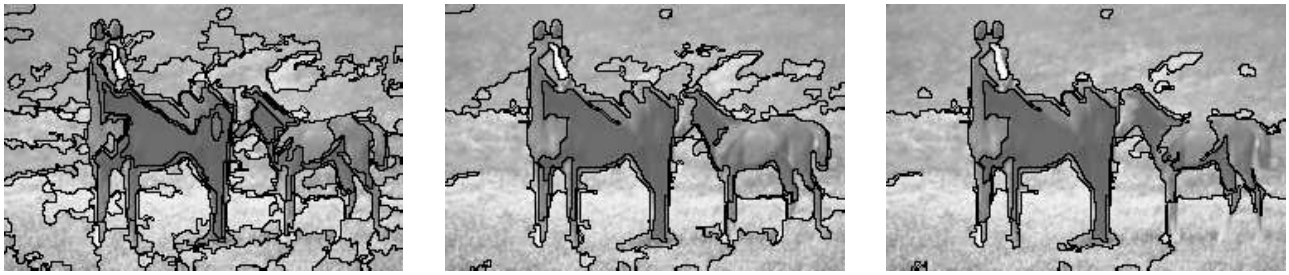
As the size of C_i increases, (11) and (12) dictate an increasingly tight upper bound $T(C_i)$ (compared to the largest weight $w(C_i)$ in C_i) for the acceptable affinity of an edge merging C_i with another region.

Examples of Local Variation Segmentation

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

Parameter k in (11) roughly controls the size of the regions in the segmentation. Larger k yields a looser constraint (12), and more merging.

k = 50 k = 150 k = 250



Merging is sensitive to the local variation within regions. Due to the increasingly tight bound (11), 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. The bound is looser for small regions, encouraging their growth, thereby avoiding many tiny regions.

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 $e = |E|$ is the number of edges.

The weak link: But two very different regions can be merged if there is even one edge with a small weight that joins them.

Graph Cuts

Graph partitioning formulations are intended to be more robust to small leaks, since the objective criteria naturally take into account the entire boundary of a hypothetical segment.

Suppose we want to partition (i.e., cut) the graph into two parts, F and $G = V - F$. Further, we only want to cut edges with small affinities (ie keep similar nodes together in the same partition). Thus, one might measure the cost of the cut using

$$L(F, G) = \sum_{\vec{x}_i \in F, \vec{x}_j \in G} a(\vec{x}_i, \vec{x}_j) \,, \tag{13}$$

where $a(\cdot, \cdot)$ is the affinity function described above.

Problem: Optimal cuts often result in F (or G) containing one node.

Solution: Formulate more suitable objective functions or graph constructions to place constraints on F and G.

Normalized Cut

The normalized cut approach of Shi and Malik [12] avoided trivial partitions of the affinity weighted graph with the use of the *normalized cut criterion*. They proposed that the optimal partition of the graph into F and $G = V - F$ should minimize

$$N(F, G) \equiv L(F, G) \left(\frac{1}{L(F, V)} + \frac{1}{L(G, V)} \right), \tag{14}$$

where L is the linkage defined in (13); i.e., $L(A, B)$ is the sum of the affinities for all edges having one end in A and the other in B.

Properties:

- The cut cost decreases when edges connecting nodes in F and G have small affinities (i.e., for small $L(F, G)$).
- $L(F, V)$ is sum of affinities on edges within F, and those connecting F to the rest of the graph. And similarly for $L(G, V)$.
- Other things being equal, the cut cost decreases when the affinities on edges within F and G are larger (i.e., for *homogeneous* regions).

Problem: Unfortunately, the resulting graph partitioning problem,

$$F = \arg \min_{F \subset V} N(F, V - F), \tag{15}$$

is computationally intractable [12]. Therefore we must seek algorithms that provide approximate solutions of (15).

Normalized Cut (cont)

Shi and Malik [12] prove that (15) is equivalent to the discrete optimization problem, for labelling $\vec{y} = (y_1, ..., y_N)^T$ with $N = |X|$,

$$\arg\min_{\vec{y}} \frac{\vec{y}^T(D-A)\vec{y}}{\vec{y}^T D\vec{y}}\ , \quad \text{subject to } y_i \in \{1, -b\} \text{ and } \vec{d}^T\vec{y} = 0\ . \quad (16)$$

Here,

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

Given a solution $\vec{y}$ of (16), the corresponding solution for partition F of (15) 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 $\vec{d}^T\vec{y} = 0$.

Shi and Malik [12] proposed to relax the discrete problem. They first find a real-valued solution to (16), for which they ignore the binary constraint that $y_i \in \{1, -b\}$. Then they threshold the result to produce a discrete (approximate) solution.

Rayleigh Quotient

Equation (16) is a discrete version of a standard eigenvector formulation, namely the Rayleigh quotient.

By way of background, the matrix $D - A$ is known as the Laplacian matrix. Among its generalized eigenvectors, $\vec{y}$, satisfying

$$(D - A)\,\vec{y} \;=\; \mu\, D\,\vec{y}\,, \tag{17}$$

the eigenvector corresponding to the smallest eigenvector is known to be the solution to the following problem:

$$\arg\min_{\vec{y}} \frac{\vec{y}^{\,T}(D - A)\vec{y}}{\vec{y}^{\,T} D \vec{y}}\,. \tag{18}$$

It is well-known that because $D - A$ is symmetric, real, positive semidefinite, that (1) $D - A$ has orthogonal eigenvectors, and (2) that the associated eigenvalues must be greater than or equal to zero. More specifically the vector of all ones $\vec{1}$ is known to be an eigenvector of $D - A$, with eigenvalue 0.

Now consider the relaxation of (16), in which we ignore the constraint that $y_i \in \{1, -b\}$, and therefore treat $\vec{y}$ as real-valued. Further consider the transformation $\vec{y} = D^{-1/2}\vec{u}$, for which the relaxed problem becomes

$$\arg\min_{\vec{u}} \frac{\vec{u}^{\,T}(I - B)\vec{u}}{\vec{u}^{\,T}\vec{u}}\,, \quad \text{subject to } \vec{d}^{\,T/2}\vec{u} = 0\,, \tag{19}$$

for symmetric matrix $B = D^{-1/2} A D^{-1/2}$, and real-valued $\vec{u}$. This is a standard eigenvalue problem in linear algebra. In particular, it can be shown that

- $\vec{u} = \vec{d}^{\,1/2}$ is an eigenvector of B with eigenvalue 1, so it must also be an eigenvector of $I - B$ with eigenvalue 0;
- since $D - A$ is symmetric positive semidefinite, so is $I - B$, and therefore all the eigenvalues associated with $I - B$ are greater than or equal to zero, and the corresponding eigenvectors are orthogonal;
- all the eigenvalues of $I - B$ are in the interval $[0, 2]$.

Hence, from (19) it follows that the linear constraint on $\vec{u}$ simply says that $\vec{u}$ must be orthogonal to the eigenvector of $I - B$ having the smallest eigenvalue. Accordingly, the vector minimizing the Rayleigh Quotient and orthogonal to $\vec{d}^{\,1/2}$ is simply the eigenvector of $I - B$ with the *second smallest* eigenvalue.

Spectral Approximation for Normalized Cut

The relaxed version of (16), ignoring the binary constraint $y_i \in \{1, -b\}$, and taking $\vec{y}$ to be real-valued, is a standard problem in linear algebra known as the Rayleigh Quotient.

- The solution $\vec{y}$ is the generalized eigenvector of the Laplacian matrix, $D - A$, with the second smallest eigenvalue μ:

$$(D - A)\,\vec{y} \;=\; \mu\, D\,\vec{y}\,. \tag{20}$$

To obtain a discrete (approximate) solution to (16), Shi and Malik threshold $\vec{y}$. They consider several thresholds τ, and choose the one with the lowest value of the Ncut objective function (14). Regions, F and $G = V - F$, are recursively partitioned using the same method, until a user-specified number of segments are found. (see pp.5-6.)

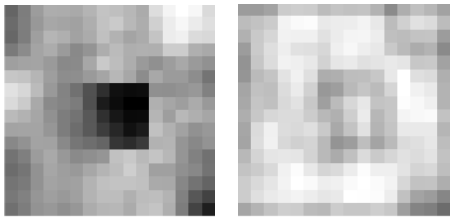
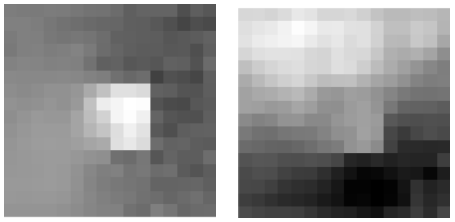


image eig 1



eig 2 eig 3

image

eig 1

eig 2

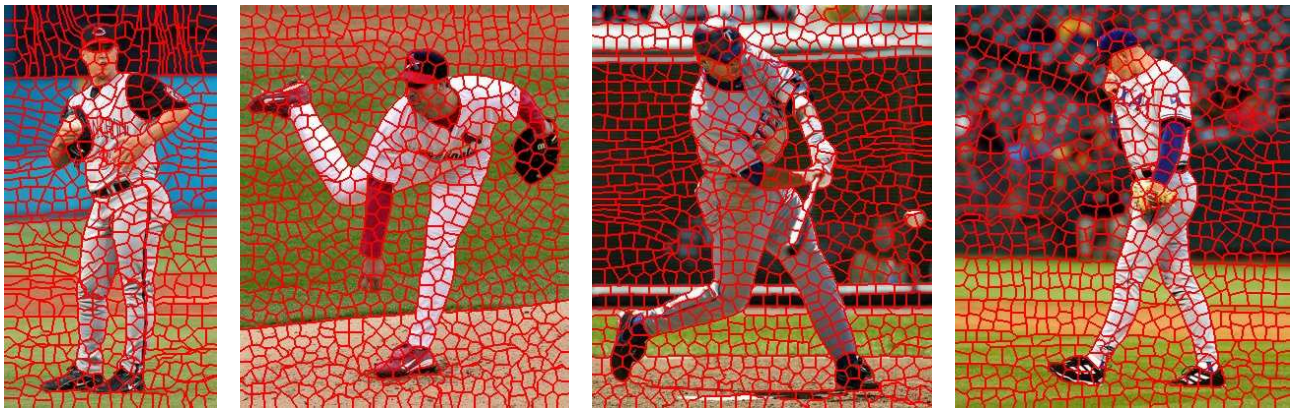
eig 3

Remarks on Ncuts

The step of thresholding the second largest eigenvector to provide a partitioning proposal is a key limitation. In practice, the approximation only appears to be consistently reliable when there is just one clear way to partition the data. Yu and Shi [13] attempt to alleviate this problem by extracting K segments from the subspace spanned by the K eigenvectors of $I - B$ with the smallest eigenvalues.

Ncuts can be very slow for large images since the matrix becomes prohibitively large. Efficient approximations to the eigenvector problem have also been proposed.

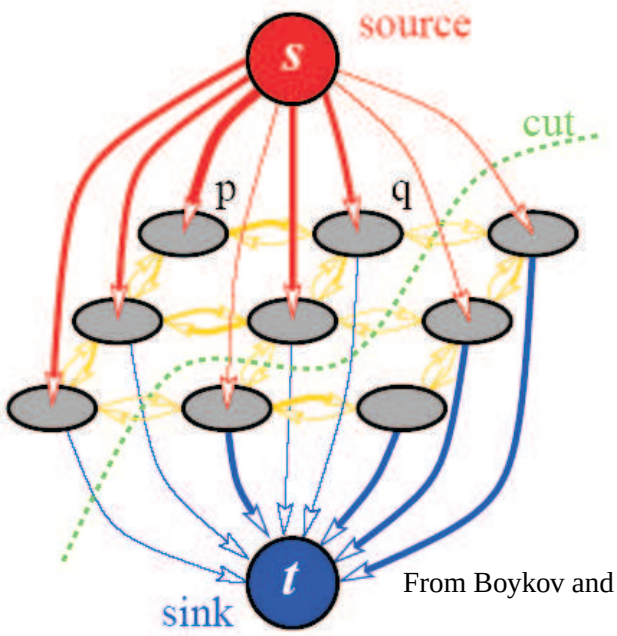
A common use of Ncuts is for the computation of significantly over-segmented images (called superpixels), to propose atomic (indivisible) regions as a basis for subsequent processing.



For further information, see the reading list in the CVPR 2004, graph-based segmentation tutorial [11].

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.



From Boykov and Kolmogorov [2].

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 F and $G = V - F$, $s \in F$, and $t \in G$, that minimizes the linkage

$$L(F, G) \;=\; \sum_{\vec{x}_i \in F,\, \vec{x}_j \in G} a(\vec{x}_i, \vec{x}_j)\;. \tag{21}$$

Efficient algorithms exist to solve the S-T min-cut problem (see [2]).

The S-T min-cut problem is computationally simpler than the general **graph partitioning problem** (to find a non-empty partition F and G that minimizes $L(F, G)$, i.e., without constraints like $s \in F$ and $t \in G$).

Source-Sink Minimum Cut (Cont.)

To take advantage of efficient solutions to the S-T min-cut problem, we first need to construct 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 graphs, we include two directed edges $\langle \vec{x}_i, \vec{x}_j \rangle$ and $\langle \vec{x}_j, \vec{x}_i \rangle$. Both are weighted by the affinity $a(\vec{x}_i, \vec{x}_j)$.
- Two additional nodes s and t are created, namely the source and sink nodes, respectively.
- 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 $\vec{x}_j \in T$. This ensures that nodes in S and T can never be cut from s and t.

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

Seed Regions for S-T Min Cut

Sets S and T (nodes connected to source and sink) 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.

Interactive Min-Cut:

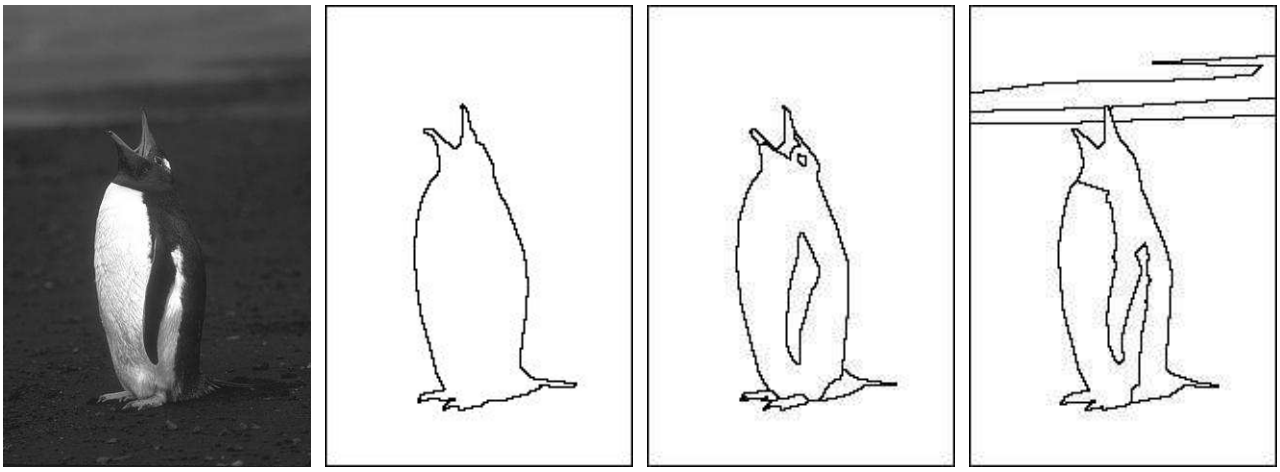
Spectral Min-Cut: One suitable generation process is discussed in Estrada et al. [7]. It is based on spectral properties of a matrix representing the affinities. Sample results are given by the **Spectral Min-Cut** technique (SMC on pp. 5,6). The process is much more computationally intensive than the previous ones. Several hundred min-cut problems are typically solved for different S, T.

Berkeley Segmentation Database

The Berkeley Segmentation Dataset [9, 10] 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.

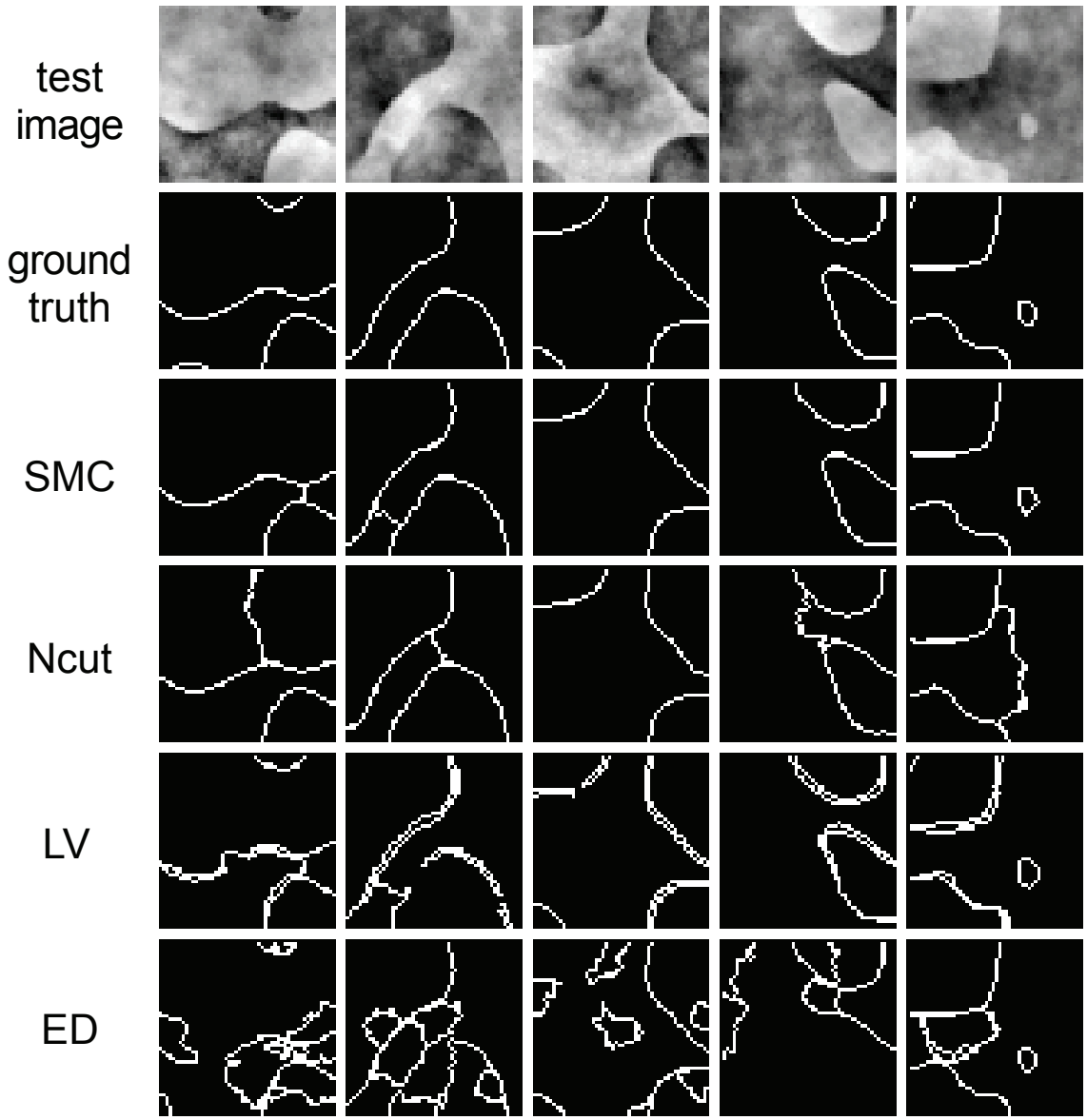
The public portion of this dataset consists of the segmentations of 300 images by roughly 5 humans each, done separately for greylevel and colour versions of the images. Three examples from one image are shown below:



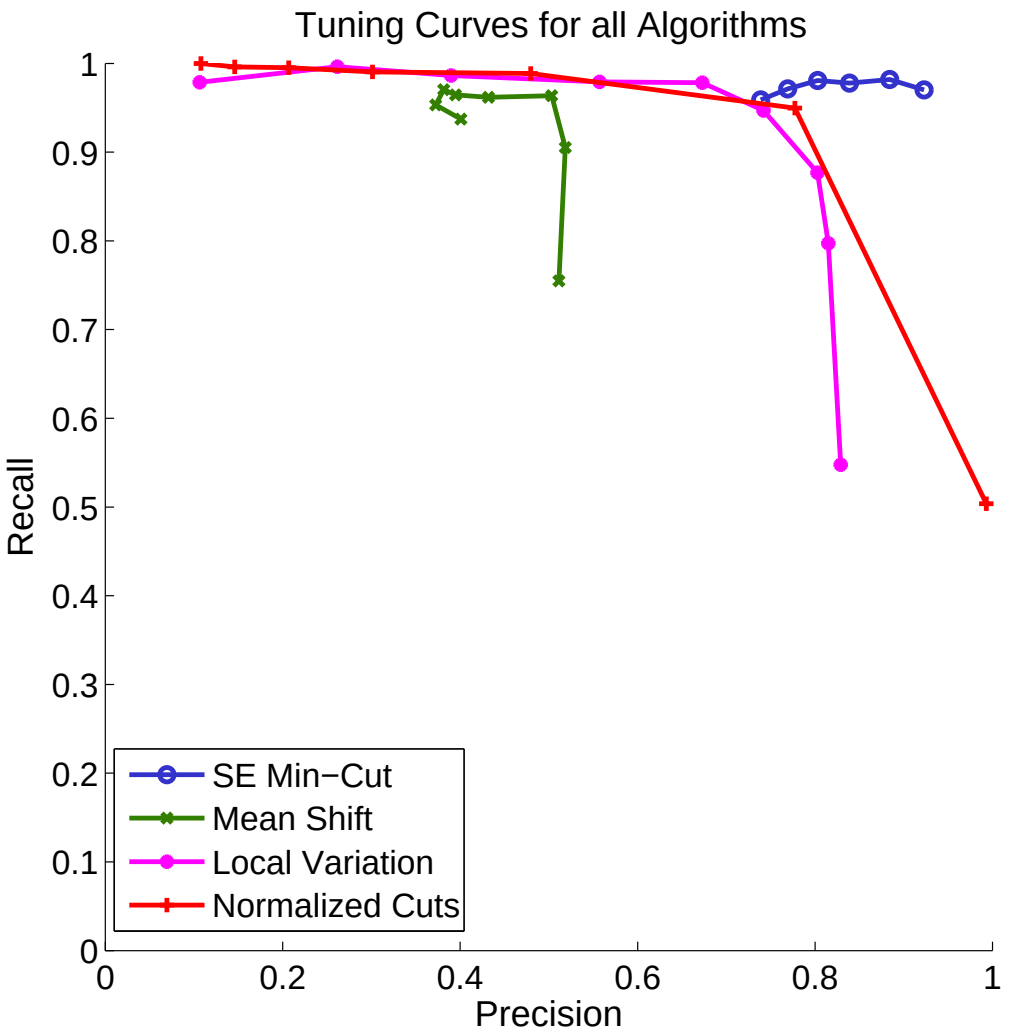
Note these segmentations appear to be consistent, except different subjects have decided to resolve particular regions into more or less detail. This variability should be taken into account in a quantitative comparison of two segmentations.

Benchmarking Segmentation

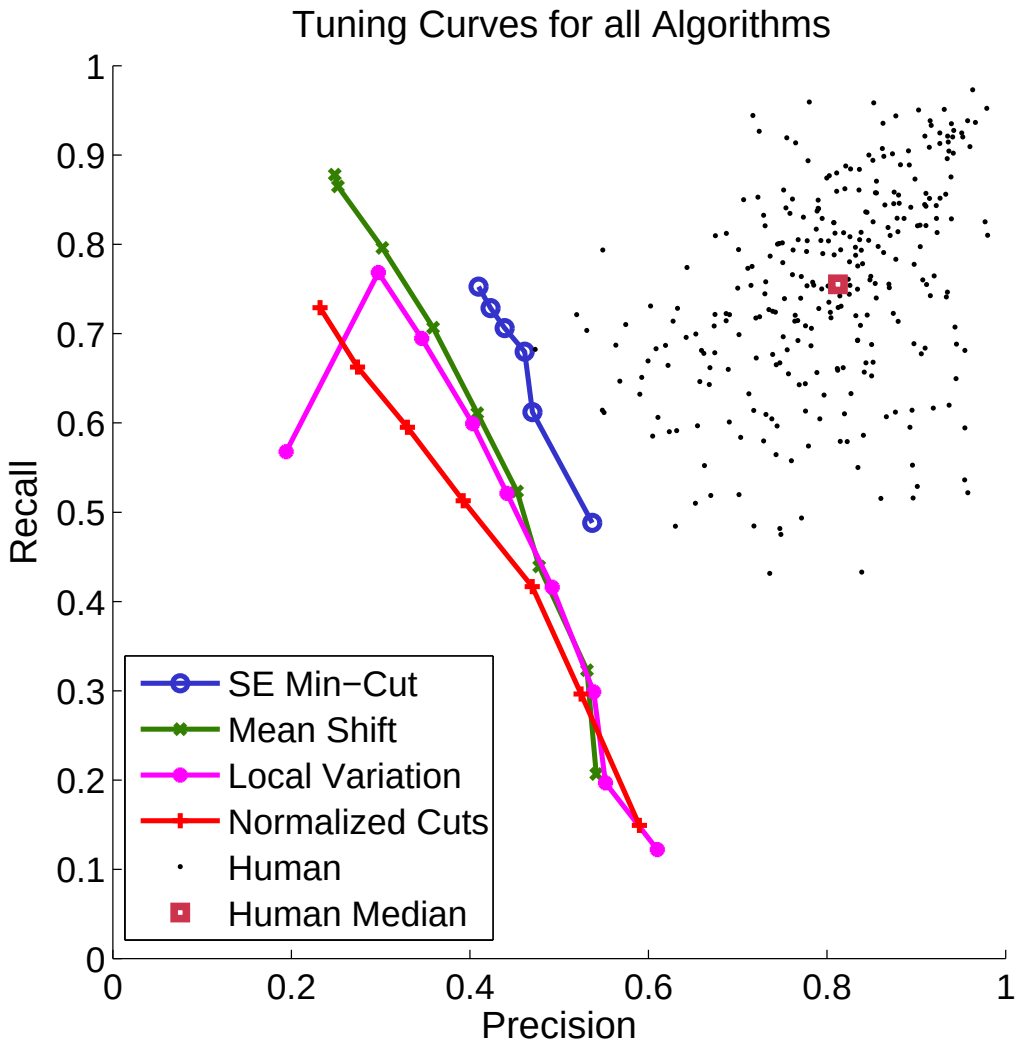
Segmentation algorithms have been benchmarked on synthetic fractal images. The precision-recall curves for the detection of segment boundary points were computed (Estrada and Jepson, 2004).



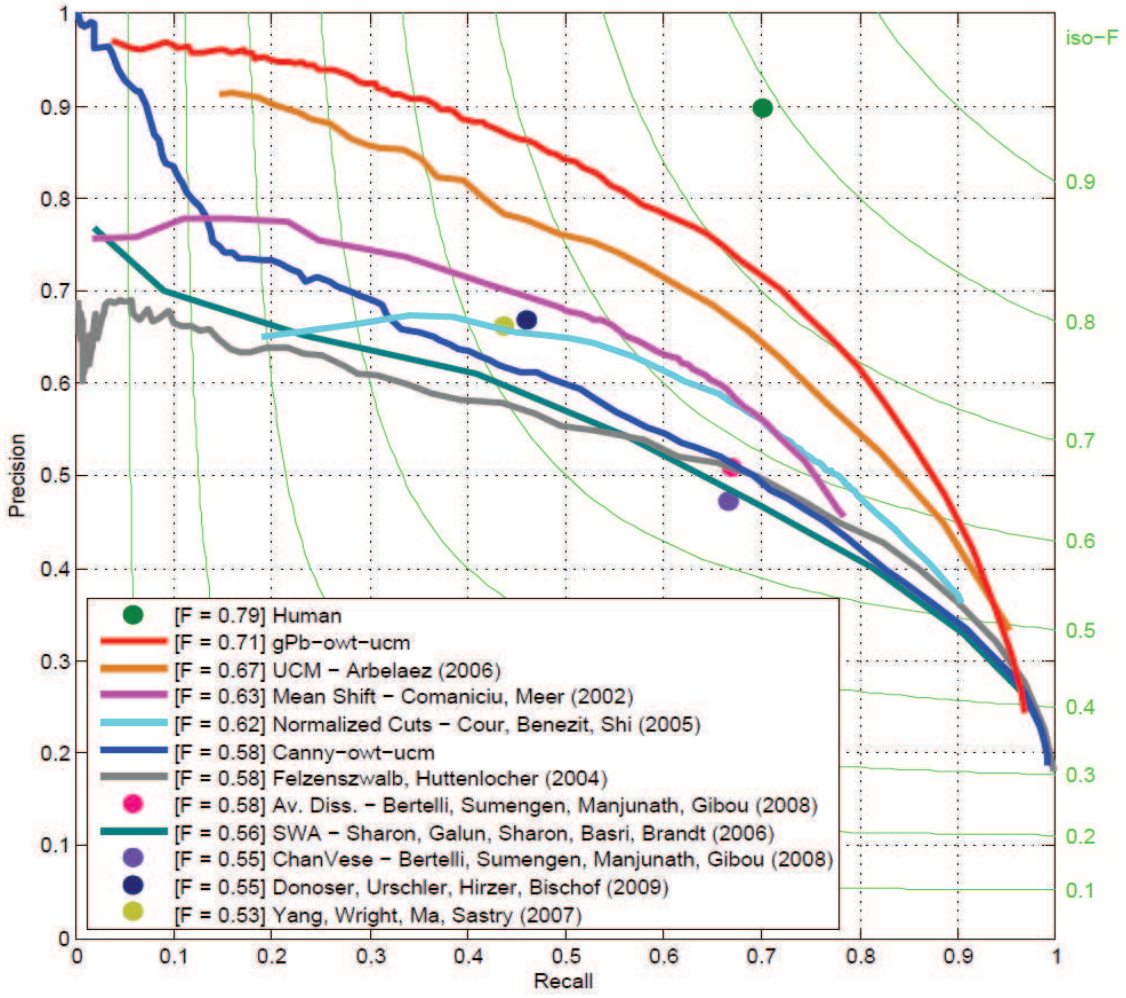
Precision-Recall on Fractal Images



Precision-Recall on Berkeley Dataset



Precision-Recall on Berkeley Dataset



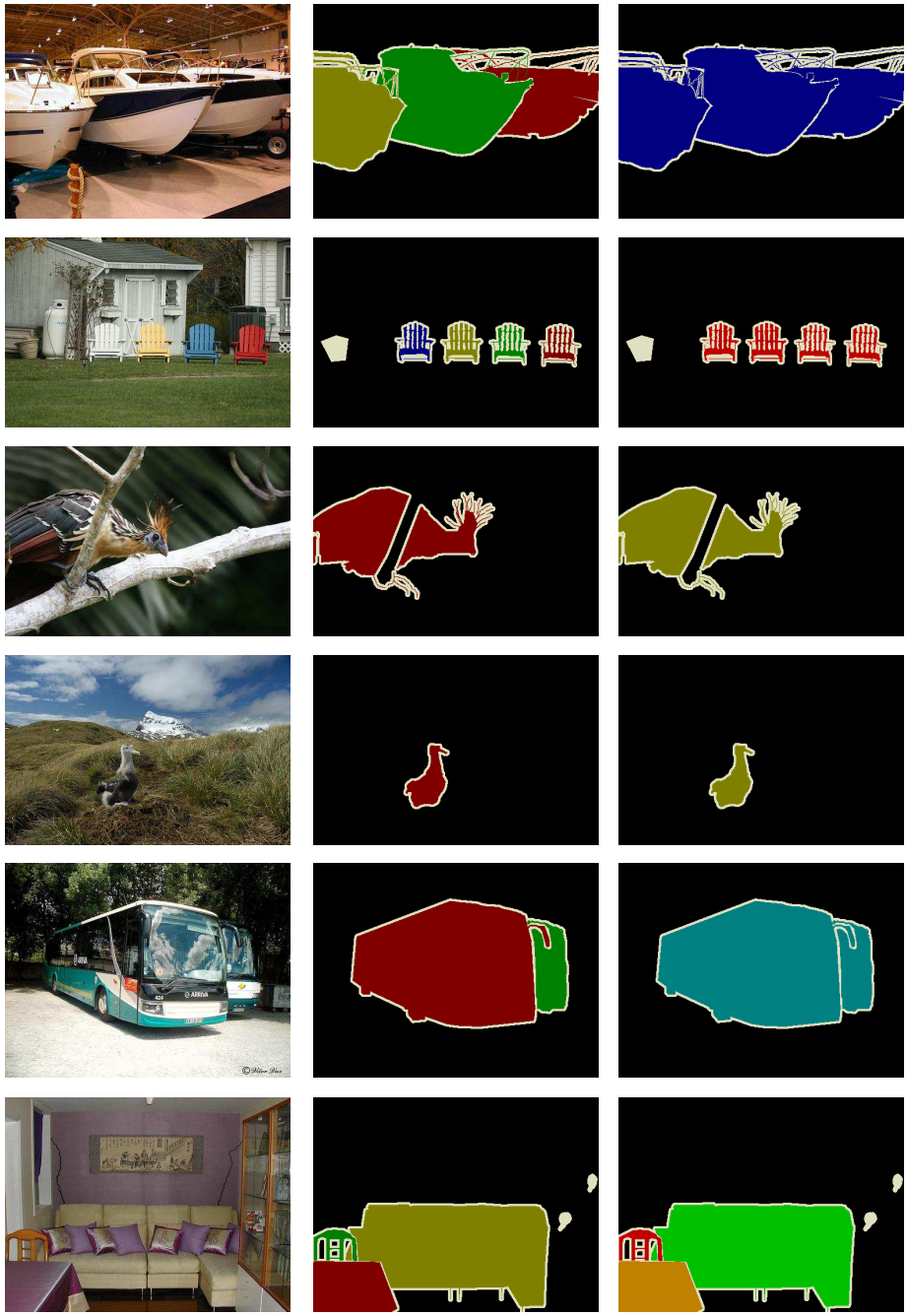
Here the F-score is the maximum of the harmonic mean of precision and recall

$$F = \max\left[\frac{2}{\frac{1}{precision} + \frac{1}{recall}}\right] = \max\left[2\,\frac{precision \cdot recall}{precision + recall}\right].$$

More recent segmentation methods make use of trained image boundary detectors which incorporate both colour and texture information. See Arbeláez et al. [1]. (Note the Precision-Recall axes above have been swapped to the standard arrangement.)

Pascal VOC Challenge

Training examples for the 2009 segmentation/labelling challenge, each with (1) training image; (2) object segmentation (colors specify first, second, third object etc.); and (3) class segmentation (colors specify class: 1=aeroplane, 2=bicycle, 3=bird, 4=boat, 5=bottle, 6=bus, ...).



References

[1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. PAMI*, 33(5):898–916, 2011. See `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html`.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. PAMI*, 26(9):1124–1137, 2004.

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

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

[5] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. PAMI*, 24:603–619, 2002.

[6] 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/`.

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

[8] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004.

[9] D. Martin and C. Fowlkes. *The Berkeley Segmentation Dataset and Benchmark*. 2007. `http://www.cs.berkeley.edu/projects/vision/grouping/segbench/`.

[10] 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 *IEEE ICCV*, volume 2, pages 416–423, 2001.

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

[12] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. PAMI*, 22(8):888–905, 2000.

[13] S. Yu and J. Shi. Multiclass spectral clustering. In *IEEE ICCV*, 2003.