Graphical Models in Computer Vision

Andreas Geiger

Max Planck Institute for Intelligent Systems
Perceiving Systems

July 18, 2016

MAX-PLANCK-GESELLSCHAFT

Syllabus

11.04.2016	Introduction
18.04.2016	Graphical Models 1
25.04.2016	Graphical Models 2 (Sand 6/7)
02.05.2016	Graphical Models 3
09.05.2016	Graphical Models 4
23.05.2016	Body Models 1
30.05.2016	Body Models 2
06.06.2016	Body Models 3
13.06.2016	Body Models 4
20.06.2016	Object Detection 1
27.06.2016	Object Detection 2
04.07.2016	Stereo
11.07.2016	Optical Flow
18.07.2016	Segmentation

Todays topic

Image Segmentation

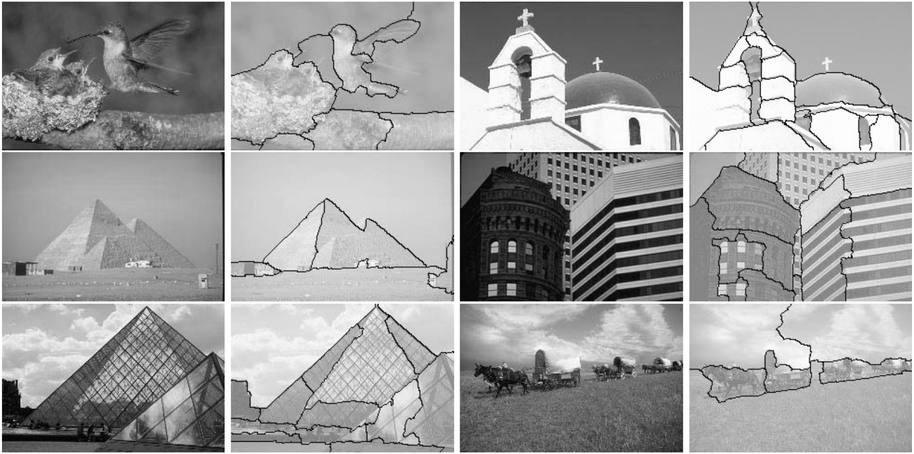
- Motivation
- Unsupervised Segmentation
  - K-means
  - Simple Linear Iterative Clustering
  - Mean-Shift
  - Spectral Clustering
- Interactive Segmentation
  - Grab Cut
- Semantic Segmentation
  - TextonBoost

Motivation

What is Image Segmentation?

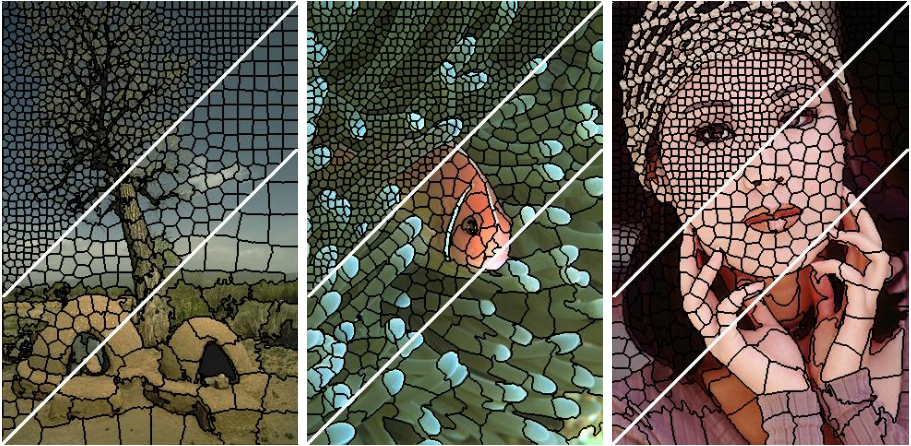
- An image contains an extremely large number of pixels
- Segmentation is the grouping of similar pixels in the image
- Thus, the representation becomes much more compact
(small number of regions instead of large number of pixels)
- This makes some tasks significantly easier
(*e.g.*, probabilistic models for recognition)
- Sometimes the segmentation is of interest itself
(*e.g.*, segmenting a tumor)

Some Examples



[Ren & Malik, 2003]

Superpixels



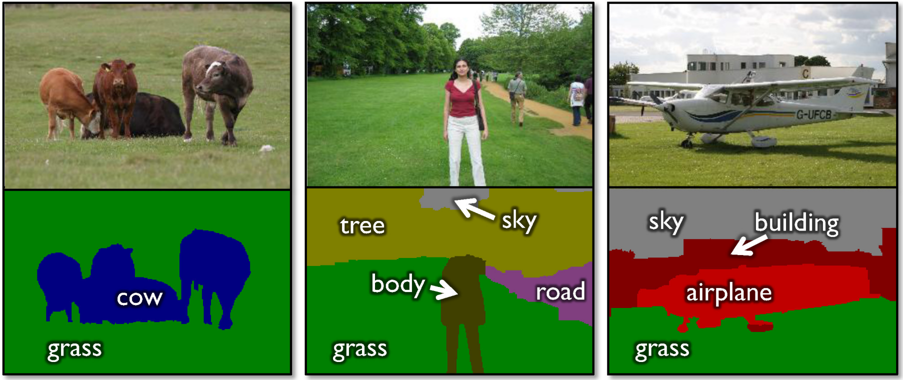
[Achanta *et al.*, 2011]

- Superpixels are a representation which can serve as substitute for pixels in many applications (*e.g.*, to lower the computational burden)

Figure-Ground Separation

- One way of thinking about segmentation is the separation of figure (*i.e.*, foreground) from ground (*i.e.*, background)
- In this case: 2 classes (boy vs. background)
- This separation can be ambiguous (separation might be possible, but we can't tell which is which)

Semantic Segmentation

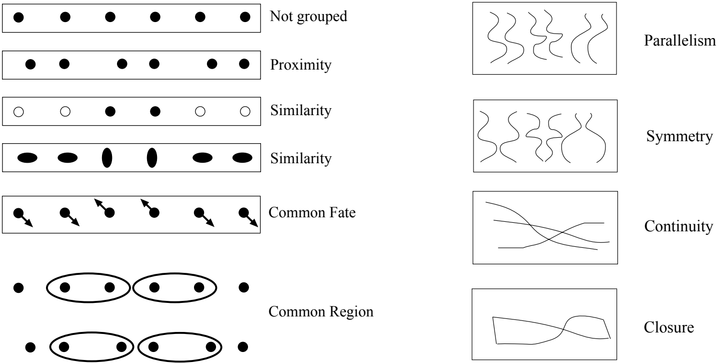


[Shotton *et al.*, 2007]

- More than 2 classes, each class has a semantic meaning
- Important for higher-level processes (*e.g.*, scene understanding)

What belongs together?

Gestalt psychology in the early 20th century (Wertheimer *et al.*)

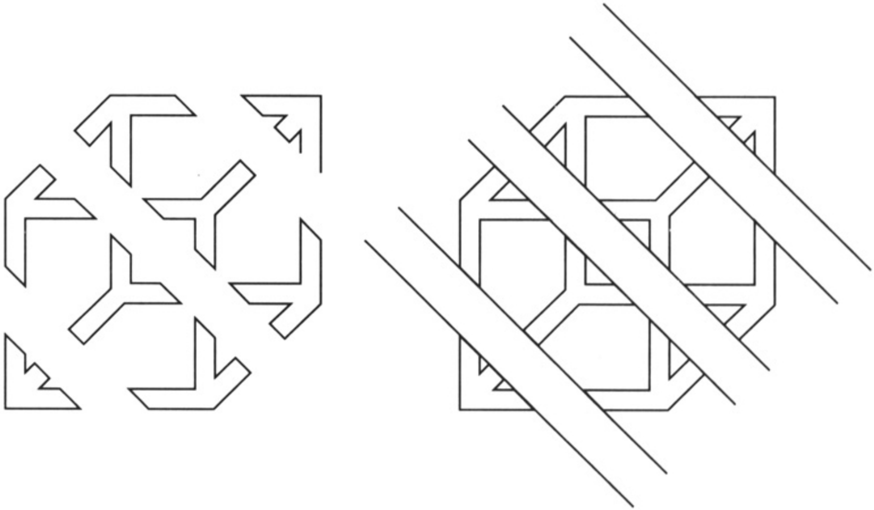


[Gordon]

- These factors offer some insights of what we want to have
- Turning them into a robust algorithm is a hard problem, however

Importance of Occlusion

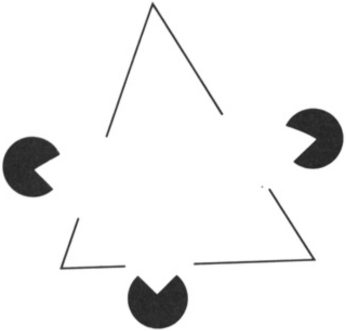
What do you see?

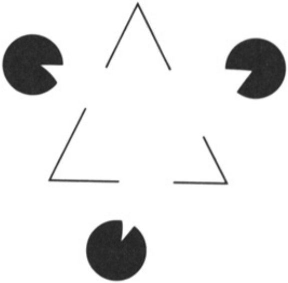


Gestalt and Occlusion Cues in Surrealism

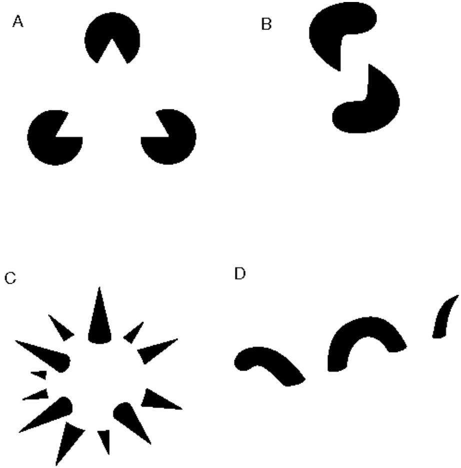
[“Le Blanc Seing”, Rene Magritte, 1965]

Grouping by Completion – Kanisza Triangle





Grouping by Completion – More Examples

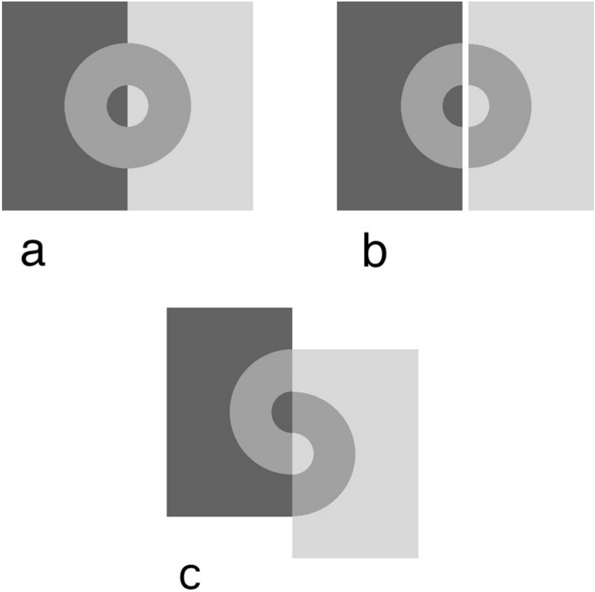


The Ultimate Challenge

The Ultimate Challenge

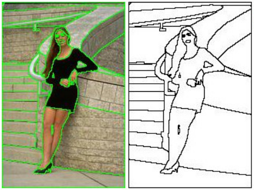
The Ultimate Challenge

Grouping Influences Lightness Perception

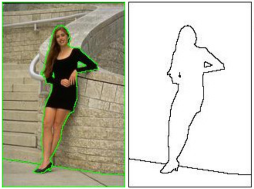


Annotation Ambiguity

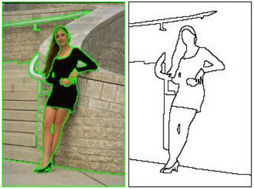
Annotation Ambiguity



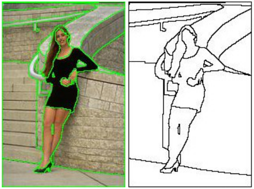
User #1105
61 Segments



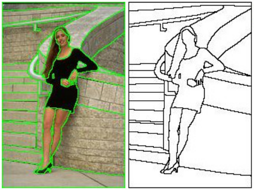
User #1109
6 Segments



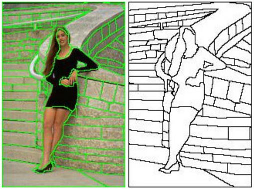
User #1114
19 Segments



User #1115
35 Segments



User #1123
54 Segments



User #1130
110 Segments

Conclusions so far

- Segmentation is generally quite difficult
- Often it is even hard to characterize what it is
- We humans are very good at it
- Thus it must be important for our visual processing system
- On a computer we can only implement some of the simple cues
- What we talk about today will not solve all of these examples
- But we can solve simpler instances which is still useful!

Unsupervised Segmentation

- We focus on the unsupervised setting first (*i.e.*, no image annotations available)
- Goal: Decompose an arbitrary image into coherent regions
- One way of doing this is by considering segmentation as a clustering problem:
  - Clustering algorithms try to group data points in some feature space together
  - First, identify each pixel with a feature vector which may include the pixel location, color as well as a texture descriptor
  - Cluster the pixels into regions using clustering algorithms (many machine learning toolboxes available!)

K-Means Clustering

- Let $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ denote a dataset, $\mathbf{x}_n \in \mathbb{R}^D$
- Let $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$ be a set of K cluster centers, $\boldsymbol{\mu}_k \in \mathbb{R}^D$
- Let $r_{n,k} = 1$ if $\mathbf{x}_n$ is assigned to cluster k, and $r_{n,k} = 0$ otherwise
- We want to minimize the following objective function:

$$E(\boldsymbol{\mu}, \mathbf{r}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

- Intuitively, we want to minimize the distance of each data point (*i.e.*, pixel) to the cluster it is assigned to by simultaneously manipulating the cluster centers **and** the assignments!
- As $r_{n,k}$ is discrete and μ_k is continuous, joint optimization is difficult, but we can formulate an alternation scheme where we update each of these two types of variables at a time while keeping the other fixed

K-Means Clustering

- Let $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ denote a dataset, $\mathbf{x}_n \in \mathbb{R}^D$
- Let $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K\}$ be a set of K cluster centers, $\boldsymbol{\mu}_k \in \mathbb{R}^D$
- Let $r_{n,k} = 1$ if $\mathbf{x}_n$ is assigned to cluster k, and $r_{n,k} = 0$ otherwise
- We want to minimize the following objective function:

$$E(\boldsymbol{\mu}, \mathbf{r}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

- 1. step: Pick K and initialize $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K\}$ randomly
- 2. step: Minimize $E(\boldsymbol{\mu}, \mathbf{r})$ wrt. $\mathbf{r}_n = \{r_{n,1}, \dots, r_{n,K}\}$ $(\forall n)$:

$$\mathbf{r}_n^* = \underset{\mathbf{r}_n}{\operatorname{argmin}}\, E(\boldsymbol{\mu}, \mathbf{r}) = \underset{\mathbf{r}_n}{\operatorname{argmin}} \sum_{k=1}^{K} r_{n,k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 = ?$$

$$\Rightarrow r_{n,k} = \left[k = \underset{j}{\operatorname{argmin}}\, \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \right]$$

K-Means Clustering

- Let $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ denote a dataset, $\mathbf{x}_n \in \mathbb{R}^D$
- Let $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K\}$ be a set of K cluster centers, $\boldsymbol{\mu}_k \in \mathbb{R}^D$
- Let $r_{n,k} = 1$ if $\mathbf{x}_n$ is assigned to cluster k, and $r_{n,k} = 0$ otherwise
- We want to minimize the following objective function:

$$E(\boldsymbol{\mu}, \mathbf{r}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

- 3. step: Minimize $E(\boldsymbol{\mu}, \mathbf{r})$ wrt. $\boldsymbol{\mu}_k$ ($\forall k$):

$$\boldsymbol{\mu}_k^* = \operatorname*{argmin}_{\boldsymbol{\mu}_k} E(\boldsymbol{\mu}, \mathbf{r}) = \operatorname*{argmin}_{\boldsymbol{\mu}_k} \sum_{n=1}^{N} r_{n,k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 = ?$$

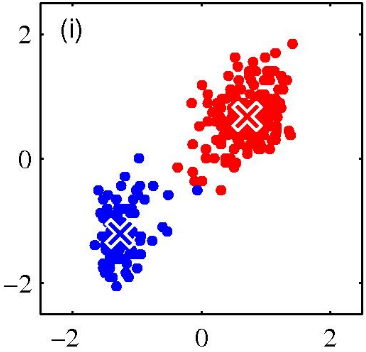
$$\Rightarrow \sum_{n=1}^{N} r_{n,k} \left(\mathbf{x}_n - \boldsymbol{\mu}_k\right) = 0 \Rightarrow \boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{n,k} \mathbf{x}_n}{\sum_{n=1}^{N} r_{n,k}}$$

- Repeat until convergence! (guaranteed to converge)

K-Means Clustering

Illustration using the Old Faithful dataset (2 clusters):

1. Select number of clusters K
2. Randomly initialize centers
3. Assign each point to closest center
4. Update center to centroid of assigned points
5. Go to step 3. and repeat until convergence



Questions:

- What is the right number of clusters K?
- How can this be applied to the task of image segmentation?

Simple Linear Iterative Clustering (SLIC)

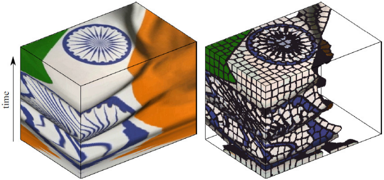
SLIC: Technique for generating "superpixels" [Achanta *et al.*, 2011]

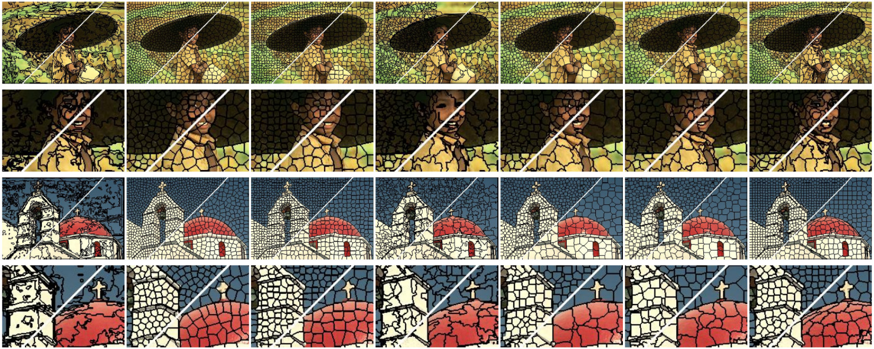
- Let $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be the features of all N pixels in the image
- Define $\mathbf{x}_n = (\mathbf{x}_n^{col}, \mathbf{x}_n^{pos})$, where
  - $\mathbf{x}_n^{col} = (l_n, a_n, b_n)$ represents the pixel color in LAB color space
  - $\mathbf{x}_n^{pos} = (x_n, y_n)$ denotes the pixel location in the image
- Initialize $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$ by sampling the cluster centers using a regular grid in the image (intuition: should lead to regular superpixels)
- Minimize the following objective function:

$$E(\boldsymbol{\mu}, \mathbf{r}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} \left(\|\mathbf{x}_n^{col} - \boldsymbol{\mu}_k^{col}\|^2 + \lambda \|\mathbf{x}_n^{pos} - \boldsymbol{\mu}_k^{pos}\|^2 \right)$$

- Large weights (λ) will lead to very regular superpixels, while small weights will emphasize color consistency

Simple Linear Iterative Clustering (SLIC)

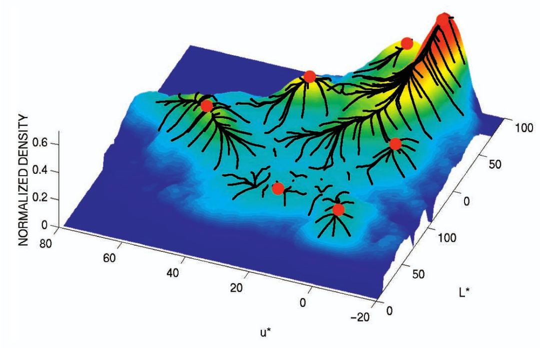




GS04 NC05 TP09 QS09 GCa10 GCb10 SLIC

Mean Shift Clustering

- Mean shift is a method for finding modes in a cloud of data points (*i.e.*, finding the location where the data points are most dense)
- The black lines indicate various search paths obtained by starting at different points and following the gradient of the point density
- For segmentation we use pixel feature vectors, start one path per pixel and assign points to the same class if they converge to the same mode

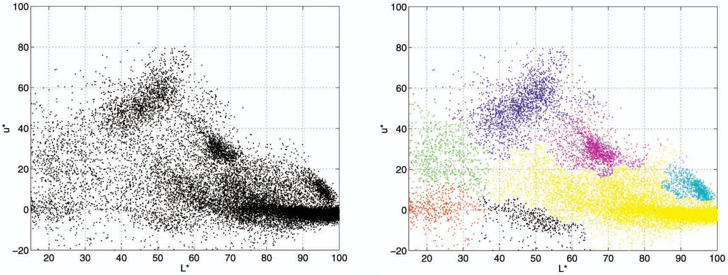


Mean Shift Clustering

- The density can be estimated using kernel density estimation (KDE):

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^D} \sum_{n=1}^{N} k\left(\left\|\frac{\mathbf{x} - \mathbf{x}_n}{h}\right\|\right)$$

  - Here, $k(\cdot)$ is a kernel with width h, *e.g.*: $k(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right)$
  - This is a so-called non-parametric density estimate
- The mean shift procedure is obtained by following the gradient of $\hat{f}(\mathbf{x})$, starting at each data point $\mathbf{x}_n$ $(\forall n \in \{1, \ldots, N\})$:



Mean Shift Clustering

Algorithm (for every image pixel do):

- Compute the mean shift vector:

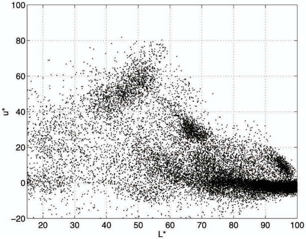
$$\mathbf{m}(\mathbf{x}) = \frac{\sum_{n=1}^{N} \mathbf{x}_n \frac{\partial}{\partial \mathbf{x}} k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_n}{h}\right\|\right)}{\sum_{n=1}^{N} \frac{\partial}{\partial \mathbf{x}} k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_n}{h}\right\|\right)} - \mathbf{x}$$

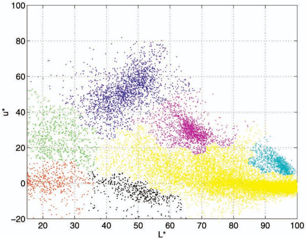
- This vector points into the direction of maximum increase in density
- Now, lets move the point by the mean shift vector:

$$\mathbf{x} \leftarrow \mathbf{x} + \mathbf{m}(\mathbf{x})$$

- ... and repeat until convergence
- More details can be found in [Comaniciu & Meer, 2002]

Mean Shift Clustering





Mean Shift Clustering

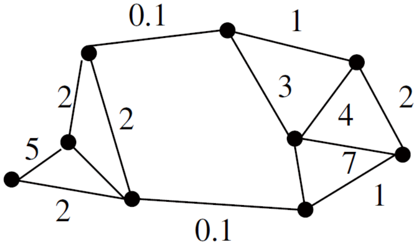
Comments about mean shift:

- We have noticed that we didn't need to specify the number of segments K as for the K-means clustering algorithm
- Thus, do we no longer have to choose this number by hand?
- Yes and no: We don't choose it directly, but we need to specify the kernel as well as the kernel bandwidth
- The number of segments depends on the kernel width
- Thus, we have just shifted the problem to a different place
- But: Mean shift does not depend on the initialization!

Spectral Clustering

There is another way of looking at the problem:

- Consider a weighted graph $G = (V, E)$ with vertices V and edges E
- Each node $u \in V$ corresponds to a pixel
- Each edge $(u, v) \in E$ connects two nodes (*i.e.*, pixels) and is assigned a weight $w(u, v) \geq 0$ which determines the similarity of node u and v (*i.e.*, a large $w(u, v)$ means that u and v are likely to cluster together)
- Goal: Find a partition $V = V_1 \cup \cdots \cup V_K$ such that the similarity within each V_i is high, and across any V_i, V_j is low
- Intuition: We want to cut the graph at low-affinity edges!



Spectral Clustering

Similarity criteria for edge weights $w(u, v)$:

- Similarity by distance ($\mathbf{x}(u)$ = location of pixel u):

$$w(u, v) = \exp\left\{-\frac{\|\mathbf{x}(u) - \mathbf{x}(v)\|^2}{2\sigma^2}\right\}$$

- Similarity by intensity ($I(u)$ = intensity at pixel u):

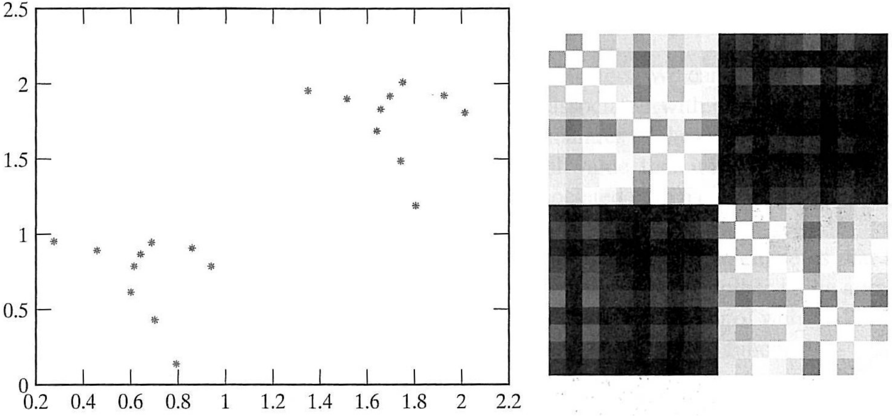
$$w(u, v) = \exp\left\{-\frac{\|I(u) - I(v)\|^2}{2\sigma^2}\right\}$$

- Similarity by texture/color ($\mathbf{f}(u)$ = feature vector at pixel u):

$$w(u, v) = \exp\left\{-\frac{\|\mathbf{f}(u) - \mathbf{f}(v)\|^2}{2\sigma^2}\right\}$$

Spectral Clustering

Example of a similarity matrix $W_{u,v} = w(u, v)$ for a 2D point set:



Spectral Clustering

A simple approach:

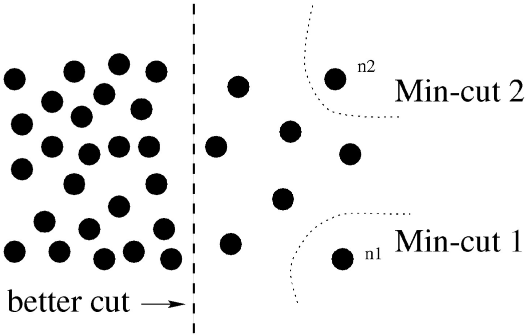
- Assume the vertex set V can be partitioned into two sets, A and B, by simply removing edges connecting the two parts (of course, this implies $A \cap B = \emptyset$ and $A \cup B = V$)
- A simple measure of dissimilarity is the total weight of edges connecting A and B (*i.e.*, all edges which have been removed):

$$cut(A, B) = \sum_{a \in A, b \in B} w(a, b)$$

- The optimal bipartitioning of the graph is the one that minimizes this cut value such that $|A| \geq 1$ and $|B| \geq 1$
- Optimal polynomial time algorithm exist for solving this problem! (max-flow min-cut theorem)

Spectral Clustering

What is the problem? Assume the edge weights are inversely proportional to the distance between nodes ...



- As the cut measure increases with the number of edges going across the two partitioned parts, the min cut solution favors cutting small sets of isolated nodes!

Normalized Cuts [Shi & Malik, 2000]

- To avoid this unnatural bias for small partitions, Shi & Malik propose to minimize the following criterion (wrt. the partitioning) instead

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

where $V = A \cup B$ and

$$cut(A, B) = \sum_{a \in A, b \in B} w(a, b) \quad \text{and} \quad assoc(A, B) = \sum_{a \in A, v \in V} w(a, v)$$

- Instead of looking at the value of total edge weight between A and B, this measure computes the cut cost as a fraction with respect to the total edge connections to all the nodes in the graph!
- Unfortunately, this is a NP-hard problem! [Papadimitriou, 1997]

Normalized Cuts [Shi & Malik, 2000]

The equation from the previous slide reads as:

$$Ncut(A, B) = \frac{\sum_{a \in A, b \in B} w(a, b)}{\sum_{a \in A, v \in V} w(a, v)} + \frac{\sum_{a \in A, b \in B} w(a, b)}{\sum_{b \in B, v \in V} w(b, v)}$$

- Let $\mathbf{x} \in \{-1, +1\}^{|V|}$ be a $|V|$-dimensional indicator vector (*i.e.*, for $a \in A$: $x_a = +1$, and for $b \in B$: $x_b = -1$)
- Let $\mathbf{D}$ be a diagonal matrix with $d_u = \sum_{v \in V} w(u, v)$ on its diagonal
- Let $\mathbf{W}$ be the similarity weight matrix with $W_{u,v} = w(u, v)$
- Then, the equation above can then be rewritten as

$$Ncut(A, B) = \frac{\sum_{x_a > 0, x_b < 0} -W_{a,b} x_a x_b}{\sum_{x_a > 0} d_a} + \frac{\sum_{x_a > 0, x_b < 0} -W_{a,b} x_a x_b}{\sum_{x_b < 0} d_b}$$

- Or in matrix notation (using $k = \sum_{x_a > 0} d_a / \sum_u d_u$):

$$4 \cdot Ncut(\mathbf{x}) = \frac{(\mathbf{1} + \mathbf{x})^T (\mathbf{D} - \mathbf{W})(\mathbf{1} + \mathbf{x})}{k \mathbf{1}^T \mathbf{D} \mathbf{1}} + \frac{(\mathbf{1} - \mathbf{x})^T (\mathbf{D} - \mathbf{W})(\mathbf{1} - \mathbf{x})}{(1 - k) \mathbf{1}^T \mathbf{D} \mathbf{1}}$$

Normalized Cuts [Shi & Malik, 2000]

- In their paper, Shi & Malik show how the $Ncut(A, B)$ problem can be transformed to minimizing (with respect to $\mathbf{y}$):

$$\frac{\mathbf{y}^T(\mathbf{D}-\mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}}$$

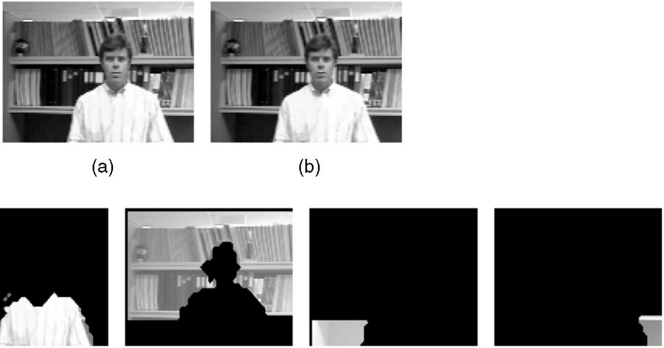
  where $\mathbf{y} = (\mathbf{1}+\mathbf{x}) - \frac{k}{1-k}(\mathbf{1}-\mathbf{x})$
- To solve this minimization, the discrete vector $\mathbf{y}$ is relaxed to take on real values $\mathbf{y} \in \mathbb{R}^{|V|}$, leading to the generalized eigenvalue problem:

$$(\mathbf{D}-\mathbf{W})\mathbf{y} = \lambda\mathbf{D}\mathbf{y}$$

- The smallest eigenvalue is zero yielding the trivial solution with eigenvector $\mathbf{y} = \mathbf{1}$ (due to a constraint) $\rightarrow$ we are interested in the eigenvector $\mathbf{y}$ corresponding to the second smallest eigenvalue!
- As the eigensystem is large, the procedure is typically relatively slow

Normalized Cuts [Shi & Malik, 2000]

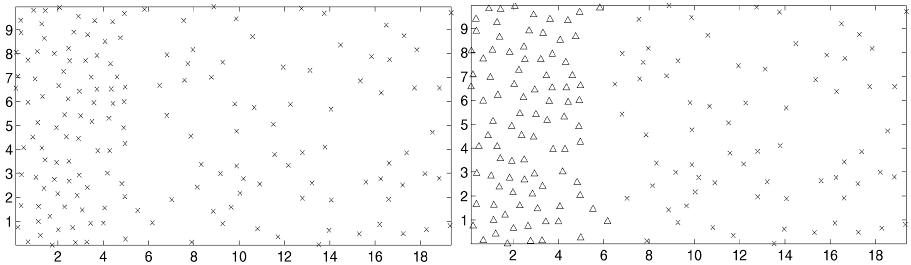
- For more than two partitions ($K > 2$):
  - Apply this procedure recursively (repartitioning) – or –
  - Apply k-means clustering in space of eigenvectors
- For large images: approximations required (sparse matrix $\mathbf{W}$)
- Some results (using distance, intensity & texture features):



(a) (b)

(c) (d) (e) (f) (g)

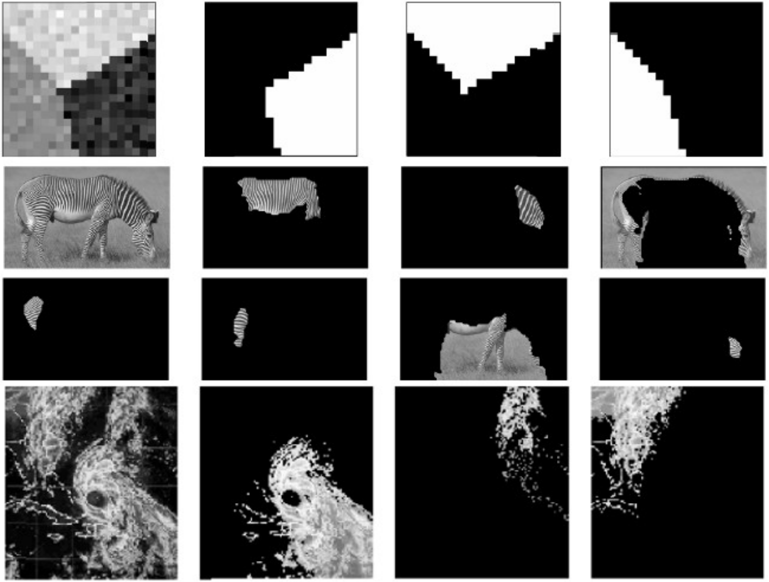
Normalized Cuts [Shi & Malik, 2000]



- Left: Points generated by 2 Poisson processes (intensity: 2.5 and 1.0)
- Right: Partition of the point set using normalized cuts
- Similarity by distance was used here:

$$w(u,v) = \exp\left\{-\frac{\|\mathbf{x}(u)-\mathbf{x}(v)\|^2}{2\sigma^2}\right\}$$

Normalized Cuts [Shi & Malik, 2000]



Normalized Cuts [Shi & Malik, 2000]

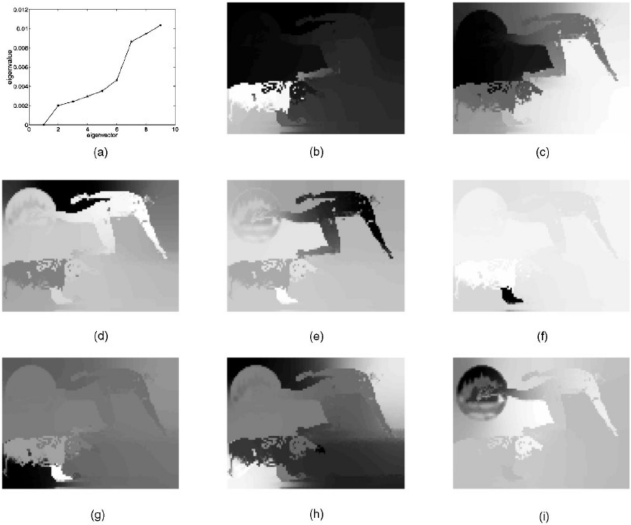
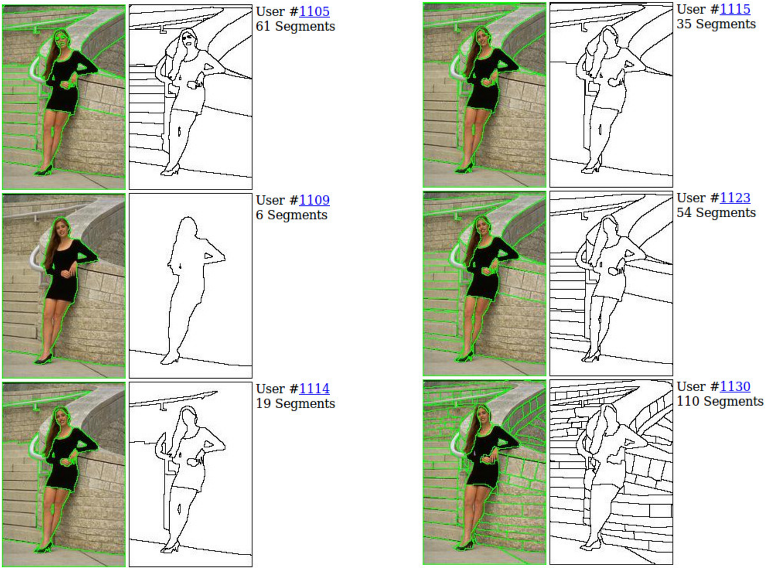


Fig. 3. Subplot (a) plots the smallest eigenvectors of the generalized eigenvalue system (11). Subplots (b)-(i) show the eigenvectors corresponding the second smallest to the ninth smallest eigenvalues of the system. The eigenvectors are reshaped to be the size of the image.

Is there a correct segmentation?



Is there a correct segmentation?

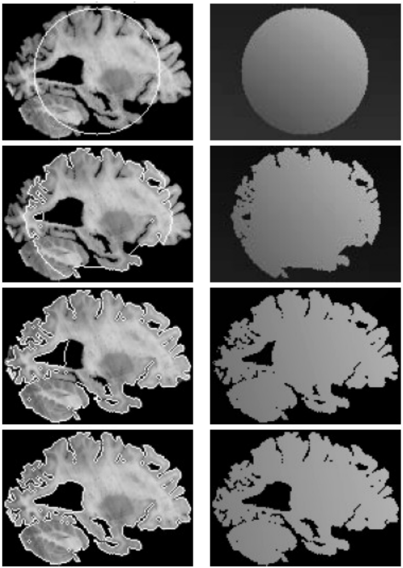
Is there a correct segmentation?

- To answer this question we need to think about the purpose
- For a superpixel segmentation we might want to expect each superpixel to correspond to a smooth surface in 3D, thus the precise form of the segmentation is irrelevant
- But, we might also be interested in separating one (or several) foreground object(s) from the background (or from each other)
- If the object identity is clear, this makes the task well-defined
- Segmentation should be coupled to the task we want to solve with it

Now:

- Figure-ground segmentation: Segmenting one foreground object from the background
- Interactive: We will help the algorithm to identify the foreground!

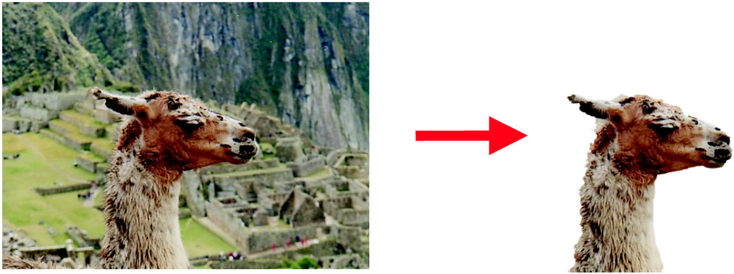
Applications



Applications

Photomontage

- To assemble several photos into a montage we need to separate the object of interest from the background:



- How does the algorithm know what I am interested in?

Photomontage

- Basic idea: let the user annotate some examples of foreground & background:

input image

user annotation

Interactive Segmentation

- A per-pixel classifier trained on the annotation will lead to noisy results if it only considers a local neighborhood in the image and is applied to each pixel separately (exception: deep neural networks)
- But objects in the world tend to be compact and smooth
- We can thus formulate the (interactive) segmentation problem as a discrete MRF to incorporate these smoothness assumptions ...
- ... and apply the inference techniques we have learned about!

Interactive Segmentation

Let us specify interactive segmentation as a discrete MRF:

- Let $\mathbf{I} \in \{0, \ldots, 255\}^{M \times N \times 3}$ denote the RGB image of size $M \times N$
- Let $\mathbf{S} \in \{0, 1\}^{M \times N}$ denote the desired binary segmentation
- Specify a MRF in terms of its Gibbs energy $p(\mathbf{S}) \propto \exp\{-E(\mathbf{S})\}$

$$E(\mathbf{S}) = \sum_i \psi_{data}(s_i) + \lambda \sum_{i \sim j} \psi_{smooth}(s_i, s_j)$$

with smoothness weight λ and $i \sim j$ indicating neighboring pixels.

Interactive Segmentation

$$E(\mathbf{S}) = \sum_i \psi_{data}(s_i) + \lambda \sum_{i \sim j} \psi_{smooth}(s_i, s_j)$$

Data term:

- Prefer pixels that look similar to labeled foreground pixels (scribbles) to be labeled as foreground, and vice versa for the background
- Assume i.i.d. likelihood
- Simplest approach: Color log-likelihood ($\mathbf{c}_i$: color at pixel i)

$$\psi_{data}(s_i) = \begin{cases} \infty & \text{if } i \in \mathcal{A} \wedge s_i \neq a_i \\ -\log p_{s_i}(\mathbf{c}_i) & \text{otherwise} \end{cases}$$

where $p_0(\cdot)$ and $p_1(\cdot)$ are the background and foreground color distribution estimated from the user scribbles for image $\mathbf{I}$, $\mathcal{A}$ denotes the set of annotated pixels and a_i the annotation (hard constraint)

Interactive Segmentation

There are several ways to represent/estimate $p_0(\cdot)$ and $p_1(\cdot)$:

- We can discretize the (RGB or LAB) color space, calculate a histogram and normalize it to obtain a step-wise approximation
- We can directly work in the continuous space and use non-parametric kernel density estimation (KDE)
- We can fit a Gaussian to the data (parametric approach)

$$p(\mathbf{c}) = \mathcal{N}(\mathbf{c}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

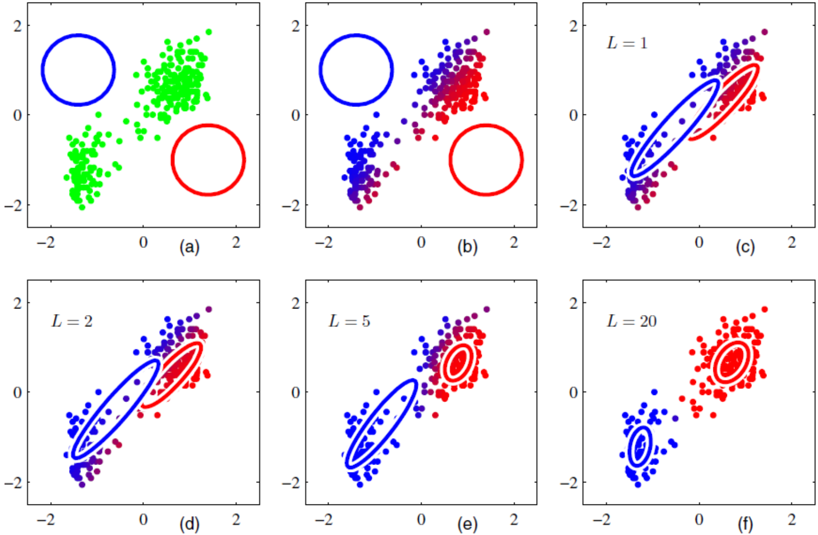
- We can fit a Gaussian mixture model to the data

$$p(\mathbf{c}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{c}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Gaussian mixtures can represent any probability distribution with arbitrary precision by increasing the number of mixture components

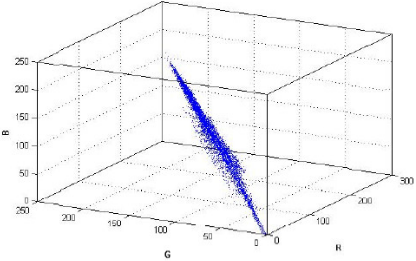
Interactive Segmentation

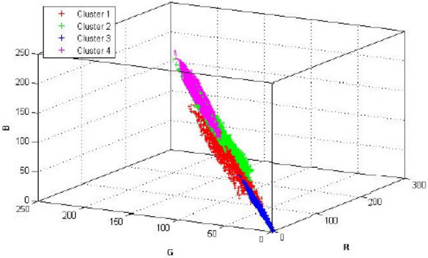
Estimating the parameters of a Gaussian mixture using the EM algorithm:

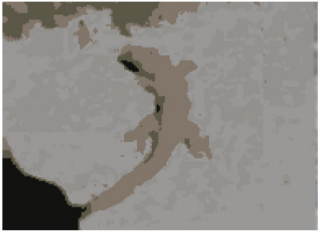


Interactive Segmentation

Estimating the parameters of a Gaussian mixture using the EM algorithm:







Interactive Segmentation

$$E(\mathbf{S}) = \sum_i \psi_{data}(s_i) + \lambda \sum_{i \sim j} \psi_{smooth}(s_i, s_j)$$

Smoothness term:

- Penalize label changes
- Simplest approach: Potts model

$$\psi_{smooth}(s_i, s_j) = [s_i \neq s_j]$$

  where $[\cdot]$ denotes the Iverson bracket
- Problem: The smoothness term is agnostic to the image! (segment boundaries are not encourage to coincide with image edges)
- Better: contrast-sensitive Potts model (I_i: intensity at pixel i):

$$\psi_{smooth}(s_i, s_j; \mathbf{I}) = \exp\left\{-\beta(I_i - I_j)^2\right\}[s_i \neq s_j]$$

- Intuition: downweight smoothness penalty at image edges/gradients

Interactive Segmentation

Did you notice something?

- We made $\psi_{smooth}(s_i, s_j; \mathbf{I})$ dependent on the image!
- Before, only the data term was image dependent, but we omitted this in our (sloppy) notation. Correctly, we should have written:

$$E(\mathbf{S}; \mathbf{I}) = \sum_i \psi_{data}(s_i; \mathbf{I}) + \lambda \sum_{i \sim j} \psi_{smooth}(s_i, s_j) \tag{1}$$

- Now, also the prior term depends on the image:

$$E(\mathbf{S}; \mathbf{I}) = \sum_i \psi_{data}(s_i; \mathbf{I}) + \lambda \sum_{i \sim j} \psi_{smooth}(s_i, s_j; \mathbf{I}) \tag{2}$$

- This is a particular instance of a conditional random field (CRF)
- While (1) can be interpreted via Bayes rule $p(\mathbf{S}|\mathbf{I}) = \frac{p(\mathbf{I}|\mathbf{S})p(\mathbf{S})}{p(\mathbf{I})}$, in (2) we directly model the posterior $p(\mathbf{S}|\mathbf{I})$ (no generative interpretation)

Interactive Segmentation

$$E(\mathbf{S};\mathbf{I}) = \sum_i \psi_{data}(s_i;\mathbf{I}) + \lambda \sum_{i\sim j} \psi_{smooth}(s_i, s_j;\mathbf{I})$$

- We can pick our favorite inference technique for graphical models to solve this equation
- Popular choices include belief propagation and variational methods
- For binary problems which are submodular, the **global optimum** can be obtained in polynomial time using graph cuts!

Submodularity

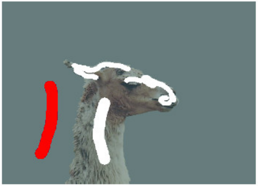
$\psi(s_i, s_j)$ is submodular if:

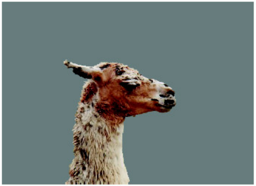
$$\psi(0,1) + \psi(1,0) \geq \psi(0,0) + \psi(1,1)$$

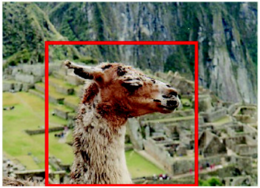
- Readings: [Boykov & Jolly, 2001], [Kolmogorov & Zabih, 2004]

Interactive Segmentation

- Interactive Graph cut requires a lot of user interaction
- In particular in textured and ambiguous areas
- Better: Simply specify a bounding box around the object of interest!



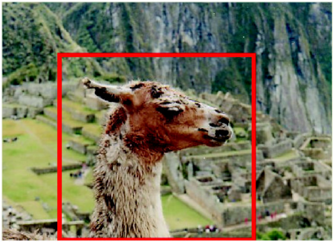




Interactive Segmentation

First attempt:

- Create color model inside and outside the bounding box to define the foreground and background likelihood:



Annotation

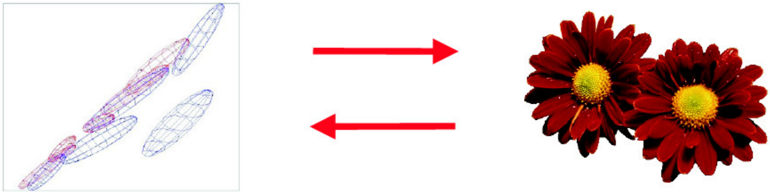
Result

- Problem: The foreground region contains a lot of background!
- This leads to inaccurate results

Grab Cut

Solution proposed in [Rother et al., 2004]:

- Iterate between
  - Determining the histograms from current foreground and background
  - Segmenting the image with the current likelihood
- An additional border matting post-processing step is introduced
- This technique is called “Grab Cut”
- It is implemented in MS Office 2010 as “Background removal tool”



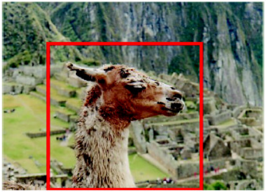
Gaussian mixture model of FG/BG

Graph cut segmentation

Grab Cut

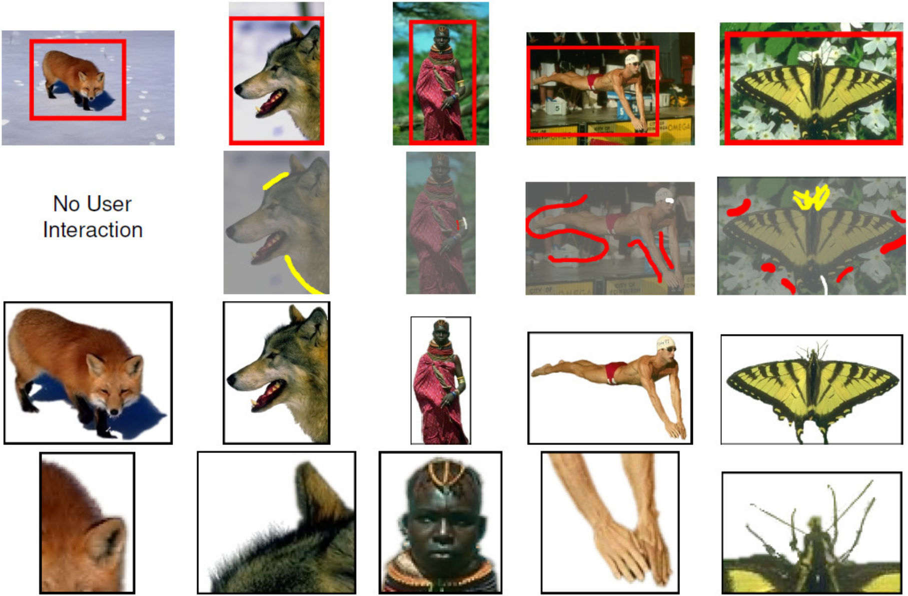
Solution proposed in [Rother et al., 2004]:

- Iterate between
  - Determining the histograms from current foreground and background
  - Segmenting the image with the current likelihood
- An additional border matting post-processing step is introduced
- This technique is called “Grab Cut”
- It is implemented in MS Office 2010 as “Background removal tool”



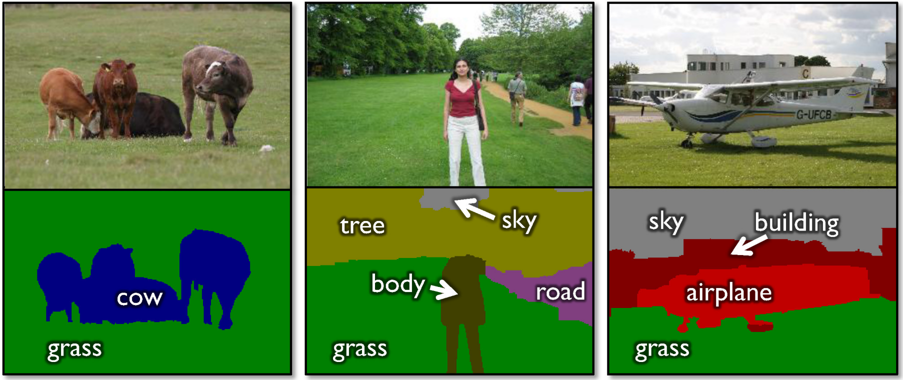
Progressing iterations...

Grab Cut Results



Semantic Segmentation

- So far, we have looked at generic unsupervised segmentation as well as foreground-background segmentation
- We can also couple segmentation with recognition!
- Goal: Segment the image and determine category at every pixel
- Note: No information about object instances is recovered (just class)



TextonBoost

TextonBoost [Shotton et al., 2006]

- Idea: Learn a per-pixel semantic classifier on a set of training images and incorporate it as a unary into a CRF
- Let $\mathbf{I} \in [0, \ldots, 255]^{M \times N \times 3}$ denote the image
- Let $\mathbf{S} \in [1, \ldots, K]^{M \times N}$ denote the desired output (K classes)
- The Gibbs energy is defined as

$$E(\mathbf{S}) = \sum_i \underbrace{\pi(s_i; \mathbf{I})}_{color} + \underbrace{\lambda(s_i, i)}_{location} + \underbrace{\psi_i(s_i; \mathbf{I})}_{appearance} + \sum_{i \sim j} \underbrace{\phi(s_i, s_j; \mathbf{I})}_{smoothness}$$

where $i \sim j$ denotes adjacent pixels

TextonBoost Model

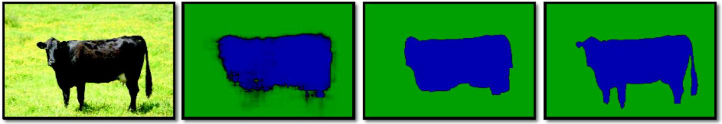
$$E(\mathbf{S}) = \sum_i \underbrace{\pi(s_i;\mathbf{I})}_{color} + \underbrace{\lambda(s_i,i)}_{location} + \underbrace{\psi_i(s_i;\mathbf{I})}_{appearance} + \sum_{i\sim j} \underbrace{\phi(s_i,s_j;\mathbf{I})}_{smoothness}$$

Color term:

- Capture the color distribution of the instances of a class for a particular image
- Gaussian Mixture model in RGB color space

$$\pi(s_i;\mathbf{I}) = -\log \sum_k P(k|s_i)\mathcal{N}(c_i|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$$

- Shared mixture components between different classes
- Estimated based on initial labeling of test image



TextonBoost Model

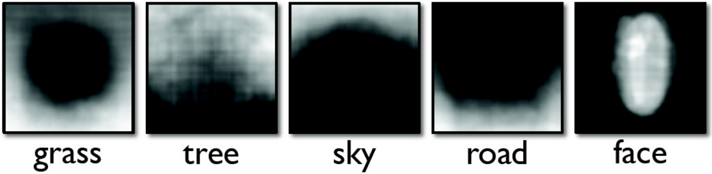
$$E(\mathbf{S}) = \sum_i \underbrace{\pi(s_i; \mathbf{I})}_{\text{color}} + \underbrace{\lambda(s_i, i)}_{\text{location}} + \underbrace{\psi_i(s_i; \mathbf{I})}_{\text{appearance}} + \sum_{i \sim j} \underbrace{\phi(s_i, s_j; \mathbf{I})}_{\text{smoothness}}$$

Location term:

- Capture the weak dependence of the class label on the absolute location of the pixel in the image
- Gaussian Mixture model in RGB color space

$$\lambda(s_i; i) = -\log P(i|s_i)$$

- Basically counts the frequency of a class label at a pixel



TextonBoost Model

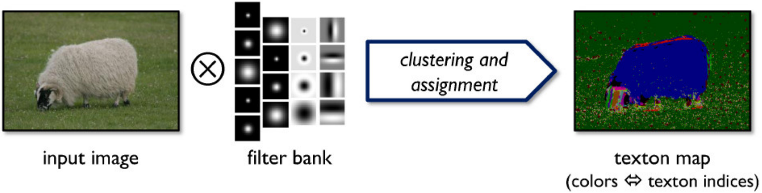
$$E(\mathbf{S}) = \sum_i \underbrace{\pi(s_i; \mathbf{I})}_{\text{color}} + \underbrace{\lambda(s_i, i)}_{\text{location}} + \underbrace{\psi_i(s_i; \mathbf{I})}_{\text{appearance}} + \sum_{i \sim j} \underbrace{\phi(s_i, s_j; \mathbf{I})}_{\text{smoothness}}$$

Appearance term:

- Use features selected by boosting to represent the shape, texture and appearance context of the object classes. Use classifier directly:

$$\psi_i(s_i; \mathbf{I}) = -\log P_i(s_i | \mathbf{I})$$

- Textons: Clustered filter bank responses
- Features: Sum of textons in (one of many) random rectangles



TextonBoost Model

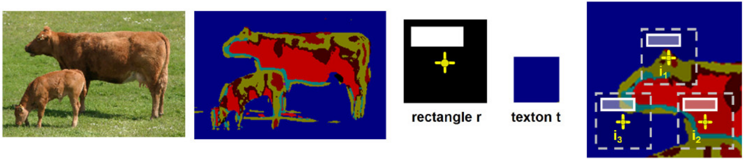
$$E(\mathbf{S}) = \sum_i \underbrace{\pi(s_i;\mathbf{I})}_{\text{color}} + \underbrace{\lambda(s_i, i)}_{\text{location}} + \underbrace{\psi_i(s_i;\mathbf{I})}_{\text{appearance}} + \sum_{i\sim j} \underbrace{\phi(s_i, s_j;\mathbf{I})}_{\text{smoothness}}$$

Appearance term:

- Use features selected by boosting to represent the shape, texture and appearance context of the object classes. Use classifier directly:

$$\psi_i(s_i;\mathbf{I}) = -\log P_i(s_i|\mathbf{I})$$

- Textons: Clustered filter bank responses
- Features: Sum of textons in (one of many) random rectangles



(a) Input image (b) Texton map (c) Feature pair = (r,t) (d) Superimposed rectangles

TextonBoost Model

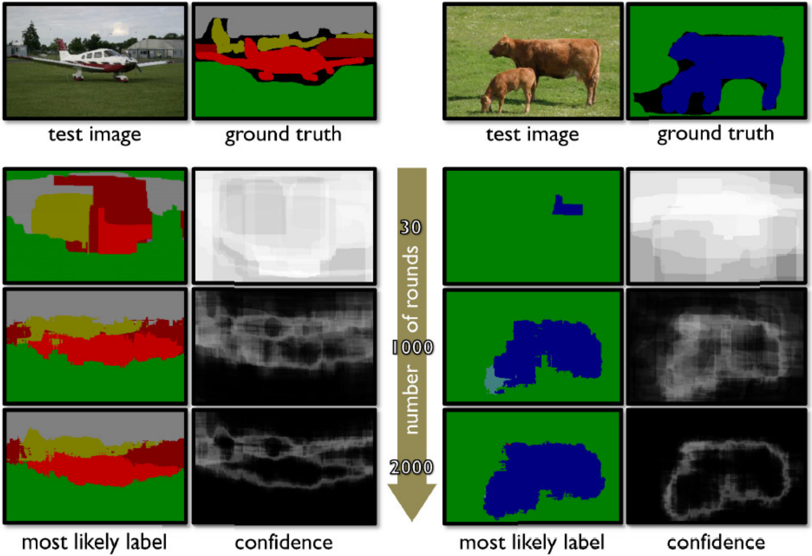
$$E(\mathbf{S}) = \sum_i \underbrace{\pi(s_i;\mathbf{I})}_{color} + \underbrace{\lambda(s_i,i)}_{location} + \underbrace{\psi_i(s_i;\mathbf{I})}_{appearance} + \sum_{i\sim j} \underbrace{\phi(s_i,s_j;\mathbf{I})}_{smoothness}$$

Smoothness term:

- Contrast-sensitive Potts model:

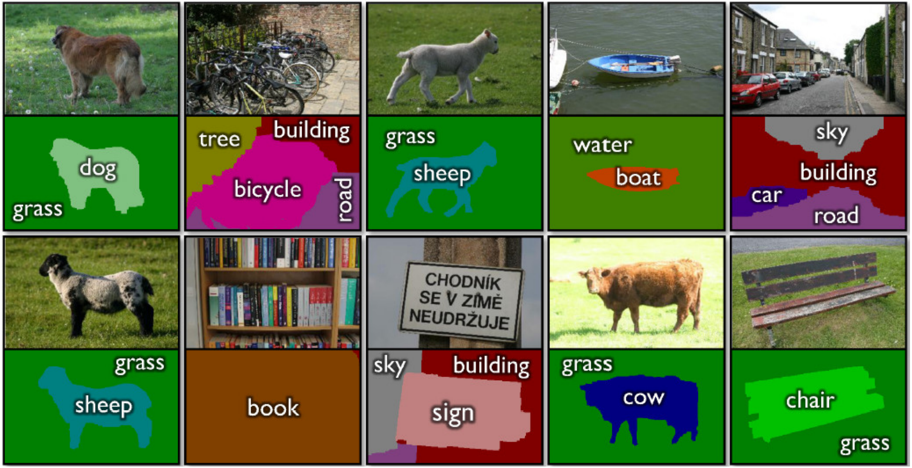
$$\phi(s_i,s_j;\mathbf{I}) = -(\alpha + \exp\{-\beta\|I_i - I_j\|^2\}) \cdot [s_i \neq s_j]$$

TextonBoost: Accuracy wrt. Boosting Rounds

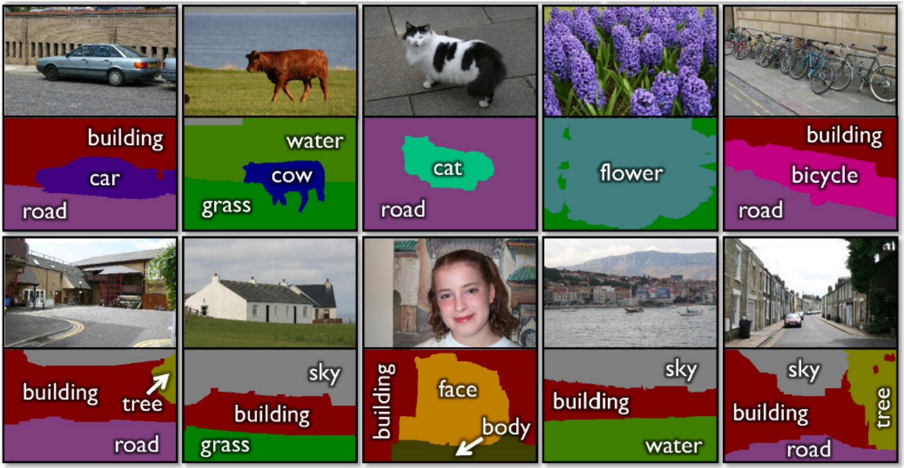


TextonBoost: MSRC Dataset

TextonBoost: Results



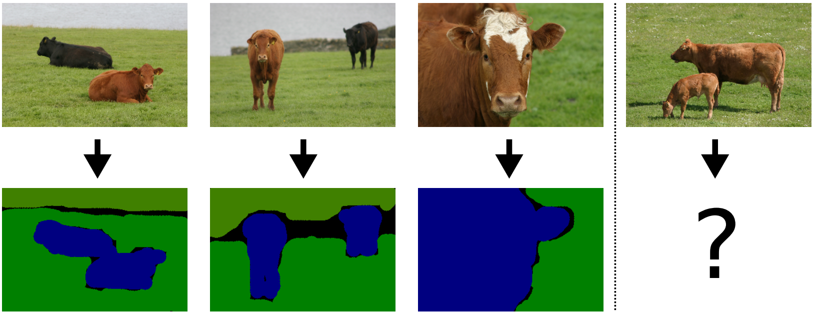
TextonBoost: Results



Exercise: Segment Cow from Background

You are going to segment (brown) cows this week!

- … using a simple color model estimated from training cows
- … using the max-product belief propagation code from exercise 4



training images test image

- Merry Christmas and a Happy New Year!