Outlines

- What are edges
- Image gradient
- Smoothing and image gradient
- Laplacian for edge detection / Laplacian of Gaussian
- Gradient-based edge detection
- Hysteresis
- Edge detection by function fitting.
What are edges

- What is an edge? A sharp change in brightness
- What generates an edge (Where edges occur)?
  - Boundaries between objects
  - Reflectance changes (within object)
  - Illumination changes: e.g., cast shadow boundary (within object)
  - Change in surface orientation (within object)

- Edge: A sharp change in brightness
- But which changes we would like to mark? Meaningful changes. Hard to defined.
- How to tell a semantically meaningful edge from a nuisance edge?
- Both low level and high level information
Biological

- We have seen evidence before of edge/bar detectors at different stages of our visual system.

*Figure 1.8* Bar stimuli of different orientations (left) and the responses they evoke from a simple cell in primary visual cortex (right). From D. H. Hubel, Eye, Brain, and Vision, New York, Scientific American Library, 1988.
Figure 1.9 Illustration of the idea that simple cells result from the feedforward convergence of a set of center-surround cells.

Figure 1.11 Idealized depiction of the organization of orientation selectivity and ocular dominance in primary visual cortex.
Characteristic of an edge

- Edge: A sharp change in brightness
- Ideal edge is a step function in certain direction.

![Graph showing ideal edge characteristics]

Characteristic of an edge

- Ideal edge is a step function in certain direction.
- The first derivative of $I(x)$ has a peak at the edge.
- The second derivative of $I(x)$ has a zero crossing at the edge.

![Graph showing ideal edge characteristics]
• More realistically, image edges are blurred and the regions that meet at those edges have noise or variations in intensity.
  – blur - high first derivatives near edges
  – noise - high first derivatives within regions that meet at edges

Mathematical Edge Model

An edge is a step function of unknown height in the presence of stationary additive Gaussian noise

\[ \text{edge}(x) = A U(x) + n(x) \]

\[ U(x) = \begin{cases} 
0 & x < 0 \\
1 & x > 0 
\end{cases} \]
Edge detection in 2-D: Image Gradient

- Let \( f(x,y) \) be the image intensity function. It has derivatives in all directions
  - the gradient is a vector whose first component is the direction in which the first derivative is highest, and whose second component is the magnitude of the first derivative in that direction.
- If \( f \) is continuous and differentiable, then its gradient can be determined from the directional derivatives in any two orthogonal directions - standard to use \( x \) and \( y \)
  
  \[
  \begin{align*}
  \text{magnitude} &= \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \\
  \text{direction} &= \tan^{-1}\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right)
  \end{align*}
  \]

Partial Image derivatives

- With a digital image, the partial derivatives are replaced by finite differences:
  - \( \Delta_x f = f(x,y) - f(x-1, y) \)
  - \( \Delta_y f = f(x,y) - f(x, y-1) \)
- Alternatives are:
  - \( \Delta_x^2 f = f(x+1,y) - f(x-1,y) \)
  - \( \Delta_y^2 f = f(x,y+1) - f(x,y-1) \)
- Robert’s gradient
  - \( \Delta_x f = f(x+1,y+1) - f(x,y) \)
  - \( \Delta_y f = f(x,y+1) - f(x+1, y) \)

\[
\begin{array}{ccc}
\text{Prewitt} & \text{Sobel} \\
\begin{bmatrix} -1 & 0 & 1 \\ -1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & -1 & 0 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}
\end{array}
\]
Finite differences and noise

• Finite difference filters respond strongly to noise
  – obvious reason: image noise results in pixels that look very different from their neighbors

• What is to be done?
  – intuitively, most pixels in images look quite a lot like their neighbors
  – this is true even at an edge; along the edge they’re similar, across the edge they’re not
  – suggests that smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors
The response of a linear filter to noise (Recall)

- **Recall noise model:** stationary independent additive Gaussian noise with zero mean (non-zero mean is easily dealt with)
- **Mean:**
  - output is a weighted sum of inputs
  - so we want mean of a weighted sum of zero mean normal random variables
  - must be zero
  \[ f_{\text{observed}}(x, y) = f(x, y) + N(0, \sigma^2) \]
  \[ g \ast f_{\text{observed}} = g \ast f + g \ast N(0, \sigma^2) \]

- **Variance:**
  - recall
  - variance of a sum of random variables is sum of their variances
  - variance of constant times random variable is constant^2 times variance
  - then if \( \sigma^2 \) is noise variance and kernel is \( K \), variance of response is
  \[ \sigma^2 \sum_{u,v} K^2_{u,v} \]

This can magnify or reduce the variance of the noise based on \( \sum_{u,v} K^2_{u,v} \)

If \( \sum_{u,v} K_{u,v} = 1 \Rightarrow \sum_{u,v} K^2_{u,v} \leq 1 \) This reduces noise variance
Smoothing + Differentiation

- So smoothing should help.
- Recall: smoothing and differentiation are linear filters
- Recall also: linear filters are associative

\[ K_{\partial I/\partial x} * (g * I) = (K_{\partial I/\partial x} * g) * I = \frac{\partial g}{\partial x} * I \]

- Smoothing then differentiation ≡ convolution with the derivative of the smoothing kernel.
- If Gaussian is used for smoothing: We need to convolve the image with derivative of the Gaussian

\[ \frac{\partial G_\sigma}{\partial x} * I \quad \frac{\partial G_\sigma}{\partial y} * I \]

Noise σ=3%  Noise σ=9%

x- derivative - No smoothing

Convolution with x-derivative of a Gaussian (σ=1 pixel)
• The scale ($\sigma$) of the Gaussian has significant effects on the results

1 pixel  3 pixels  7 pixels

Detecting Edges

Approach 1: Using Laplacian

• Recall: an edge corresponds to a zero crossing at the second derivative

• Laplacian:

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

• Its digital approximation is:

$$\nabla^2 f(x, y) = [f(x+1, y) - f(x, y)] - [f(x, y) - f(x-1, y)] + [f(x, y+1) - f(x, y)] - [f(x, y) - f(x, y-1)]$$

$$= [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4 f(x, y)$$
Laplacian of Gaussian

- Laplacian is a linear filter
- Bad idea to apply a Laplacian without smoothing
- If we smooth by a Gaussian before applying a Laplacian:

\[ K_{\nabla^2} \ast (G_\sigma \ast I) = (K_{\nabla^2} \ast G_\sigma) \ast I = (\nabla^2 G_\sigma) \ast I \]

Laplacian of Gaussian (LoG)

"Mexican Hat"

- Can be approximated as difference of two Gaussians
- This is called Difference of Gaussians filter DoG

\[ \nabla^2 g(x) \approx c_1 e^{\frac{x^2}{2\sigma_1^2}} - c_2 e^{\frac{x^2}{2\sigma_2^2}} \]

\( \sigma_1 < \sigma_2 \)
Algorithm:
• Convolve the image with a LoG
• Mark the point with zero crossings:
  – these are pixels whose LoG is positive and which have neighbor’s
    whose LoG is negative or zero
• Check these points to ensure the gradient magnitude is
  large (to avoid low contrast edges)

• Note: Two parameters: Gaussian scale, contrast threshold
Laplacian of Gaussian

13 x 13 Mexican hat

zero crossings

sigma=2
contrast=1
contrast=4
LoG zero crossings

sigma=2
contrast=4
Problems with the Laplacian approach

- Poor behavior at corners
- Computationally: we need to compute both the LoG and the gradient.

We have unfortunate behavior at corners
Gradient-Based Edge Detectors

General strategy

- determine image gradient (with smoothing)
- now mark points where gradient magnitude is particularly large wrt neighbors

- The gradient magnitude is large along thick trail; how do we identify the significant points?

We wish to mark points along the curve where the magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: at which point is the maximum, and where is the next one?
Non-maxima suppression

- Non-maxima suppression - Retain a point as an edge point if:
  - its gradient magnitude is higher than a threshold
  - its gradient magnitude is a local maxima in the gradient direction

Simple thresholding will compute thick edges.

Non-maximum suppression

At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.
Predicting the next edge point

Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).

Problem of scale and threshold

• Usually, any single choice of $\sigma$ does not produce a good edge map
  – a large $\sigma$ will produce edges from only the largest objects, and they will not accurately delineate the object because the smoothing reduces shape detail
  – a small $\sigma$ will produce many edges and very jagged boundaries of many objects.

• Scale-space approaches
  – detect edges at a range of scales $[\sigma_1, \sigma_2]$
  – combine the resulting edge maps
    • trace edges detected using large $\sigma$ down through scale space to obtain more accurate spatial localization.
fine scale high threshold
coarse scale, high threshold

course scale, low threshold
Hysteresis

- Which threshold we should use?
- Fine scale: fine details.
- Coarser scale: fine details disappear.
- Low threshold: low contrast edges. A variety of new edge points of dubious significance are introduced.
- High threshold: loose low contrast edges ⇒ broken edges
- Solution: use two thresholds
  - Larger threshold: more certain edge, use to start an edge chain
  - Smaller threshold: use to follow the edge chain

Edge detection by function fitting

- General approach
  - Fit a function to each neighborhood of the image
  - Use the gradient of the function as the digital gradient of the image neighborhood
Edge detection by function fitting

- Example: fit a plane to a 2x2 neighborhood
  - \( z = ax + by + c; \) \( z \) is gray level - need to determine \( a, b, c \)
  - gradient is then \((a^2 + b^2)^{1/2}\)
  - neighborhood points are \( f(x,y), f(x+1,y), f(x,y+1) \) and \( f(x+1,y+1) \)

- Need to minimize

\[
E(a, b, c) = \sum_{i=0}^{1} \sum_{j=0}^{1} [a(x+i) + b(y+j) + c - f(x+i,y+j)]^2
\]

- Solve this and similar problems by:
  - differentiating with respect to \( a, b, c \), setting results to 0, and
  - solving for \( a, b, c \) in resulting system of equations

\[
a = \frac{[f(x+1,y) + f(x+1,y+1) - f(x,y) - f(x+1,y)]}{2}
b = \frac{[f(x,y+1) + f(x+1,y+1) - f(x,y) - f(x+1,y)]}{2}
\]

- \( a \) and \( b \) are the \( x \) and \( y \) partial derivatives
Edge detection by function fitting

- Could also fit a higher order surface than a plane
  - with a second order surface we could find the (linear) combination of pixel values that corresponds to the higher order derivatives, which can also be used for edge detection
- Would ordinarily use a neighborhood larger than 2x2
  - better fit
  - for high degree functions need more points for the fit to be reliable.

Edge linking and following

- Group edge pixels into chains and chains into large pieces of object boundary.
  - can use the shapes of long edge chains in recognition
    - slopes
    - curvature
    - Corners
- Basic steps
  - thin connected components of edges to one pixel thick
  - find simply connected paths
  - link them at corners into a graph model of image contours
  - compute local and global properties of contours and corners
Sources

- Forsyth and Ponce, Computer Vision a Modern approach: chapter 8.
- Slides by
  - D. Forsyth @ Berkeley
  - L.S. Davis @UMD
  - G.C. Stockman @MSU