CS 1699: Intro to Computer Vision

Edges and Binary Images

Prof. Adriana Kovashka
University of Pittsburgh
September 15, 2015
Plan for today

• Edge detection
• Binary image analysis
Homework 1

- Due on 9/22, 11:59pm

- Review slides from 9/03 regarding how to do Part IV

- There are different ways to compute image gradients

- Office hours for 9/17 will be at 11am-12pm on 9/18
Edge detection

• **Goal**: map image from 2d array of pixels to a set of curves or line segments or contours.

• **Why?**

• **Main idea**: look for strong gradients, post-process

Figure from J. Shotton et al., PAMI 2007

Source: K. Grauman
Origin of Edges

Edges are caused by a variety of factors:
- depth discontinuity
- surface color discontinuity
- illumination discontinuity
- surface normal discontinuity

Source: S. Seitz
What causes an edge?

Depth discontinuity: object boundary

Reflectance change: appearance information, texture

Change in surface orientation: shape

Cast shadows

Source: K. Grauman
Edges/gradients and invariance

Source: K. Grauman
Closeup of edges
Closeup of edges

Source: D. Hoiem
Characterizing edges

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

Source: L. Lazebnik
Intensity profile
With a little Gaussian noise

Source: D. Hoiem
Effects of noise

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

\[ f(x) \]

\[ \frac{d}{dx} f(x) \]

Where is the edge?

Source: S. Seitz
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?

Source: D. Forsyth
Solution: smooth first

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

Source: S. Seitz
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:
  \[ \frac{d}{dx} (f * g) = f * \frac{d}{dx} g \]

- This saves us one operation:

![Graphs showing f, \frac{d}{dx} g, and f * \frac{d}{dx} g](image)

Source: S. Seitz
Derivative of Gaussian filter

- Is this filter separable?

\[ * [1 \ 0 \ -1] = \]

Source: D. Hoiem
Tradeoff between smoothing and localization

- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Source: D. Forsyth
Designing an edge detector

• Criteria for a good edge detector:
  – **Good detection**: find all real edges, ignoring noise or other artifacts
  – **Good localization**
    • detect edges as close as possible to the true edges
    • 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: L. Fei-Fei
Gradients -> edges

Primary edge detection steps:
1. Smoothing: suppress noise
2. Edge enhancement: filter for contrast
3. Edge localization
   Determine which local maxima from filter output are actually edges vs. noise
   • Threshold, Thin

Source: K. Grauman
Thresholding

- Choose a threshold value \( t \)
- Set any pixels less than \( t \) to zero (off)
- Set any pixels greater than or equal to \( t \) to one (on)

Source: K. Grauman
Gradient magnitude image
Thresholding gradient with a lower threshold

Source: K. Grauman
Thresholding gradient with a higher threshold

Source: K. Grauman
Canny edge detector

• This is probably the most widely used edge detector in computer vision
• Theoretical model: step-edges corrupted by additive Gaussian noise
• Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization

Canny edge detector

• Filter image with derivative of Gaussian
• Find magnitude and orientation of gradient
• **Non-maximum suppression:**
  – Thin wide “ridges” down to single pixel width
• **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’);`
• `>>help edge`

Source: D. Lowe, L. Fei-Fei
Example

input image (“Lena”)
Derivative of Gaussian filter

\[ x \text{-direction} \quad y \text{-direction} \]

Source: L. Lazebnik
Compute Gradients (DoG)

X-Derivative of Gaussian

Y-Derivative of Gaussian

Gradient Magnitude

Source: D. Hoiem
Get Orientation at Each Pixel

- Threshold at minimum level
- Get orientation

\[ \theta = \text{atan2}(-gy, gx) \]

Source: D. Hoiem
The Canny edge detector

Source: K. Grauman
The Canny edge detector

Source: K. Grauman
The Canny edge detector

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

Source: K. Grauman
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

Source: K. Grauman
Bilinear Interpolation

\[ f(x, y) \approx [1 - x \quad x] \begin{bmatrix} f(0, 0) & f(0, 1) \\ f(1, 0) & f(1, 1) \end{bmatrix} \begin{bmatrix} 1 - y \\ y \end{bmatrix}. \]

http://en.wikipedia.org/wiki/Bilinear_interpolation
The Canny edge detector

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

thinning
(non-maximum suppression)

Source: K. Grauman
Hysteresis thresholding

- Use a high threshold to start edge curves, and a low threshold to continue them.

Source: S. Seitz
Hysteresis thresholding

original image

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Source: L. Fei-Fei
Hysteresis thresholding

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Source: L. Fei-Fei
Recap: Canny edge detector

- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient
- **Non-maximum suppression:**
  - Thin wide “ridges” down to single pixel width
- **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');`
- `>>help edge`

Source: D. Lowe, L. Fei-Fei
The choice of $\sigma$ depends on desired behavior

- large $\sigma$ detects large scale edges
- small $\sigma$ detects fine features

Source: S. Seitz
Low-level edges vs. perceived contours

Source: K. Grauman
Low-level edges vs. perceived contours

Berkeley segmentation database:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Source: L. Lazebnik
pB Boundary Detector

Martin, Fowlkes, Malik 2004: Learning to Detection Natural Boundaries…
Learn from humans which combination of features is most indicative of a “good” contour?
pB Boundary Detector

Figure from Fowlkes
Results

[Image of a wolf on snow-covered ground]

Human (0.95)

Pb (0.88)

Source: D. Hoiem
Results

Source: D. Hoiem
For more:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/bench/html/108082-color.html
Plan for today

• Edge detection
• Binary image analysis
Binary images
Binary image analysis: basic steps

- Convert the image into binary form
  - Thresholding
- Clean up the thresholded image
  - Morphological operators
- Extract separate blobs
  - Connected components
- Describe the blobs with region properties

Source: K. Grauman
Binary images

• Two pixel values
  – Foreground and background
  – Mark region(s) of interest

Source: K. Grauman
Thresholding

• Grayscale -> binary mask
• Useful if object of interest’s intensity distribution is distinct from background

\[
F_T[i, j] = \begin{cases} 
1 & \text{if } F[i, j] \geq T \\
0 & \text{otherwise.}
\end{cases}
\]

\[
F_T[i, j] = \begin{cases} 
1 & \text{if } T_1 \leq F[i, j] \leq T_2 \\
0 & \text{otherwise.}
\end{cases}
\]

\[
F_T[i, j] = \begin{cases} 
1 & \text{if } F[i, j] \in Z \\
0 & \text{otherwise.}
\end{cases}
\]

• Example

Source: K. Grauman
Thresholding

- Given a grayscale image or an intermediate matrix $\rightarrow$ threshold to create a binary output.

Example: edge detection

```
fg_pix = find(gradient_mag > t);
```

Looking for pixels where gradient is strong.

Source: K. Grauman
Thresholding

• Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: background subtraction

Looking for pixels that differ significantly from the “empty” background.

Source: K. Grauman

\[ \text{fg\_pix} = \text{find}(\text{diff} > t); \]
Thresholding

• Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: intensity-based detection

Looking for dark pixels

\[ \text{fg\_pix} = \text{find}(\text{im} < 65); \]

Source: K. Grauman
Thresholding

• Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: color-based detection

```matlab
fg_pix = find(hue > t1 & hue < t2);
```

Looking for pixels within a certain hue range.

Source: K. Grauman
A nice case: bimodal intensity histograms

Ideal histogram, light object on dark background

Actual observed histogram with noise

Source: K. Grauman
Issues

• What to do with “noisy” binary outputs?
  – Holes
  – Extra small fragments

• How to demarcate multiple regions of interest?
  – Count objects
  – Compute further features per object

Source: K. Grauman
Morphological operators

• Change the shape of the foreground regions via intersection/union operations between a scanning structuring element and binary image.

• Useful to clean up result from thresholding

• Basic operators are:
  – Dilation
  – Erosion

Source: K. Grauman
Dilation

- Expands connected components
- Grow features
- Fill holes

Before dilation

After dilation

Source: K. Grauman
Erosion

- Erode connected components
- Shrink features
- Remove bridges, branches, noise

Before erosion

After erosion

Source: K. Grauman
Structuring elements

• **Masks** of varying shapes and sizes used to perform morphology, for example:

• Scan mask across foreground pixels to transform the binary image

```
>> help strel
```
Dilation vs. Erosion

At each position:

- **Dilation**: if current pixel is foreground, OR the structuring element with the input image.

Source: K. Grauman
Example for Dilation (1D)

Input image

Structuring Element

Output Image

\[ g(x) = f(x) \oplus SE \]
Example for Dilation

Input image

\[1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1\]

Structuring Element

\[1 \ 1 \ 1 \ 1\]

Output Image

\[1 \ 1\]

Source: K. Grauman
Example for Dilation

Input image

Structuring Element

Output Image

Source: K. Grauman
Example for Dilation

Input image:  

1 0 0 0 1 1 1 1 0 1 1 1

Structuring Element:  

1 1 1 1

Output Image:  

1 1 0 0
Example for Dilation

Input image:

```
1 0 0 0 1 1 1 1 0 1 1
```

Structuring Element:

```
1 1 1 1
```

Output Image:

```
1 1 0 1 1 1 1 1 1 1 1
```

Source: K. Grauman
Example for Dilation

Input image: 1 0 0 0 1 1 1 1 0 1 1 1

Structuring Element: 1 1 1 1

Output Image: 1 1 0 1 1 1 1 1 1 1 1
Example for Dilation

Input image:

```
1 0 0 0 1 1 1 1 0 1 1 1
```

Structuring Element:

```
1 1 1 1
```

Output Image:

```
1 1 0 1 1 1 1 1 1 1
```
Example for Dilation

**Input image**

```
1 0 0 0 1 1 1 1 1 0 1 1
```

**Structuring Element**

```
1 1 1 1
```

**Output Image**

```
1 1 0 1 1 1 1 1 1 1 1 1
```

Note that the object gets bigger and holes are filled.

`>> help imdilate`

Source: K. Grauman
2D example for dilation

(a) Binary image $B$

(b) Structuring element $S$

(c) Dilation $B \oplus S$

Source: Shapiro and Stockman
Dilation vs. Erosion

At each position:

- **Dilation**: if current pixel is foreground, OR the structuring element with the input image.
- **Erosion**: if every pixel under the structuring element’s nonzero entries is foreground, OR the current pixel with S.
Example for Erosion (1D)

Input image:

```
1 0 0 0 1 1 1 1 0 1 1 1
```

Structuring Element:

```
1 1 1
```

Output Image:

```
0
```

\[
g(x) = f(x) \Theta SE
\]
Example for Erosion (1D)

Input image: 1 0 0 0 1 1 1 1 0 1 1 1

Structuring Element: 1 1 1 1

Output Image: 0 0

\[ g(x) = f(x) \ominus SE \]

Source: K. Grauman
Example for Erosion

Input image

1 0 0 0 1 1 1 1 1 0 1 1

Structuring Element

1 1 1 1

Output Image

0 0 0
Example for Erosion

Input image

1 0 0 0 1 1 1 1 0 1 1

Structuring Element

1 1 1 1

Output Image

0 0 0 0 0 0 0 0 0 0 0

Source: K. Grauman
Example for Erosion

Input image:

```
1 0 0 0 1 1 1 1 0 1 1 1
```

Structuring Element:

```
1 1 1 1
```

Output Image:

```
0 0 0 0 0 0
```
Example for Erosion

Input image

```
1 0 0 0 1 1 1 1 0 1 1
```

Structuring Element

```
1 1 1 1
```

Output Image

```
0 0 0 0 0 0 0 0 1
```
Example for Erosion

Input image

Structuring Element

Output Image

Source: K. Grauman
Example for Erosion

Input image:

```
1 0 0 0 1 1 1 1 0 1 1 1
```

Structuring Element:

```
  1 1 1
```

Output Image:

```
0 0 0 0 0 0 1 0 0 0
```
Example for Erosion

Input image

| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

Structuring Element

| 1 | 1 | 1 | 1 |

Output Image

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
Example for Erosion

Input image

1 0 0 0 1 1 1 1 0 1 1

Structuring Element

1 1 1

Output Image

0 0 0 0 0 0 1 1 0 0 0 0 0 1

Note that the object gets smaller

>> help imerode

Source: K. Grauman
2D example for erosion

(a) Binary image $B$

(b) Structuring element $S$

(d) Erosion $B \ominus S$
Opening

• Erode, then dilate
• Remove small objects, keep original shape

Before opening

After opening

Source: K. Grauman
Closing

- Dilate, then erode
- Fill holes, but keep original shape

Before closing

After closing

Applet: [http://bigwww.epfl.ch/demo/jmorpho/start.php](http://bigwww.epfl.ch/demo/jmorpho/start.php)

Source: K. Grauman
Summary

• Can use filters to process and describe local neighborhood
  – Derivatives to locate gradients
  – Convolution properties will influence efficiency
• Edge detection processes the image gradient to find curves
• Clean up thresholding outputs with binary image morphology operations

Adapted from K. Grauman
Next time

• Computing image features: Detecting and describing keypoints