CS664 Computer Vision

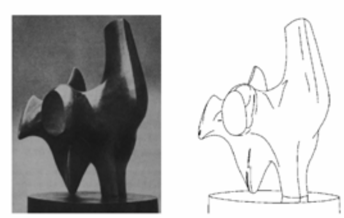
3. Edges

Dan Huttenlocher

Cornell University
Faculty of Computing and Information Science

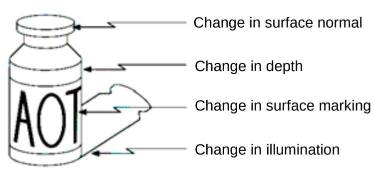
Edge Detection

- Convert a gray or color image into set of curves
  - Represented as binary image
- Capture properties of shapes



Several Causes of Edges

- Sudden changes in various properties of scene can lead to intensity edges
  - Scene changes result in changes of image brightness/color

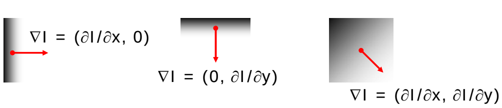


Detecting Edges

- Seek sudden changes in intensity
  - Various derivatives of image
- Idealized continuous image I(x,y)
- Gradient (first derivative), vector valued
  $$\nabla I = (\partial I/\partial x, \partial I/\partial y)$$
- Squared gradient magnitude
  $$\|\nabla I\|^2 = (\partial I/\partial x)^2 + (\partial I/\partial y)^2$$
  - Avoid computing square root
- Laplacian (second derivative)
  $$\nabla^2 I = \partial^2 I/\partial x^2 + \partial^2 I/\partial y^2$$

The Gradient

- Direction of most rapid change



- Gradient direction is atan($\partial I/\partial y$,$\partial I/\partial x$)
  - Normal to edge
- Strength of edge given by grad magnitude
  - Often use squared magnitude to avoid computing square roots

Finite Differences

- Images are digitized
  - Idealized continuous underlying function I(x,y) realized as discrete values on a grid I[u,v]
- Approximations to derivatives (1D)
  $dF/dx \approx F[u+1] - F[u]$
  $d^2F/dx^2 \approx F[u-1] - 2F[u] + F[u+1]$

1	0	1	0	10	11	11	0	1	
-1	1	-1	10	1	0	-11	1	0	dF: edge at extremum
-2	2	-2	11	-9	-1	-11	12	-2	d^2F: edge at zero crossing

- First derivative shifts grid

Discrete Gradient

- Partial derivatives estimated for boundaries between adjacent pixels
  - E.g., pixel and next one in x,y directions
- Yields estimates at different points in each direction if use x,y directions

2	1	2	3
1	0	1	2
1	0	1	2
2	1	2	3

- Generally use 45° directions to solve this
  - Magnitude fine, but gradient orientation needs to be rotated to correspond to axes

Estimating Discrete Gradient

- Gradient at u,v with 45° axes
  - Down-right: $\partial I/\partial x' \approx I[u+1,v+1]-I[u,v]$
  - Down-left: $\partial I/\partial y' \approx I[u,v+1]-I[u+1,v]$
- Handle image border, e.g., no change

u,v	u+1,v
u,v+1	u+1,v+1

I

2	1	1	2
7	6	7	6
1	6	7	7
2	2	7	6

$\partial I/\partial x'$

4	6	5	0
-1	1	0	0
1	1	-1	0
0	0	0	0

$\partial I/\partial y'$

0	6	5	5
0	-5	-1	1
0	-4	-5	0
0	0	0	0

$\|I\|^2$

16	72	50	25
1	26	1	1
1	17	26	0
0	0	0	0

Discrete Laplacian

- Laplacian at u,v
  - $\partial^2 I/\partial x^2 = I[u-1,v]-2I[u,v]+I[u+1,v]$
  - $\partial^2 I/\partial y^2 = I[u,v-1]-2I[u,v]+I[u,v+1]$
  - $\nabla^2 I$ is sum of directional second derivatives: $I[u-1,v]+I[u+1,v]+I[u,v-1]+I[u,v+1]-4I[u,v]$
- Can view as 3x3 mask or kernel
  - Value at u,v given by sum of product with I
- Grid yields poor rotational symmetry

	1	
1	-4	1
	1	

1		1
	-4	
1		1

Local Edge Detectors - Convolution

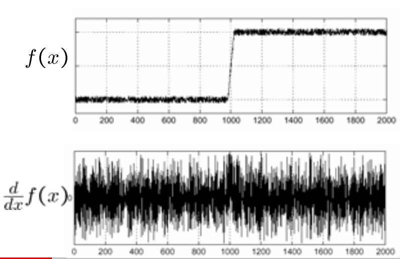
- Historically several local edge operators based on derivatives
  - Simple local weighting over small set of pixels
- For example Sobel operator
  - First derivatives in x and y
  - Weighted sum
  - 3x3 mask for symmetry
  - Today can do better with larger masks, fast algorithms, faster computers

-1		1
-2		2
-1		1

1	2	1
-1	-2	-1

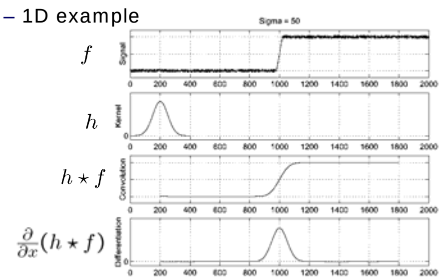
Problems With Local Detectors

- 1D example illustrates effect of noise (variation) on local measures



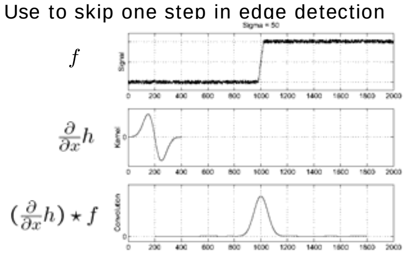
Convolution and Derivatives

- Smooth and then take derivative
  - 1D example



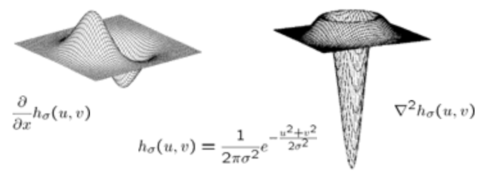
Derivatives and Convolutions

- Another useful identity for convolution is d/dx(A★B) = (d/dx A)★B = A★(d/dx B)
  - Use to skip one step in edge detection



Area of Support for Derivative Operators

- Directional first derivatives and second derivative (Laplacian) of Gaussian
  - Sigma controls scale, larger yields fewer edges



Derivatives Using Convolution

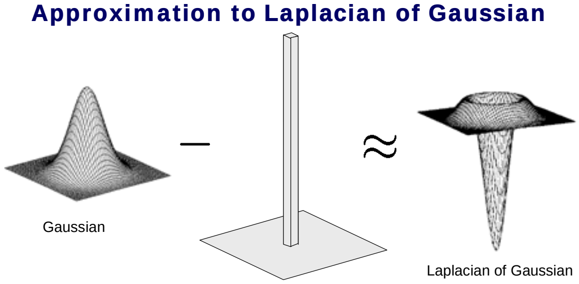
- When smoothing all weights of mask h are positive
  - Sum to 1
  - Maximum weight at center of mask
- For derivatives have negative weights
  - Compute differences (derivatives)
  - E.g., Laplacian H = $\begin{array}{|c|c|c|} \hline 1 & 4 & 1 \\ \hline 4 & -20 & 4 \\ \hline 1 & 4 & 1 \\ \hline \end{array}$
  - H★F = ∇^2F
  - Sum to 0

Derivatives and Convolution

- Difference of image and smoothed version

Original | Smoothed | Difference (brightened)

Approximation to Laplacian of Gaussian



(A★G)-(A★I) = A★(G-I) ≈ (∇^2G)★A = ∇^2(A★G)

Linear Operators

- Linear shift invariant (LSI) system
  - Given a “black box” h: f → $\boxed{h}$ → g
  - Linearity: $af_1+bf_2 \rightarrow \boxed{h} \rightarrow ag_1+bg_2$
  - Shift invariance: f(x-u) → $\boxed{h}$ → g(x-u)
- Convolution with arbitrary h equivalent to these properties
  - Beyond this course to show it
- Linearity is “simple to understand” but real world not always linear
  - E.g., saturation effects

Gradient Magnitude

- Also use smoothed image

  $\|\nabla(I \star h_\sigma)\| = ((\partial(I \star h_\sigma)/\partial x)^2 + (\partial(I \star h_\sigma)/\partial y)^2)^{.5}$


What Makes Good Edge Detector

- Goals for an edge detector
  - Minimize probability of multiple detection
    - Two pixels classified as edges corresponding to single underlying edge in image
  - Minimize probability of false detection
  - Minimize distance between reported edge and true edge location
- Canny analyzes in detail 1D step edge
  - Shows that derivative of Gaussian is optimal with respect to above criteria
  - Analysis does not extend easily to 2D

Canny Edge Detector

- Based on gradient magnitude and direction of Gaussian smoothed image
  - Magnitude: $\|\nabla(G_\sigma \star I)\|$
  - Direction (unit vector): $\nabla(G_\sigma \star I) / \|\nabla(G_\sigma \star I)\|$
- Ridges in gradient magnitude
  - Peaks in direction of gradient (normal to edge) but not along edge
- Hysteresis mechanism to threshold strong edges
  - Ridge pixel above lo threshold
  - Connected via ridge to pixel above hi threshold

Canny Edge Definition

- Let $(\delta_x,\delta_y) = \nabla(G_\sigma \star I) / \|\nabla(G_\sigma \star I)\|$
  - Note compute without explicit square root
- Let $m = \|\nabla(G_\sigma \star I)\|^2$
- Non-maximum suppression (NMS)
  - $m(x,y) > m(x+\delta_{\mathbf{x}}(x,y), y+\delta_{\mathbf{y}}(x,y))$
  - $m(x,y) \geq m(x-\delta_{\mathbf{x}}(x,y), y-\delta_{\mathbf{y}}(x,y))$
  - Select “ridge points”
- Still leaves many candidate edge pixels
  - E.g., $\sigma=1$

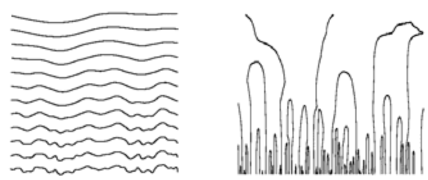

Canny Thresholding

- Two level thresholding of candidate edge pixels (those that survive NMS)
  - Above lo and <u>connected to</u> pixel above hi
- Start by keeping (classifying as edges) all candidates above hi threshold
  - <u>Recursively</u> if pixel above lo threshold and adjacent to an edge pixel keep it
- Perform recursion using bfs/dfs
  - E.g., $\sigma=1$, lo=5, hi=10 and lo=10, hi=20


Multiscale Edges

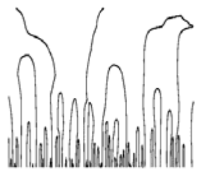
- Multi-scale image

  $I(x,y,\sigma) = I(x,y) \star G_\sigma(x,y)$
- Extract edges at across scales
  - Notion of scale-space introduced by Witkin



Scale Space

- As scale increases
  – edge position can change
  – edges can disappear
  – new edges are not created



- Important to consider different scales
  – Or know certain scale is important a priori