### BBM 413
Fundamentals of Image Processing

Erkut Erdem  
Dept. of Computer Engineering  
Hacettepe University

**Edge Detection**  
**Hough Transform**

---

**Review – Signals and Images**

- A signal is composed of low and high frequency components

  - low frequency components: smooth/piecewise smooth  
  - Neighboring pixels have similar brightness values  
  - You’re within a region

  - high frequency components: oscillatory  
  - Neighboring pixels have different brightness values  
  - You’re either at the edges or noise points

---

**Review - Low-pass, Band-pass, High-pass filters**

- **low-pass:**

- **High-pass / band-pass:**

  - ![Image Example](image.png)

  - ![Image Example](image.png)

  - ![Image Example](image.png)

---

**Today**

- Edge detection  
  - Difference filters  
  - Laplacian of Gaussian  
  - Canny edge detection

- Boundary detection  
  - Hough transform

---

Slide credit: A. Efros
Today

• Edge detection
  • Difference filters
  • Laplacian of Gaussian
  • Canny edge detection
• Boundary detection
  • Hough transform

Edge detection

• **Goal:** Identify sudden changes (discontinuities) in an image
  – Intuitively, most semantic and shape information from the image can be encoded in the edges
  – More compact than pixels

• **Ideal:** artist’s line drawing (but artist is also using object-level knowledge)

Why do we care about edges?

• Extract information, recognize objects

• Recover geometry and viewpoint

Closeup of edges

Source: J. Hays

Slide credit: D. Hoiem
What causes an edge?

- Reflectance change: appearance, information, texture
- Depth discontinuity: object boundary
- Change in surface orientation: shape
- Cast shadows

Slide credit: K. Grauman
Characterizing edges

• An edge is a place of rapid change in the image intensity function

image intensity function (along horizontal scanline) first derivative

edges correspond to extrema of derivative

Derivatives with convolution

For 2D function $f(x,y)$, the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}$$

To implement above as convolution, what would be the associated filter?

Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial f(x,y)}{\partial y}$$

Which shows changes with respect to $x$?

Assorted finite difference filters

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$; $M_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

>> My = fspecial('sobel');
>> outim = imfilter(double(im), My);
>> imagesc(outim);
>> colormap gray;
**Image gradient**

- The gradient of an image: \( \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \)

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right] \quad \nabla f = \left[ 0, \frac{\partial f}{\partial y} \right] \]

The gradient points in the direction of most rapid increase in intensity

- How does this direction relate to the direction of the edge?

The gradient direction is given by \( \theta = \tan^{-1} \left( \frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}} \right) \)

The edge strength is given by the gradient magnitude

\[ ||\nabla f|| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]

**Gradient magnitude image**

**Original Image**

**Thresholding gradient with a lower threshold**
Thresholding gradient with a higher threshold

With a little Gaussian noise

Intensity profile

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

Where is the edge?
**Effects of noise**

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What can we do about it?

**Solution: smooth first**

To find edges, look for peaks in $\frac{d}{dx}(f \ast g)$

**Smoothing with a Gaussian**

Recall: parameter $\sigma$ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.

Effect of $\sigma$ on derivatives

The apparent structures differ depending on Gaussian’s scale parameter.

Larger values: larger scale edges detected
Smaller values: finer features detected
**So, what scale to choose?**

It depends what we’re looking for.

Slide credit: K. Grauman

### Smoothing and Edge Detection

- While eliminating noise via smoothing, we also lose some of the (important) image details.
  - Fine details
  - Image edges
  - etc.
- What can we do to preserve such details?
  - Use edge information during denoising!
  - This requires a definition for image edges.

**Chicken-and-egg dilemma!**

- Edge preserving image smoothing (Next week’s topic!)

---

**Derivative theorem of convolution**

- Differentiation is convolution, and convolution is associative:
- This saves us one operation: \( \frac{d}{dx} (f * g) = f * \frac{d}{dx} g \)

Slide credit: S. Seitz

---

**Derivative of Gaussian filter**

- \( x \)-direction
- \( y \)-direction

* \([1 -1] = \)
Derivative of Gaussian filter

- Which one finds horizontal/vertical edges?

Smoothing vs. derivative filters

- Smoothing filters
  - Gaussian: remove “high-frequency” components; “low-pass” filter
  - Can the values of a smoothing filter be negative?
  - What should the values sum to?
    - One: constant regions are not affected by the filter

- Derivative filters
  - Derivatives of Gaussian
  - Can the values of a derivative filter be negative?
  - What should the values sum to?
    - Zero: no response in constant regions
  - High absolute value at points of high contrast

Reading Assignment #4

Theory of Edge Detection
D. Marr and E. Hildreth

- One of the 60 seminal articles appeared in the journal Philosophical Transactions, which is made available online due to the celebration of 350th birthday of the Royal Society in 2010. [http://trailblazing.royalsociety.org]
- Due on 5th of December

Laplacian of Gaussian

Consider \( \frac{\partial^2}{\partial x^2} (h \ast f) \)

Where is the edge?

Zero-crossings of bottom graph
2D edge detection filters

- The Laplacian operator:

\[ \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \]

\[ h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2 + v^2}{2\sigma^2}} \]

\[ \frac{\partial}{\partial x} h_\sigma(u, v) \]

\[ \nabla^2 h_\sigma(u, v) \]

Laplacian of Gaussian

slide credit: K. Grauman

Laplacian of Gaussian

original image

Source: D. Marr and E. Hildreth (1980)

Laplacian of Gaussian

convolution with

\[ \nabla^2 h_\sigma(u, v) \]

(pos. values – white, neg. values – black)

Source: D. Marr and E. Hildreth (1980)
**Laplacian of Gaussian**

Source: D. Marr and E. Hildreth (1980)

**Designing an edge detector**

- Criteria for a good edge detector:
  - **Good detection**: the optimal detector should find all real edges, ignoring noise or other artifacts
  - **Good localization**
    - the edges detected must be as close as possible to the true edges
    - the detector must return one point only for each true edge point

- Cues of edge detection
  - Differences in color, intensity, or texture across the boundary
  - Continuity and closure
  - High-level knowledge

**The Canny edge detector**

original image (Lena)

Source: K. Grauman

Slide credit: K. Grauman

thresholding

Slide credit: K. Grauman
The Canny edge detector

How to turn these thick regions of the gradient into curves?

Non-maximum suppression

Check if pixel is local maximum along gradient direction, select single max across width of the edge
- requires checking interpolated pixels p and r

The Canny Edge Detector

Problem: pixels along this edge didn’t survive the thresholding

Hysteresis thresholding
- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

thinning
(non-maximum suppression)
**Hysteresis thresholding**

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use hysteresis
    - use a high threshold to start edge curves and a low threshold to continue them.

---

**Recap: Canny edge detector**

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. **Non-maximum suppression:**
   - Thin wide “ridges” down to single pixel width
4. **Linking and thresholding (hysteresis):**
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

- MATLAB: `edge(image, 'canny');`
Effect of $\sigma$ (Gaussian kernel spread/size)

The choice of $\sigma$ depends on desired behavior
- large $\sigma$ detects large scale edges
- small $\sigma$ detects fine features

Low-level edges vs. perceived contours

Edge detection is just the beginning...

Learn from humans which combination of features is most indicative of a "good" contour?

[D. Martin et al. PAMI 2004]

Source: S. Lazebnik

Human-marked segment boundaries

Source: S. Lazebnik
Today

- Edge detection
  - Difference filters
  - Laplacian of Gaussian
  - Canny edge detection
- Boundary detection
  - Hough transform

Edges vs. Boundaries

- Edges
  - abrupt changes in the intensity
  - discontinuities in intensity values
  - a local entity
- Edge detection may result in
  - Breaks in the edges due to non-uniform illumination
  - Spurious edges
- Boundaries
  - related to regions
  - a global entity
  - assemble of meaningful edge points
- Boundary detection requires grouping or fitting

Fitting

- Want to associate a model with observed features

For example, the model could be a line, a circle, or an arbitrary shape.

Fitting: Main idea

- Choose a parametric model to represent a set of features
- Membership criterion is not local
  - Can’t tell whether a point belongs to a given model just by looking at that point
- Three main questions:
  - What model represents this set of features best?
  - Which of several model instances gets which feature?
  - How many model instances are there?
- Computational complexity is important
  - It is infeasible to examine every possible set of parameters and every possible combination of features
Example: Line fitting

- Why fit lines?
  - Many objects characterized by presence of straight lines

Wait, why aren’t we done just by running edge detection?

Difficulty of line fitting

- **Extra** edge points (clutter), multiple models:
  - Which points go with which line, if any?
- Only some parts of each line detected, and some parts are **missing**:
  - How to find a line that bridges missing evidence?
- **Noise** in measured edge points, orientations:
  - How to detect true underlying parameters?

Voting

- It’s not feasible to check all combinations of features by fitting a model to each possible subset.
- **Voting** is a general technique where we let the features vote for all models that are compatible with it.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that receive a lot of votes.
- Noise & clutter features will cast votes too, but typically their votes should be inconsistent with the majority of “good” features.

Fitting lines: Hough transform

- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?

- **Hough Transform** is a voting technique that can be used to answer all of these questions.
  - **Main idea**:
    1. Record vote for each possible line on which each edge point lies.
    2. Look for lines that get many votes.
Finding lines in an image: Hough space

Connection between image \((x,y)\) and Hough \((m,b)\) spaces
- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  * given a set of points \((x,y)\), find all \((m,b)\) such that \(y = mx + b\)

What are the line parameters for the line that contains both \((x_0,y_0)\) and \((x_1,y_1)\)?
- It is the intersection of the lines \(b = -x_0m + y_0\) and \(b = -x_1m + y_1\)

How can we use this to find the most likely parameters \((m,b)\) for the most prominent line in the image space?
- Let each edge point in image space vote for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

Slide credit: K. Grauman
Polar representation for lines

Issues with usual \((m,b)\) parameter space: can take on infinite values, undefined for vertical lines.

\[
d : \text{perpendicular distance from line to origin}
\]
\[
\theta : \text{angle the perpendicular makes with the x-axis}
\]

\[
x \cos \theta - y \sin \theta = d
\]

Point in image space \(\rightarrow\) sinusoid segment in Hough space

Hough transform algorithm

Using the polar parameterization:

\[
x \cos \theta - y \sin \theta = d
\]

Basic Hough transform algorithm

1. Initialize \(H[d, \theta] = 0\)
2. for each edge point \(I[x,y]\) in the image
   
   for \(\theta = \left[\theta_{\min} \text{ to } \theta_{\max}\right]\) // some quantization
   
   \[
   d = x \cos \theta - y \sin \theta
   \]
   
   \(H[d, \theta] += 1\)
3. Find the value(s) of \((d, \theta)\) where \(H[d, \theta]\) is maximum
4. The detected line in the image is given by \(d = x \cos \theta - y \sin \theta\)

Time complexity (in terms of number of votes per pt)?
Impact of noise on Hough

What difficulty does this present for an implementation?

Hough transform for circles

- Circle: center (a,b) and radius r
  \[(x_i - a)^2 + (y_i - b)^2 = r^2\]
- For a fixed radius r, unknown gradient direction

Intersection: most votes for center occur here.
**Hough transform for circles**

- Circle: center \((a, b)\) and radius \(r\)
  \[(x_i - a)^2 + (y_i - b)^2 = r^2\]
- For an unknown radius \(r\), unknown gradient direction

**Hough space**

**Image space**

For every edge pixel \((x, y)\):

For each possible radius value \(r\):

For each possible gradient direction \(\theta\):

- \(a = x - r \cos(\theta)\) // column
- \(b = y + r \sin(\theta)\) // row

\[H[a, b, r] += 1\]

end

end

Time complexity per edgel??
**Example: detecting circles with Hough**

Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

Slide credit: K. Grauman

**Example: detecting circles with Hough**

Coin finding sample images from: Vivek Kwatra

Slide credit: K. Grauman

**Example: iris detection**


Slide credit: K. Grauman

**Voting: practical tips**

- Minimize irrelevant tokens first
- Choose a good grid / discretization
  
  Too fine
  
  ?
  
  Too coarse

- Vote for neighbors, also (smoothing in accumulator array)
- Use direction of edge to reduce parameters by 1
- To read back which points voted for “winning” peaks, keep tags on the votes.
**Hough transform: pros and cons**

**Pros**
- All points are processed independently, so can cope with occlusion, gaps
- Some robustness to noise: noise points unlikely to contribute consistently to any single bin
- Can detect multiple instances of a model in a single pass

**Cons**
- Complexity of search time increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- Quantization: can be tricky to pick a good grid size

**Generalized Hough Transform**

- What if we want to detect arbitrary shapes?

**Intuition:**

![Diagram](https://via.placeholder.com/150)

Now suppose those colors encode gradient directions…

**Detection procedure:**

For each edge point:
- Use its gradient orientation $\theta$ to index into stored table
- Use retrieved $r$ vectors to vote for reference point

Assuming translation is the only transformation here, i.e., orientation and scale are fixed.
Generalized Hough for object detection

• Instead of indexing displacements by gradient orientation, index by matched local patterns.

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Slide credit: S. Lazebnik

Summary

• Edge detection
  • Difference filters
  • Laplacian of Gaussian
  • Canny edge detection

• Boundary detection
  • Hough transform

Next week

• Image segmentation