Lecture 7: CS677

Sept 12, 2017

Review

- HW1 due today
- Exam 1, Oct 10, class period, closed book, closed notes
  - Other details to follow
- Previous class
  - Image filtering
  - Image segmentation intro
  - Watershed algorithm
- Today's objective
  - Superpixel algorithm: SLIC
  - Mean-shift algorithm
  - Graph-based methods
    - Normalized cut

Image Segmentation

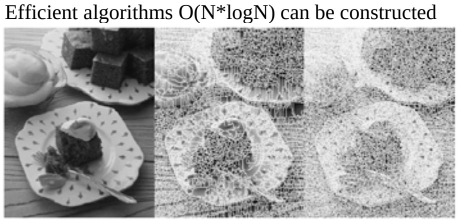
- Find boundaries of objects and surfaces in the scene
  - Typically characterized by discontinuities in range/depth of points in the scene (assumes object surfaces are continuous)
  - However, depth information is not readily available; instead we detect intensity (or color) discontinuities which may not always correspond to object boundaries
- Some examples on following slides

Superpixels

- Goal is to produce segments of similar size with high fidelity to boundaries
- Segments not likely to correspond to objects; objects to be detected in a subsequent stage
- May be useful for various forms of post-processing
- Has become a popular first step in recent years
- Watershed algorithm
- SLIC algorithm (not in book)

Watershed Algorithm

- Imagine intensity of image to represent terrain height
- Find “catchment basins” (lakes) that would form from rain fall
- Find local minima: consider them to be seeds and give a unique label to each
- For any pixel, go in direction of negative gradient, if a seed is reached (or a labeled pixel is reached), take its label.
- Efficient algorithms O(N*logN) can be constructed



On image intensity On gradient magnitude

SLIC Algorithm

- Simple Linear Iterative Clustering (SLIC)
- Construct seeds by uniform sampling of grid (at Step size S; must choose this size in advance)
- Assign each pixel label of its nearest neighbor in 5-D (x,y,L,a,b) space
  - Search only in a small spatial neighborhood (2Sx2S)
  - How to combine distance in image and color spaces?

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2}.$$

  - d_c is distance in color, d_s in image space
  - m controls trade-off between fidelity to boundaries and size of superpixels
- Algorithm is iterative: cluster centers are updated
- Efficient compared to k-means as search is in a limited neighborhood.
- Sizes of superpixels are similar (is this good?)

A SLIC Result

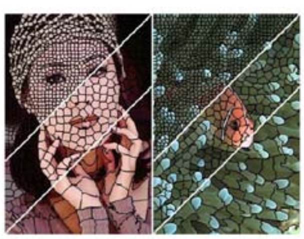


Fig. 1. Images segmented using SLIC into superpixels of size 64, 256, and 1,024 pixels (approximately).

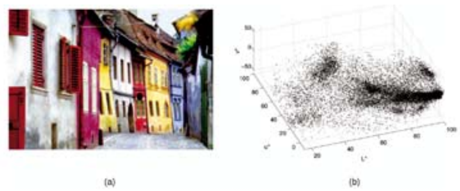
Ref: R. Achanta *et al*, “SLIC Superpixels Compared to State-of-the-Art Superpixel Methods”, IEEE Trans PAMI, Nov 2012, pp. 2274-2281 (posted on class webpage, not for distribution)

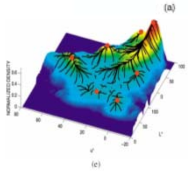
Mean Shift Filtering

- From work of Comaniciu and Meer (see RS 5.3.2, FP 9.3.4, 9.3.5)
- Filtering while preserving regions/edges
- May also be viewed as process of clustering by estimating the probability density function
  - Objective is similar to that of k-means clustering but we need not set the value of k in advance

Example

- Where are the clusters in figure (b), (L, u, v space)

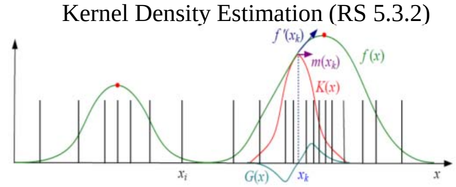




If we could estimate density function, finding peaks would be easy, and the number need not be fixed in advance.

Density distribution in L-u space

Kernel Density Estimation (RS 5.3.2)



- Data is given as "bumps" (vertical bars)
- **f(x)** is the estimate, **K(x)** is the *kernel* function, **G(x)** is derivative of K(x)

$$f(\boldsymbol{x}) = \sum_i K(\boldsymbol{x} - \boldsymbol{x}_i) = \sum_i k\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{h^2}\right)$$

where x_i are the input samples and k(r) is the kernel function; *h* is the width of the kernel

- Direct computation can be expensive, instead, compute only the maxima using the *mean-shift* method

Meanshift Vector (Derivation Optional)

- Compute gradient of *f*(*x*) (remember G(x) is derivative of K(x))

$$\nabla f(\boldsymbol{x}) = \sum_i (\boldsymbol{x}_i - \boldsymbol{x}) G(\boldsymbol{x} - \boldsymbol{x}_i) = \sum_i (\boldsymbol{x}_i - \boldsymbol{x}) g\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{h^2}\right), \quad (5.35)$$

where

$$g(r) = -k'(r), \quad (5.36)$$

and $k'(r)$ is the first derivative of $k(r)$. We can re-write the gradient of the density function as

$$\nabla f(\boldsymbol{x}) = \left[\sum_i G(\boldsymbol{x} - \boldsymbol{x}_i)\right] \boldsymbol{m}(\boldsymbol{x}), \quad (5.37)$$

where the vector

$$\boldsymbol{m}(\boldsymbol{x}) = \frac{\sum_i \boldsymbol{x}_i G(\boldsymbol{x} - \boldsymbol{x}_i)}{\sum_i G(\boldsymbol{x} - \boldsymbol{x}_i)} - \boldsymbol{x} \quad (5.38)$$

m(*x*) is called the *meanshift* vector: it is the difference between current value, *x*, and the weighted average of neighbor values around *x*.
Max of *f*(*x*) achieved by setting gradient of *f*(*x*) = 0

Meanshift Optimization

- Replace current estimate, y_k, of the mode at iteration *k*, with

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \boldsymbol{m}(\boldsymbol{y}_k) = \frac{\sum_i \boldsymbol{x}_i G(\boldsymbol{y}_k - \boldsymbol{x}_i)}{\sum_i G(\boldsymbol{y}_k - \boldsymbol{x}_i)}.$$

- It is shown that this process gives a local maximum of f(x) under reasonable conditions that k(r) is monotonically decreasing
- Two common kernels
  - Normal (Gaussian) kernel $k_N(r) = \exp\left(-\frac{1}{2}r\right)$
    - Gradient function is a Gaussian multiplied by *r*, see Figure.
  - Epanechnikov kernel $k_E(r) = \max(0, 1-r)$
    - Gradient function is a unit "ball" (constant)
- Combining spatial and spectral kernels

$$K(\boldsymbol{x}_j) = k\left(\frac{\|\boldsymbol{x}_r\|^2}{h_r^2}\right) k\left(\frac{\|\boldsymbol{x}_s\|^2}{h_s^2}\right)$$

x_r is the "range" domain (intensity or color),
x_s is the spatial domain (*x*, *y*)

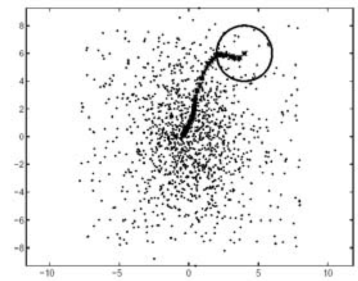
Mean Shift Filtering

- General idea
  - Choose a point
  - Weighted average over points in the neighborhood (weight depends on the *kernel* function) to compute a mean
    - Simple average for the Epanechnikov kernel
    - Note: point is in **combined** range and spatial dimensions
  - Shift center of neighborhood to new mean (hence *mean shift*)
  - Repeat until convergence
  - Replace the *range* (*e.g.* intensity or color) of original point with that of the convergent point
- Consider all points converging to the same maximum to correspond to the same cluster

13

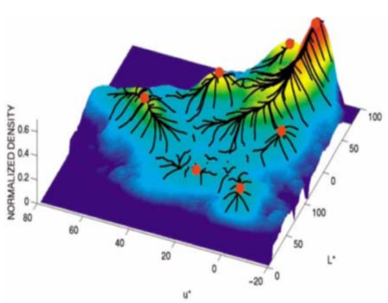
Example

- Binary function in 2-D



14

Result: Fig 5.16 RS



(e)

15

Mean Shift Filtering (FP Algorithm 9.6)

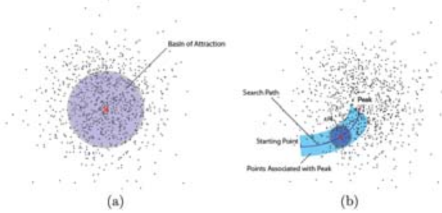
For each data point $\boldsymbol{x}_i$
  Apply the mean shift procedure (Algorithm 9.5), starting with $\boldsymbol{y}^{(0)} = \boldsymbol{x}_i$
  Record the resulting mode as $\boldsymbol{y}_i$

Cluster the $\boldsymbol{y}_i$, which should form small tight clusters.
A good choice is an agglomerative clusterer with group average distance, stopping clustering when the group average distance exceeds a small threshold

The data point $\boldsymbol{x}_i$ belongs to the cluster that its mode $\boldsymbol{y}_i$ belongs to.

16

Speed UP



(a) Associate all points within a neighborhood of peak with this peak (need not recompute for these points)

(b) Associate points within some distance of the path to the peak with this peak (need not recompute). Note this distance need not be of same size as the smoothing kernel and represents an approximation.

Efficient data structures used to speed up indexing of points

17

Example

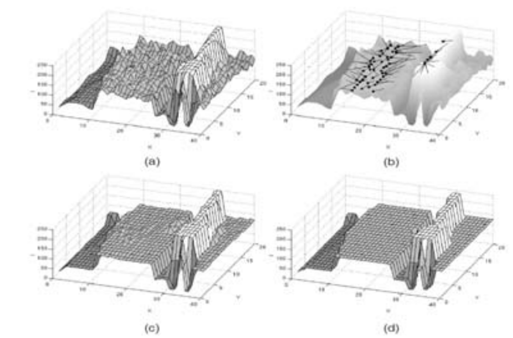


Fig. 4. Visualization of mean shift-based filtering and segmentation for gray-level data. (a) Input. (b) Mean shift paths for the pixels on the plateau and on the line. The black dots are the points of convergence. (c) Filtering result $(h_s, h_r) = (8, 4)$. (d) Segmentation result.

18

Image Example

(a) (b)

19

Mean Shift Segmentation (FP Algorithm 9.7)

For each pixel, p_i, compute a feature vector $\boldsymbol{x}_i = (\boldsymbol{x}_i^s, \boldsymbol{x}_i^r)$ representing spatial and appearance components, respectively.

Choose h_s, h_r the spatial (resp. appearance) scale of the smoothing kernel.

Cluster the $\boldsymbol{x}_i$ using this data and mean shift clustering (Algorithm 9.6).

(Optional) Merge clusters with fewer than t_{min} pixels with a neighbor; the choice of neighbor is not significant, because the cluster is tiny.

The i'th pixel belongs to the segment corresponding to its cluster center (for example, one could label the cluster centers $1 \ldots r$, and then identify segments by computing a map of the labels corresponding to pixels).

20

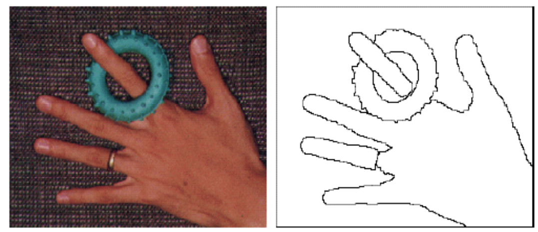
Cameraman Result

Figure 7: Segmentation with $(\sigma_s, \sigma_r, M) = (8, 4, 10)$ and reconstruction of the *cameraman* image after the elimination of regions representing sky and grass.

21

Color Image Result

Note: Filtering/clustering in five dimensional space



(a) (b)

Figure 9: *Hand* image. (a) Original. (b) Contours $(\sigma_s, \sigma_r, M) = (16, 19, 40)$.

22

Mean Shift: More Results, Natual Scenes

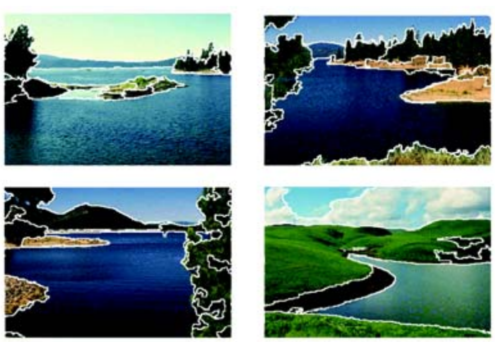
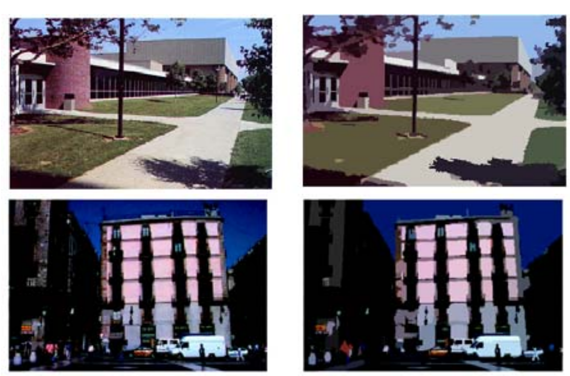


Fig. 10. *Landscape images*. All the region boundaries were delineated with $(h_s, h_r, M) = (8, 7, 100)$ and are drawn over the original image.

23

Mean Shift: More Results, buildings



24

Discussion

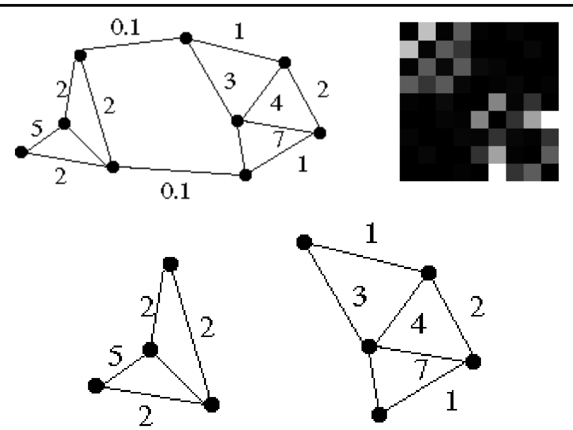
- Mean shift segmentation is computationally efficient (only a few seconds for 512x512 images)
  - With careful implementation and some approximations
- Dependence on three parameters (σ_s, σ_r, M)
  - Parameters are somewhat meaningful
  - May be set based on application
- Very impressive results in paper, but performance is not always this good
  - Basic limitations of segmentation by using color properties alone remain

25

Graph Based Methods

- Construct a graph from image
  - Each pixel (or superpixel) is a vertex
  - Edges between nodes, weight is large if nodes are similar
    - Based on differences in intensity, color, distance…
    - Defines an *affinity matrix* (rows and columns are indices of nodes)
- Cut this graph to get sub-graphs with strong interior links
  - Cut edges with low weights
- Example on next slide

26



27

Affinity Functions

Property	Affinity function	Notes
Distance	$\exp\left\{-\left((x-y)^t(x-y)/2\sigma_d^2\right)\right\}$	
Intensity	$\exp\left\{-\left((I(x)-I(y))^t(I(x)-I(y))/2\sigma_I^2\right)\right\}$	$I(x)$ is the intensity of the pixel at x.
Color	$\exp\left\{-\left(\mathrm{dist}(c(x),c(y))^2/2\sigma_c^2\right)\right\}$	$c(x)$ is the color of the pixel at x.
Texture	$\exp\left\{-\left((f(x)-f(y))^t(f(x)-f(y))/2\sigma_I^2\right)\right\}$	$f(x)$ is a vector of filter outputs describing the pixel at x computed as in Section 6.1.

28

Graph Algorithms

- Normalized cut (FP 9.4, RS 5.4)
- Given a graph V and two components A an B, define:
  - cut (A,B) = Sum of weights of edges in V connecting A and B
  - assoc (A, V) = Sum of weights of all edges that have at least one end in A (all edges out of nodes in A)
- Minimize following to achieve a *normalized cut*

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

  - Score is low if two components have few low weight edges between them and high weight edges internally
  - Minimization is an NP-hard problem
  - Approximate solutions by using “spectral graph” analysis techniques; FP and we omit details (may be found in RS book, section 5.4)

29

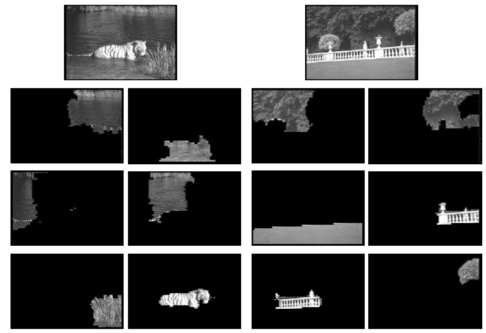


Figure from “Image and video segmentation: the normalized cut framework”, by Shi and Malik, copyright IEEE, 1998

30

Next Class

- Chapter 9, section 9.4
- Chapter 5, sections 5.1, 5.2

USC CS574: Computer Vision, Fall 2014

31