Lecture 8: CS677

Sept 14, 2017

Admin

- HW2 to be posted today, due Sept 24, 9AM
- Exam 1, Oct 10, class period
  - Closed book, closed notes
  - Topics: what we cover until Oct 5
- Make-up classes
  - Instructor needs to be absent Oct 24 and 26 (ICCV)
  - As our classes are recorded, we can make up by holding extra sessions on Oct 13 and Oct 20 (both Fridays)
    - Physical attendance is not required
  - Available slots are 8-9:50AM or 3:30-5:00 pm
    - Latter slot can accommodate only 24 students in class
    - Which do we prefer?

Review

- Previous class
  - Superpixel algorithm: SLIC
  - Mean-shift algorithm
  - Graph-based methods: Normalized cuts
  - Note: both mean-shift and normalized cut papers cited >10K times
- Today's objective
  - Two more graph-based algorithms
  - Edge and curve detection

FH Region Growing Method

- Agglomerative method, FP 9.4.2
- Algorithm does not have a memorable name, so referred to just by author names (Felzenswalb and Huttenlocher, we will call at F-H method).
- Construct a graph as in normalized cuts. Let $w(v_1,v_2)$ be weight of edges connecting vertices v_1 and v_2. Weight is high if pixels are *dissimilar* (opposite of the convention in normalized cuts)
- Start from small clusters, merge as appropriate
- Informally, the idea is to merge clusters if weight of best edge connecting them is less than the internal differences of the two clusters.
- Formal definitions on next page

FH Region Growing Method

- Difference between two components is defined by:

$$\mathrm{diff}(\mathcal{C}_1, \mathcal{C}_2) = \min_{v_1 \in \mathcal{C}_1, v_2 \in \mathcal{C}_2, (v_1, v_2) \in \mathcal{E}} w(v1, v2)$$

- To define internal difference of a component, C, first construct a minimum spanning tree, M (C), then

$$\mathrm{int}(\mathcal{C}) = \max_{e \in M(\mathcal{C})} w(e)$$

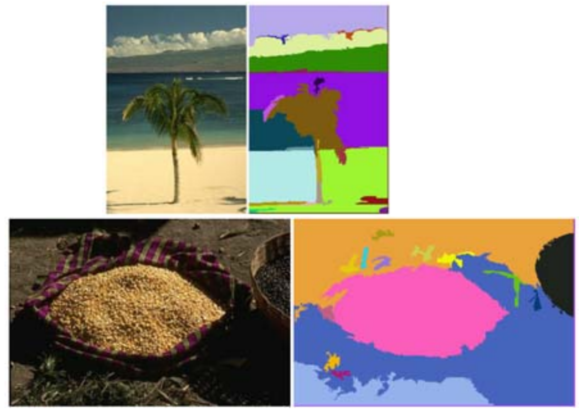
- Now define: $\mathrm{MInt}(\mathcal{C}_1, \mathcal{C}_2) = \min(\mathrm{int}(\mathcal{C}_1) + \tau(\mathcal{C}_1), \mathrm{int}(\mathcal{C}_2) + \tau(\mathcal{C}_2))$

  $\tau(\mathcal{C}) = k / |\mathcal{C}|$ k is a constant parameter

  τ (C) creates a bias against small clusters
- Merge C_1 and C_2 if diff (C_1 , C_2) < Mint (C_1 , C_2)
- Start with one cluster per pixel, merge clusters iteratively
- Sort edges by weight, start with the smallest until a merge happens
  - Stop if no changes take place
- Method much faster than normalized cuts but less popular ("only" 4K citations), perhaps because it is more complex to implement.
- Not in OpenCV but code available from the authors

5

FH Results



6

Energy Minimization Approach

- FP 9.4.3 but following is presented a bit differently
- General Idea
  - Task is to label each pixel as being *foreground* or *background*
  - Requires a model of FG/BG (*e.g.* an intensity/color *pdf*)
  - Label assigned to a pixel depends not only on its local properties but also on the labels of the pixels in the neighborhood, to enforce some smoothness.
- Commonly expressed as the task of minimizing an energy function

$$E(f) = E_{smooth}(f) + E_{data}(f)$$

- Data and smoothness functions need to be defined by the designer
  - For example, data energy can be the absolute difference between observed intensity and expected intensity (based on FG/BG models)
  - Smoothness term on next slide

7

Energy Minimization Approach

- Smoothness energy can be defined as being sum of terms for each neighbor considered
  - Common to use four neighbors
  - One commonly used smooth energy term:
    - The term is zero if the two neighbors have the same label, and another value if they are different.
    - This value may be different for different labels (background or foreground) and for different neighbors.
- For certain classes of energy functions ("sub-modular" functions), the total energy can be minimized by solving a related graph-cut problem (different from normalized cut approach)
  - See next slide

8

Graph Cut Formulation

- S and T are two “terminal” nodes corresponding to object and background
- Each pixel node has links to both S and T nodes: strength (capacity) depends on similarity of node to background/foreground
- Links between neighboring nodes; strength depends on similarity of labels
- Minimum energy (max probability) found by min-cut or max-flow algorithms
- We omit details of how to compute the min-cut

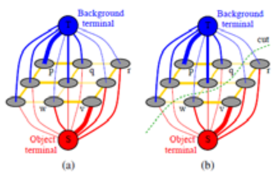


Fig 5.23, RS Book

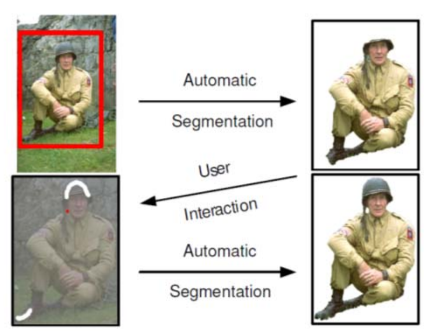
Grabcut

- A bounding box is provided around the object of interest.
- This box is used to compute properties (such as a color distribution) of the object and the background
- Min-cut method is used to segment
- After initial segmentation, improved models can be obtained and the process can be repeated iteratively.
- Further interactions may be used for further refinement.
- Some examples on following slides.

Grabcut Example

Figure 5.24: *GrabCut image segmentation (Rother et al. 2004) © 2004 ACM: (a) the user draws a bounding box in red; (b) the algorithm guesses color distributions for the object and background and performs a binary segmentation; (c) the process is repeated with better region statistics.*

Another Example with Editing



Region Segmentation Summary

- Several divisive and agglomerative methods
- Performance depends on choice of measures of uniformity (affinity), energy functions, thresholds in some cases
- Common problems
  - Regions of interests are not necessarily uniform in a property
  - Shape and context affect segmentation but hard to accommodate in current computer algorithms.

13

Edge Detection

- Compute discontinuities in image properties (intensity, color, texture ……)
  - More local approach than region segmentation
  - Better localization, incomplete boundaries, *over*-segmentation
- Approach
  - Measurement of derivatives (Gradient, Laplacian…..)

14

Computing Derivatives

- 1-D function, f(x)
  - Differences of neighboring values
    - $\Delta(x)= f(x)- f(x+1)$ or $f(x) – f(x-1)$ or $f(x+1) – f(x-1)$
    - Consider more neighbors, weight the differences
- 2-D function, f(x,y)
  - Can compute derivatives *w.r.t.* x and y separately, say f_x, f_y
  - Gradient is the vector (f_x, f_y)
    - Magnitude: $\sqrt{(f_x^2 + f_y^2)}$
    - Angle: $\tan^{-1}(f_y / f_x)$
  - Laplacian (non-drectional) :

$$(\nabla^2 f)(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

15

Convolution

- Computing derivatives and gradients is equivalent to convolution of image with an *edge mask*
- Examples (1-D: difference, 2-D: Sobel edge mask)

-1	0	1

-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

16

Smoothing and Computing Gradients

- Common to smooth with Gaussian (scale parameter to be supplied) prior to differentiation
- Convolve image with Gaussian and then differentiate
- Equivalent to differentiate the Gaussian and then convolve

$$\frac{\partial (G_\sigma * * I)}{\partial x} = (\frac{\partial G_\sigma}{\partial x}) * * I$$

  - Notation: “**” stands for convolution
  - Derivative of smoothing functions can be pre-computed

Canny Edge Detector

- >27000 citations
- Smooth with Gaussian (need to supply a scale parameter)
- Compute gradient direction
- Compute (estimate) derivative in the direction of the gradient
  - Non-maxima suppression to detect position
- Thresholding
  - High threshold: always accept points with higher values
  - Low threshold: always reject points with lower values
  - Accept points with in between values if connected to an existing edge (hysteresis)
- Some details may be found in:

http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/MARBLE/low/edges/canny.htm

Canny Examples

- Three scales

  - How to choose the “right” scale
    - From *a priori* knowledge of desired objects
    - Use large sigma to find prominent edges, reduce smoothing for better localization, requires tracking across scales

Color Edges

- Color is a 3-D quantity
  - How to compute its gradient?
    - Compute edges in each component and combine (use OR or AND operator?)
    - Combine three components in one (basically gives intensity)
    - Computing derivative of hue is difficult (angle is measured *modulo* 360 degrees)
- Some Observations
  - Intensity and hue edges are highly correlated
    - Unlikely that two surfaces have different hue but same intensity
    - Humans do not localize pure hue edges well