Edges & Edge Detection

Baba Vemuri
Professor, Dept. of CISE, UFL
Contour Line Drawing
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)

Source: D. Lowe
Origin of edges

- Edges are caused by a variety of factors:
  - Depth discontinuity
  - Surface color discontinuity
  - Illumination discontinuity
  - Surface normal discontinuity

Source: Steve Seitz
Edges in the Visual Cortex

- Extract compact, generic, representation of image that carries sufficient information for higher-level processing tasks.

- Essentially what the area V1 in our visual cortex does.
Visual Cortex

http://www.studyblue.com

http://wiki.bethanycrane.com/introducingtheeye
The gradient points in the direction of most rapid increase in intensity.

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

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

Source: Steve Seitz
Differentiation and convolution

• Recall, for 2D function, \( f(x,y) \):

\[
\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0} \left( \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon} \right)
\]

• This is linear and shift invariant, so must be the result of a convolution.

• We could approximate this as

\[
\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}
\]

(which is obviously a convolution)

\[
\begin{array}{cc}
-1 & 1
\end{array}
\]

Source: D. Forsyth, D. Lowe
Finite difference filters

Other approximations of derivative filters exist:

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

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

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

Source: K. Grauman
Finite differences: example

- Which one is the gradient in the x-direction (resp. y-direction)?
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? Note, derivative amplifies noise

Source: S. Seitz
Differentiation is ill-posed

- Let $f(x)$ be perturbed by a small noise term $f(x) + e\sin(wx) = g(x)$.
- $g(x)$ and $f(x)$ can be made arbitrarily close to each other for small ‘$e$’, but derivatives are very far for large ‘$w$’.
- Characterizing intensity changes requires evaluation of derivatives of image intensity.
- Hence, a problem in Numerical Differentiation which is ill-posed.
Ill-posed Problem

• A mathematical problem is said to be ill-posed when its solution
  – Does not exist
  – Is not unique
  – Does not depend continuously on the initial data
    i.e., the solution is not robust in the presence of noise.
Differentiation is Ill-posed

- Differentiation of a function $f(x)$ is a typical ill-posed problem since it can be seen as the solution to the inverse problem $g(x) = Af(x)$.
- $g(x) = y$ is data, $Af(x)$ is an integral operator, $f(x) = z$ is the unknown. Hence, finding $z$, given $y$.
- Solution: Regularization
Regularization Techniques

• Find $z$, given the data $y$ such that, $Az = y$
  – Requires a suitable choice of a $||.||$ and a stabilizing functional $||Pz||$
  – Choice is dictated by a combination of mathematical and physical considerations.
  – There are 3 methods of regularization.
Regularization (contd.)

• Among $z$ that satisfy $||Pz|| \leq C_1$ (a constant), find $z$ that minimizes $||Az-y||$. Finds a $z$ that satisfies the constraint and best approximates the data.

• Among $z$ that satisfy $||Az-y|| \leq C_2$ (a constant) find $z$ that minimizes $||Pz||$. Finds a $z$ that is close to the data and is most regular.

• Find $z$ that minimizes: $||Az-y||^2 + \lambda ||Pz||^2$.
  First term captures closeness to data, second term captures degree of regularization/smoothness of the solution (embodies the physical constraints necessary to solve the problem). $\Lambda$ controls the compromise between the two terms.
Regularizing with Splines

• Simplest choice for $P=d^2/dx^2$ and the usual $L_2$-norm. This choice corresponds to an appropriate choice of smoothness on the intensity function.

• Physical justification: Noiseless image is smooth, all derivatives have to exist and be bounded because the image is band-limited.

• Given, $\{x_k\}$ and $f(x_k) = f_k$ for all $k=1,..,n$, finding $f'(x_k)$ is ill-posed.
Spline Interpolation

• Applying second regularizing method with stabilizing operator $P = \frac{d^2}{dx^2}$ and the $L_2$-norm to search for a function $S(x)$ such that,

(a) $S(x_k) = f_k$ and 
(b) $||Ps(x)||$ is minimized i.e., $\int |S''(x)|^2 dx$ is minimized.

• The solution is a cubic spline $S(x)$ that interpolates $f(x)$ in the given interval. Hence, $f'_k = S'(x_k)$. 
Smoothing Spline

• For inexact data, use the 3\textsuperscript{rd} regularizing technique.

• Find $S(x)$ such that,

$$\sum (f_k - S(x_k))^2 + \lambda \int |S(x)|^2 \, dx$$

is minimized. The solution is a smoothing cubic spline.
Effects of noise

- Finite 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 is to be done?
  - Smoothing the image should help, by forcing pixels different from their neighbors (=noise pixels?) to look more like neighbors

Source: D. Forsyth
Solution: smooth first

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

Source: S. Seitz
Derivative theorem of convolution

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

Source: S. Seitz
Derivative of Gaussian filter

- Which one finds horizontal/vertical edges?
Laplacian of Gaussian

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

\[
\frac{\partial^2}{\partial x^2} h
\]

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

- Where is the edge? • Zero-crossings of bottom graph
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:

\[ \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \]
Smoothed derivative removes noise, but blurs edge. Also finds edges at different scales.

Source: D. Forsyth
Implementation issues

- The gradient magnitude is large along a thick "trail" or "ridge," so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Source: D. Forsyth
Designing an edge detector

• Criteria for an “optimal” edge detector:
  – **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  – **Good localization:** the edges detected must be as close as possible to the true edges
  – **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge

Source: L. Fei-Fei
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
• MATLAB: edge(image, ‘canny’)


Source: L. Fei-Fei
Canny edge detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” down to single pixel width

Source: D. Lowe, L. Fei-Fei
Non-maximum suppression

At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

Source: D. Forsyth
Example

• original image (Lena)
Example

norm of the gradient
Example

thresholding
Example

Non-maximum suppression
Canny edge detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
   - Thin multi-pixel wide “ridges” down to single pixel width
4. Linking of edge points

Source: D. Lowe, L. Fei-Fei
Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).

Source: D. Forsyth
Canny edge detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
   - Thin multi-pixel wide “ridges” down to single pixel width
4. Linking of edge points
   - Hysteresis thresholding: use a higher threshold to start edge curves and a lower threshold to continue them

Source: D. Lowe, L. Fei-Fei
Hysteresis thresholding

- Use a high threshold to start edge curves and a low threshold to continue them
  - Reduces drop-outs

Source: S. Seitz
Hysteresis thresholding

original image

high threshold (strong edges)
low threshold (weak edges)
hysteresis threshold

Source: L. Fei-Fei
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

Source: S. Seitz
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/)
Antialiasing

What can be done?

1. Raise sampling rate by *oversampling*
   - *Sample at k times the resolution*

2. Lower the max frequency by *prefiltering*
   - Smooth the signal enough
   - Works on discrete signals
   - Especially useful in multi-resolution pyramids
Multi-resolution Image Pyramids
Up-sampling

Expand Image

Interpolation
Down- or sub-sampling

two-stage resampling using a separable filter
Constructing a pyramid by taking every second pixel leads to layers that badly misrepresent the top layer.
Constructing a pyramid by taking every second pixel leads to layers that badly misrepresent the top layer.

Solution?
The Gaussian Pyramid

\[ G_4 = (G_3 \ast \text{gaussian}) \downarrow 2 \]
\[ G_3 = (G_2 \ast \text{gaussian}) \downarrow 2 \]
\[ G_2 = (G_1 \ast \text{gaussian}) \downarrow 2 \]
\[ G_1 = (G_0 \ast \text{gaussian}) \downarrow 2 \]
\[ G_0 = \text{Image} \]
The Gaussian Pyramid

\[ G_4 = (G_3 \ast \text{gaussian}) \downarrow 2 \]
\[ G_3 = (G_2 \ast \text{gaussian}) \downarrow 2 \]
\[ G_2 = (G_1 \ast \text{gaussian}) \downarrow 2 \]
\[ G_1 = (G_0 \ast \text{gaussian}) \downarrow 2 \]

\( G_0 \) = Image blur

Sub-sample
The Gaussian Pyramid

\[ G_4 = (G_3 \ast \text{gaussian}) \downarrow 2 \]
\[ G_3 = (G_2 \ast \text{gaussian}) \downarrow 2 \]
\[ G_2 = (G_1 \ast \text{gaussian}) \downarrow 2 \]
\[ G_1 = (G_0 \ast \text{gaussian}) \downarrow 2 \]

\( G_0 = \text{Image} \)
The Gaussian Pyramid

\[ G_4 = (G_3 \ast \text{gaussian}) \downarrow 2 \]
\[ G_3 = (G_2 \ast \text{gaussian}) \downarrow 2 \]
\[ G_2 = (G_1 \ast \text{gaussian}) \downarrow 2 \]
\[ G_1 = (G_0 \ast \text{gaussian}) \downarrow 2 \]

\[ G_0 = \text{Image} \]
Gaussian pre-filtering

- Solution: filter the image, *then* subsample
Subsampling with Gaussian pre-filtering

Gaussian 1/2  
G 1/4  
G 1/8

• Solution: filter the image, \textit{then} subsample
Compare with...

1/2

1/4 (2x zoom)

1/8 (4x zoom)
Fun with Gaussian Smoothing

by Charles Allen Gillbert

by Harmon & Julesz

http://www.michaelbach.de/ot/cog_bl
Fun with Gaussian Smoothing
The Laplacian Pyramid

Gaussian Pyramid

\[ L_i = G_i - \text{expand}(G_{i+1}) \]

\[ G_i = L_i + \text{expand}(G_{i+1}) \]

Laplacian Pyramid

\[ L_n = G_n \]
Applications of Image Pyramids

- Coarse-to-Fine or Multi resolution strategies for computational efficiency.
- Search for correspondence
  - look at coarse scales, then refine with finer scales
- Edge tracking
  - a “good” edge at a fine scale has parents at a coarser scale
- Important in texture representation
- Image Blending and Mosaicing