**CS664 Computer Vision**

3. Edges

Dan Huttenlocher

---

**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

  - Change in depth
  - Change in surface marking
  - Change in illumination
  - Change in surface normal

---

**Detecting Edges**

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

---

**The Gradient**

- Direction of most rapid change
  \[ \nabla I = \left( \frac{\partial I}{\partial x}, 0 \right) \]
  \[ \nabla I = (0, \frac{\partial I}{\partial y}) \]
  \[ \nabla I = \left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) \]

- Gradient direction is $\text{atan}(\partial I/\partial y, \partial I/\partial x)$
  - 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)
  \[ \frac{dF}{dx} = F[u+1] - F[u] \]
  \[ \frac{dF}{dx^2} = F[u-1] - 2F[u] + F[u+1] \]
- Dots: edge at extremum
- Dots: 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
- Generally use 45° directions to solve this
  - Magnitude fine, but gradient orientation needs to be rotated to correspond to axes

**Discrete Laplacian**
- Laplacian at u,v
  \[
  \frac{\partial^2 I}{\partial x^2} = I[u-1,v] - 2I[u,v] + I[u+1,v]
  \]
  \[
  \frac{\partial^2 I}{\partial y^2} = I[u,v-1] - 2I[u,v] + I[u,v+1]
  \]
- \(\nabla^2 I\) 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

**Problems With Local Detectors**
- 1D example illustrates effect of noise (variation) on local measures

**Estimating Discrete Gradient**
- Gradient at u,v with 45° axes
  - Down-right: \(\frac{\partial I}{\partial x'} = I[u+1,v+1]-I[u,v]\)
  - Down-left: \(\frac{\partial I}{\partial y'} = I[u,v+1]-I[u+1,v]\)
- Handle image border, e.g., no change

**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

**Convolution and Derivatives**
- Smooth and then take derivative
  - 1D example
Derivatives and Convolutions

- Another useful identity for convolution is 
  \[ \frac{d}{dx}(A \ast B) = (\frac{d}{dx} A) \ast B = A \ast (\frac{d}{dx} B) \]
  - 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 \( H = \nabla^2 \)
  - Sum to 0

Approximation to Laplacian of Gaussian

\[ (A \ast G) - (A \ast I) = A \ast (G - I) = \nabla^2 G \ast A = \nabla^2 (A \ast G) \]

Linear Operators

- Linear shift invariant (LSI) system
  - Given a “black box” \( h \): \( f \longrightarrow h \longrightarrow g \)
  - Linearity: \( af_1 + bf_2 \longrightarrow h \longrightarrow ag_1 + bg_2 \)
  - Shift invariance: \( f(x-u) \longrightarrow h \longrightarrow g(x-u) \)
- 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
  \[ \| \nabla(I * h_\sigma) \| = \sqrt{\left( \frac{\partial}{\partial x}(I * h_\sigma) \right)^2 + \left( \frac{\partial}{\partial y}(I * h_\sigma) \right)^2} \]

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: \[ \| \nabla(G_\sigma * I) \| \]
  - Direction (unit vector): \[ \frac{\nabla(G_\sigma * I)}{\| \nabla(G_\sigma * I) \|} \]
- 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 \((\delta_x, \delta_y) = \frac{\nabla(G_\sigma * I)}{\| \nabla(G_\sigma * I) \|} \)
  - Note compute without explicit square root
- Let \(m = \| \nabla(G_\sigma * I) \|^2 \)
- Non-maximum suppression (NMS)
  - \(m(x,y) > m(x+\delta_x(x,y), y+\delta_y(x,y)) \)
  - \(m(x,y) \geq m(x-\delta_x(x,y), y-\delta_y(x,y)) \)
  - 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 connected to pixel above hi
- Start by keeping (classifying as edges) all candidates above hi threshold
  - Recursively if pixel above lo threshold and adjacent to an edge pixel keep it
- Perform recursion using bfs/dfs
  - E.g., \(\sigma = 1\), \(lo = 5\), \(hi = 10\), \(lo = 10\), \(hi = 20\)

Multiscale Edges

- Multi-scale image
  \[ I(x,y,\sigma) = I(x,y) * G_\sigma(x,y) \]
- 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