אתויר - סף – אם הפוך זרבירס הירוק ממחנה?
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
What can cause an edge?

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

- Edges look like steep cliffs

Source: S. Seitz
Derivatives and edges

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

Source: L. Lazebnik
Differentiation and 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?
Side note: Filters and Convolutions

• First, consider a signal in 1D...
• Let’s replace each pixel with an average of all the values in its neighborhood
• Moving average in 1D:

Source: S. Marschner
Weighted Moving Average

• Can add weights to our moving average
• \textit{Weights} $[1, 1, 1, 1, 1] / 5$

Source: S. Marschner
Weighted Moving Average

- Non-uniform weights \([1, 4, 6, 4, 1] / 16\)

Source: S. Marschner
Moving Average in 2D

\[ F[x, y] \]  

\[ G[x, y] \]
Moving Average In 2D

$F\left[ x, y \right]$  

$G\left[ x, y \right]$  

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]
Correlation filtering

Say the averaging window size is $2k+1 \times 2k+1$:

$$G[i, j] = \frac{1}{(2k + 1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i + u, j + v]$$

Attribute uniform weight Loop over all pixels in neighborhood around to each pixel image pixel $F[i,j]$

Now generalize to allow different weights depending on neighboring pixel’s relative position:

$$G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i + u, j + v]$$

Non-uniform weights
Correlation filtering

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v] \]

This is called cross-correlation, denoted \( G = H \otimes F \)

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter “kernel” or “mask” \( H[u, v] \) is the prescription for the weights in the linear combination.
Convolution

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i - u, j - v]
\]

\[
G = H \star F
\]

*Notation for convolution operator*
Convolution vs. correlation

Convolution

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i - u, j - v] \]

\[ G = H \ast F \]

Cross-correlation

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v] \]

\[ G = H \otimes F \]

Back to our question: To implement the derivatives, what would be the associated filter?

\[ \frac{\partial f(x, y)}{\partial x} \approx \frac{f(x+1, y) - f(x, y)}{1} \]
Partial derivatives of an image

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

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

\[ \begin{array}{c|c}
-1 & 1 \\
\end{array} \]
Assorted finite difference filters

\[ M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \]

\[ M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \]

Prewitt:

\[ M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \]

\[ M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \]

Sobel:

\[ M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \]

\[ M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \]

Roberts:

\[ \text{My} = \text{fspecial}('\text{sobel}'); \]
\[ \text{outim} = \text{imfilter}(	ext{double(im)}, \text{My}); \]
\[ \text{imagesc(outim)}; \]
\[ \text{colormap gray}; \]
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

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

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

The gradient direction (orientation of edge normal) 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

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

A simple edge detector using gradient magnitude

- Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters

- Compute gradient magnitude at each pixel

- If magnitude at a pixel exceeds a threshold, report a possible edge point.
Compute Spatial Image Gradients

\[ \frac{I(x+1,y) - I(x-1,y)}{2} \]

Partial derivative wrt x

\[ \frac{I(x,y+1) - I(x,y-1)}{2} \]

Partial derivative wrt y

Replace with your favorite smoothing+derivative operator

\[ I_x = \frac{\partial I(x,y)}{\partial x} \]

\[ I_y = \frac{\partial I(x,y)}{\partial y} \]
Simple Edge Detection Using Gradients

A simple edge detector using gradient magnitude

• Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters

• Compute gradient magnitude at each pixel

• If magnitude at a pixel exceeds a threshold, report a possible edge point.
Compute Gradient Magnitude

Magnitude of gradient
\[ \sqrt{I_x^2 + I_y^2} \]

Measures steepness of slope at each pixel
(= edge contrast)
Simple Edge Detection Using Gradients

A simple edge detector using gradient magnitude

• Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters

• Compute gradient magnitude at each pixel

• If magnitude at a pixel exceeds a threshold, report a possible edge point.
Threshold to Find Edge Pixels

- Example – cont.: Binary edge image

  Threshold
  Mag > 30
Issues to Address

How should we choose the threshold?

> 10
> 30
> 80
Issues to Address

Edge thinning and linking

smoothing + thresholding gives us a binary mask with “thick” edges

we want thin, one-pixel wide, connected contours
Another issue: The effects of noise

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

\[ f(x) \]

Where is the edge?
Solution: smooth first

Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \ast f)$
Smoothing with a Gaussian

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.

Too fine of a scale...can’t see the forest for the trees.
Too coarse of a scale...can’t tell the maple grain from the cherry.
Canny Edge Detector

An important case study

Probably, the most used edge detection algorithm by C.V. practitioners

Experiments consistently show that it performs very well

Recall: Practical Issues for Edge Detection

Thinning and linking

Choosing a magnitude threshold

Canny has good answers to all!
Thinning

note: do thinning before thresholding!

We want to mark points along curve where the magnitude is largest.

We can do this by looking for a maximum along a 1D intensity slice normal to the curve (non-maximum supression).

These points should form a one-pixel wide curve.
problem:

- If the threshold is too high:
  - Very few (none) edges
  - High MISDETECTIONS, many gaps
- If the threshold is too low:
  - Too many (all pixels) edges
  - High FALSE POSITIVES, many extra edges
SOLUTION: Hysteresis Thresholding

Allows us to apply both! (e.g. a “fuzzy” threshold)

• Keep both a high threshold \( H \) and a low threshold \( L \).
• Any edges with strength \( < L \) are discarded.
• Any edge with strength \( > H \) are kept.
• An edge \( P \) with strength between \( L \) and \( H \) is kept only if there is a path of edges with strength \( > L \) connecting \( P \) to an edge of strength \( > H \).

• In practice, this thresholding is combined with edge linking to get connected contours
Example of Hysteresis Thresholding

Hysteresis thresholding

T=15

T=5

Hysteresis
$T_h=15 \quad T_l = 5$
Complete Canny Algorithm

1. Compute $x$ and $y$ derivatives of image
   \[ I_x = G_{\sigma}^x * I \quad I_y = G_{\sigma}^y * I \]

2. Compute magnitude of gradient at every pixel
   \[ M(x, y) = |\nabla I| = \sqrt{I_x^2 + I_y^2} \]

3. Eliminate those pixels that are not local maxima of the magnitude in the direction of the gradient

4. Hysteresis Thresholding
   - Select the pixels such that $M > T_h$ (high threshold)
   - Collect the pixels such that $M > T_l$ (low threshold) that are neighbors of already collected edge points
Edge detection is just the beginning...

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

Much more on segmentation later...

Source: L. Lazebnik