Review – Signals and Images

- A signal is composed of low and high frequency components
  - low frequency components: smooth / piecewise smooth
  - high frequency components: oscillatory
  - Neighboring pixels have similar brightness values
  - You're within a region
  - Neighboring pixels have different brightness values
  - You're either at the edges or noise points

Edge Detection

- You're within a region
- You're either at the edges or noise points

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

- low-pass:
- High-pass / band-pass:

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

Source: J. Hays
**What causes an edge?**

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

**Characterizing edges**

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

  ![Intensity function](image)

  ![First derivative](image)

  ![Edges correspond to extrema of derivative](image)

  Slide credit: K. Grauman

**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 \( x \)?

---

**Image gradient**

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

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 \( \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \)

The edge strength is given by the gradient magnitude

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

---

**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: \( M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \); \( M_y = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \)

---

**Original Image**
Gradient magnitude image

Thresholding gradient with a lower threshold

Thresholding gradient with a higher threshold

Intensity profile

Slide credit: K. Grauman

Slide credit: K. Grauman

Slide credit: K. Grauman

Slide credit: D. Hoiem
With a little Gaussian noise

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?

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.

- Larger values: larger scale edges detected
- Smaller values: finer features detected

**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.

**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:

  \[
  \frac{d}{dx} (f * g) = f * \frac{d}{dx} g
  \]

  Differentiation is convolution, and convolution is associative:

  - This saves us one operation:

  \[
  f * \frac{d}{dx} g = g * \frac{d}{dx} f
  \]

- Which one finds horizontal/vertical edges?

**Derivative of Gaussian filter**

- Derivative of Gaussian filter
  - **x-direction**
  - **y-direction**

  \[ * [1 -1] = \]

**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
Historical note

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]


Theory of Edge Detection

D. Marr and E. Hildreth


2D edge detection filters

- The Laplacian operator:
  \[ \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \]

Laplacian of Gaussian

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

Where is the edge? Zero-crossings of bottom graph

Laplacian of Gaussian

Gaussian

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

derivative of Gaussian

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

Laplacian of Gaussian

\( \nabla^2 \sigma(u,v) \)

original image

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

Convolution with $\nabla^2 h_\sigma(u, v)$

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)

---

**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

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

Slide credit: L. Fei-Fei
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

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.

Hysteresis thresholding

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels
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
**Edge detection is just the beginning...**

- Berkeley segmentation database:
  [http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/](http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/)

**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 \(y = mx + b\)

---

**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 \(y = mx + b\)

---

**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 \(y = mx + b\)

---

**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 \(y = mx + b\)

---

**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 \(y = mx + b\)
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.

**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 \((x,y)\) in the image
   - for \(\theta = [\theta_{\text{min}} \text{ to } \theta_{\text{max}}]\) // 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)?
**Hough transform algorithm**

Showing longest segments found

---

**Impact of noise on Hough**

What difficulty does this present for an implementation?

---

**Impact of noise on Hough**

Here, everything appears to be “noise”, or random edge points, but we still see peaks in the vote space.

---

**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
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

\[
(x_i - a)^2 + (y_i - b)^2 = r^2
\]

Intersection: most votes for center occur here.

Hough space

Image space

Slide credit: K. Grauman

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

\[
(x_i - a)^2 + (y_i - b)^2 = r^2
\]

Slide credit: K. Grauman

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

\[
(x_i - a)^2 + (y_i - b)^2 = r^2
\]

Hough space

Image space

Slide credit: K. Grauman

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, known gradient direction

\[
(x_i - a)^2 + (y_i - b)^2 = r^2
\]

Hough space

Image space

Slide credit: K. Grauman
Hough transform for circles

For every edge pixel \((x,y)\) :
  For each possible radius value \(r\):
    For each possible gradient direction \(\theta\):
      // or use estimated gradient at \((x,y)\)
      \(a = x - r \cos(\theta) \)  // column
      \(b = y + r \sin(\theta)\)  // row
      \(H[a,b,r] += 1\)
  end
end

Time complexity per edgel?

Check out online demo: [http://www.markschulze.net/java/hough/](http://www.markschulze.net/java/hough/)  Slide credit: K. Grauman

Example: detecting circles with Hough

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

Example: detecting circles with Hough

Combined detections Edges Votes: Quarter

Coin finding sample images from: Vivek Kwatra

Example: iris detection

Gradient+threshold Hough space (fixed radius) Max detections

Voting: practical tips

- Minimize irrelevant tokens first
- Choose a good grid / discretization
- 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:

Now suppose those colors encode gradient directions…

Offline procedure:
At each boundary point, compute displacement vector: \( \mathbf{r} = \mathbf{a} - \mathbf{p}_i \).
Store these vectors in a table indexed by gradient orientation \( \theta \).

[Daniel H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980]
**Generalized Hough Transform**

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

![Diagram of Hough Transform]

Assuming translation is the only transformation here, i.e., orientation and scale are fixed.

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

**Generalized Hough for object detection**

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

![Diagram of Hough Transform on object detection]

“visual codeword” with displacement vectors

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

---

**Generalized Hough for object detection**

- Instead of indexing displacements by gradient orientation, index by “visual codeword”

![Diagram of Hough Transform on object detection]

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

---

---