Image gradients and edges

Prof. Xin Yang
HUST
Edge detection

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

- **Main idea**: look for strong gradients

- **Applications**?
What causes an edge?

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

Kristen Grauman, UT-Austin
Edges/gradients and invariance
Derivatives and edges

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

Source: L. Lazebnik
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

\[ \frac{\partial f(x, y)}{\partial x} \quad \text{or} \quad \frac{\partial f(x, y)}{\partial y} \]

Which shows changes with respect to \( x \)?

(showing filters for correlation)
Assorted finite difference filters

\[
\begin{align*}
\text{Prewitt:} & \quad M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad ; \quad M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \\
\text{Sobel:} & \quad M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad ; \quad M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \\
\text{Roberts:} & \quad M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad ; \quad M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{My} & = \text{fspecial('sobel')} \\
\text{outim} & = \text{imfilter(double(im), My)} \\
\text{imagesc(outim)} \\
\text{colormap gray}
\end{align*}
\]
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 change in intensity

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

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

The gradient direction (orientation of edge normal) is given by:

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

Slide credit Steve Seitz
Solution: smooth first

Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \ast f)$
Derivative theorem of convolution

\[ \frac{\partial}{\partial x} (h \ast f) = (\frac{\partial}{\partial x} h) \ast f \]

Differentiation property of convolution.
Derivative of Gaussian filters

x-direction

y-direction

Source: L. Lazebnik
Laplacian of Gaussian

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

Where is the edge? Zero-crossings of bottom graph

Slide credit: Steve Seitz
2D edge detection filters

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

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

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

- \( \nabla^2 \) is the Laplacian operator:

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

Slide credit: Steve Seitz
Smoothing with a Gaussian

Recall: parameter $\sigma$ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.
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.
Application: Seam Carving for Image Regarding

[Shai & Avidan, SIGGRAPH 2007]
Seam carving: main idea

[Shai & Avidan, SIGGRAPH 2007]
Seam carving: main idea
Seam carving: main idea

Intuition:

• Preserve the most “interesting” content
  → Prefer to remove pixels with low gradient energy
• To reduce or increase size in one dimension, remove irregularly shaped “seams”
  → Optimal solution via dynamic programming.
Seam carving: main idea

Energy\( (f) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}\)

- Want to remove seams where they won’t be very noticeable:
  - Measure “energy” as gradient magnitude
- Choose seam based on **minimum total energy path** across image, subject to 8-connectedness.
Let a vertical seam \( s \) consist of \( h \) positions that form an 8-connected path.

Let the cost of a seam be: \( \text{Cost}(s) = \sum_{i=1}^{h} \text{Energy}(f(s_i)) \)

Optimal seam minimizes this cost: \( s^* = \min_s \text{Cost}(s) \)

Compute it efficiently with dynamic programming.

\[
\text{Energy}(f) = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}
\]
How to identify the minimum cost seam?

- First, consider a **greedy** approach:

```
Energy matrix (gradient magnitude)
```

- Kristen Grauman, UT-Austin
Seam carving: algorithm

- Compute the cumulative minimum energy for all possible connected seams at each entry \((i,j)\):

\[
M(i, j) = \text{Energy}(i, j) + \min(M(i-1, j-1), M(i-1, j), M(i-1, j+1))
\]

- Then, min value in last row of \(M\) indicates end of the minimal connected vertical seam.
- Backtrack up from there, selecting min of 3 above in \(M\).
Example

\[ M(i, j) = \text{Energy}(i, j) + \min(M(i-1, j-1), M(i-1, j), M(i-1, j+1)) \]
Example

\[ M(i, j) = \text{Energy}(i, j) + \min(M(i-1, j-1), M(i-1, j), M(i-1, j+1)) \]
Real image example

Original Image

Energy Map

Blue = low energy
Red = high energy
Real image example
Other notes on seam carving

• Analogous procedure for horizontal seams
• Can also insert seams to *increase* size of image in either dimension
  – Duplicate optimal seam, averaged with neighbors
• Other energy functions may be plugged in
  – E.g., color-based, interactive,…
• Can use combination of vertical and horizontal seams
Example results from prior classes

(a) Original input

(b) Content-aware resizing

(c) Image from ‘imresize’
Results from Suyog Jain
Results from Martin Becker
Results from Jay Hennig

Original image (599 by 799)

Conventional resize (399 by 599)

Seam carving (399 by 599)
Removal of a marked object

(a) Selected an area.

(b) Object is removed.

(c) Selected an area.

(d) Object is removed.

Results from Donghyuk Shin
Removal of a marked object

Results from Eunho Yang
“Failure cases” with seam carving

By Donghyuk Shin
“Failure cases” with seam carving

By Suyog Jain
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
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)
Thresholding gradient with a lower threshold
Thresholding gradient with a higher threshold
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 Canny edge detector

original image (Lena)
The Canny edge detector

norm of the gradient
The Canny edge detector

thresholding
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

thinning
(non-maximum suppression)
Hysteresis thresholding

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

Source: Steve 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
Low-level edges vs. perceived contours

Background

Texture

Shadows

Kristen Grauman, UT-Austin
Low-level edges vs. perceived contours

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

Source: L. Lazebnik
Learn from humans which combination of features is most indicative of a “good” contour?
Summary

• Filters allow local image neighborhood to influence our description and features
  – Smoothing to reduce noise
  – Derivatives to locate contrast, gradient

• Convolution properties will influence the efficiency with which we can process images.
  – Associative
  – Filter separability

• Edge detection processes the image gradient to find curves, or chains of edgels.