Concepts in

Edge Detection

Dr. Sukhendu Das
Deptt. of Computer Science and Engg.,
Indian Institute of Technology, Madras
Chennai – 600036, India.

http://www.cs.iitm.ernet.in/~sdas
Email: sdas@iitm.ac.in
**Edge Detection**

Edge is a boundary between two homogeneous regions. The gray level properties of the two regions on either side of an edge are distinct and exhibit some local uniformity or homogeneity among themselves.

An edge is typically extracted by computing the derivative of the image intensity function. This consists of two parts:

- **Magnitude of the derivative**: measure of the strength/contrast of the edge
- **Direction of the derivative vector**: edge orientation

![Diagram of Edge Detection](image-url)
Computing the derivative: Finite difference in 1-D

\[ \frac{df}{dx} \approx \frac{f(x + dx) - f(x)}{dx} \approx \frac{f(x + dx) - f(x - dx)}{2dx} \]

\[ \frac{d^2 f}{d^2x} \approx \frac{f(x + dx) - 2f(x) + f(x - dx)}{dx^2} \]
Computing the derivative: Finite differences in 2-D

\[
\frac{\partial f}{\partial x} \approx \frac{f(x + dx, y) - f(x, y)}{dx} \\
\frac{\partial f}{\partial y} \approx \frac{f(x, y + dy) - f(x, y)}{dy}
\]

\[
\frac{f(x + dx, dy) - f(x - dx, dy)}{2dx} \\
\frac{f(x, y + dy) - f(x, y - dy)}{2dy}
\]
Differentiation using convolution:

\[ \frac{\delta f}{\delta x} = [-1 \ 1]; \quad \frac{\delta f}{\delta y} = [-1 \ 1]^{T}; \]

\[ \frac{\delta^2 f}{\delta x^2} = [1 \ -2 \ 1]; \quad \frac{\delta^2 f}{\delta y^2} = [1 \ -2 \ 1]^{T}; \]

Need to use wider masks to add an element of smoothing and better response. The traditional derivative operators used were:

**Roberts**

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

**Laplacian**

\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{bmatrix}
\]

**Prewitt**

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

**Sobel**

\[
\begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2, \\
-1 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & 2 & 1
\end{bmatrix}
\]
Two components of the edge values computed are:

Gradient values: \( G_x = \frac{\partial f}{\partial x}; G_y = \frac{\partial f}{\partial y} \).

The **magnitude** of the edge is calculated as:

\[
|G| = \left(G_x^2 + G_y^2\right)^{1/2}
\]

and **orientation** as:

\[
\theta = \arctan\left(\frac{G_y}{G_x}\right)
\]

Most of these partial derivative operators are sensitive to noise. Use of these masks resulted in thick edges or boundaries, in addition to spurious edge pixels due to noise.

Laplacian mask is highly sensitive to spike noise. Use of noise smoothing became mandatory before edge detection, specifically for noisy images. But noise smoothing, typically by the use of a **Gaussian** function, caused a blurring or smearing of the edge information or gradient values.
A Gaussian function is shown below. The width of the Gaussian depends on the variance $\sigma$. The value of $\sigma$ dictates the amount of smoothing. The expression of the Gaussian function is given as:

$$g(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}}$$

Marr and Hildreth (1980) suggested the use of the “Laplacian of the Gaussian” (LOG) operator to detect edges. This produced edges as Zero-Crossings (ZC’s) in the output function - why??

However, it did not give any idea of the gradient magnitude or orientation of the edges. But ZC’s were spread through-out an image. How do one detect true edges from ZC’s??
LOG operator in 1-D

LOG operator in 2-D
First Derivative

Ideal Step Edge

G: Gaussian Function

Smoothed Step Edge

First Derivative
\( \delta F \)

\( \delta^2 F \)
Canny in 1986 suggested an optimal operator, which uses the Gaussian smoothing and the derivative function together. He proved that the first derivative of the Gaussian function, as shown below, is a good approximation to his optimal operator.

It combines both the derivative and smoothing properties in a nice way to obtain good edges. Canny also talks of a hysteresis based thresholding strategy for marking the edges from the gradient values.

Smoothing and derivative when applied separately, were not producing good results under noisy conditions. This is because, one opposes the other. Whereas, when combined together produces the desired output.

Expression of Canny (1-D operator is):

\[ c(y) = g'(y) = \left( \frac{-y}{\sqrt{2\pi}\sigma^3} \right) \exp\left( \frac{-y^2}{2\sigma^2} \right) \]
Canny’s algorithm for edge detection:

Detect an edge, where simultaneously the following conditions are satisfied:

\[ \nabla^2 G*f = 0 \text{ and } \nabla G*f \text{ reaches a maximum.} \]

\( \nabla G \) is the first derivative of the Gaussian defined (in 1-D) as:

\[
\nabla G(x) = \frac{-x}{\sqrt{2\pi}\sigma^3} \exp\left(-\frac{x^2}{2\sigma^2}\right)
\]

and

\( \nabla^2 G \) in two-dimension is given by (also known as the “Laplacian of the Gaussian” or LOG operator):

\[
\nabla^2 G(r) = \left(\frac{1}{\pi\sigma^4}\right)(r^2/2 \sigma^2 - 1) \exp\left(-\frac{r^2}{2\sigma^2}\right)
\]
\[ g(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}} \]

\[ \nabla G(x) = \frac{-x}{\sqrt{2\pi\sigma^2}} \exp\left( -\frac{x^2}{2\sigma^2} \right) \]

\[ \nabla^2 G(x) = \frac{-\left(\frac{x}{\sigma}\right)^2 - 1}{\sqrt{2\pi\sigma^2}} \exp\left( -\frac{x^2}{2\sigma^2} \right) \]
\( G: \) Gaussian Function

\( \delta G \)

Noisy Edge, \( Sn \)

\( \delta G * Sn \)
\[ \delta G \]

\[ \delta^2 G \]

\[ \text{Noisy Edge, Sn} \]

\[ \delta G * \text{Sn} \]

\[ \delta^2 G * \text{Sn} \]
Three criteria in the optimization function used by Canny, for deriving the operator:

- **Localization, Detection and minimal response** (SNR-based).

The three-four stages of Canny’s algorithm:

- Apply Operator (often implemented as Smoothing then Derivative)

- Apply non-maximal suppression

- Apply hysteresis based (linking and) thresholding

Read about - DERICHE recursive filtering
Non-maximum suppression:
Select the single maximum point across the width of an edge.
Graphical Interpretation of non-maximal suppression

Wide ridges around the local maxima (large values around the edges)
NONMAX_SUPPRESSION (Mag, Dir)

- Consider 4 directions \( \text{Del}^+ = \{(1,0),(1,1),(0,1),(-1,1)\} \)
  \( \text{Del}^- = \{(-1,0),(-1,-1),(0,-1),(1,1)\} \)

- For each pixel \((i,j)\) do:
  1. Find the direction of gradient (normal to the edge)
     \( d = (\text{Dir}(i,j) + \pi/8) \mod \pi/4 \)
  2. If \( \text{Mag}(i,j) \) is smaller than at least one of its neigh. along \( d \) then
     \( I_N(i,j) = 0 \), otherwise, \( I_N(i,j) = \text{Mag}(i,j) \)
     If \( \text{Mag}(i,j) < \text{Mag}((i,j) + \text{Del}^+(d)) \) then \( I_N(i,j) = 0 \)
     Else If \( \text{Mag}(i,j) < \text{Mag}((i,j) + \text{Del}^-(d)) \) then \( I_N(i,j) = 0 \)
     Else \( I_N = \text{Mag}(i,j) \)

- The output is the thinned edge image \( I_N \)
Original Image, Presmoothed Image, Gradient Image, Non-maximum Suppressed Image, Final Result
Figure 3: Discrete approximation to Gaussian function with $\sigma=1.4$.

http://www-scf.usc.edu/~boqinggo/Canny.htm
Canny Edge Detection (Example)

Original image

Strong edges only

Strong + connected weak edges

Weak edges

courtesy of G. Loy
Examples of using Deriche filter on various source images

<table>
<thead>
<tr>
<th>Source image</th>
<th>Filtered image</th>
<th>Filter parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunflower</td>
<td>α = 1.5</td>
<td>α = 1.5</td>
</tr>
<tr>
<td></td>
<td>low treshold = 20</td>
<td>low treshold = 20</td>
</tr>
<tr>
<td></td>
<td>high treshold = 40</td>
<td>high treshold = 40</td>
</tr>
<tr>
<td>Checkerboard</td>
<td>α = 4.0</td>
<td>α = 4.0</td>
</tr>
<tr>
<td></td>
<td>low treshold = 50</td>
<td>low treshold = 50</td>
</tr>
<tr>
<td></td>
<td>high treshold = 90</td>
<td>high treshold = 90</td>
</tr>
<tr>
<td>Clipper</td>
<td>α = 0.8</td>
<td>α = 0.8</td>
</tr>
<tr>
<td></td>
<td>low treshold = 26</td>
<td>low treshold = 26</td>
</tr>
<tr>
<td></td>
<td>high treshold = 41</td>
<td>high treshold = 41</td>
</tr>
<tr>
<td>Leaf</td>
<td>α = 1.0</td>
<td>α = 1.0</td>
</tr>
<tr>
<td></td>
<td>low treshold = 15</td>
<td>low treshold = 15</td>
</tr>
<tr>
<td></td>
<td>high treshold = 35</td>
<td>high treshold = 35</td>
</tr>
</tbody>
</table>
Effect of threshold

\[ T_{\text{high}} = 255 \quad T_{\text{low}} = 1 \quad \sigma = 1 \]

\[ T_{\text{high}} = 255 \quad T_{\text{low}} = 220 \quad \sigma = 1 \]
Effect of threshold and of $\sigma$ (Gaussian kernel size)

- $\sigma = 1$
  - $T_{\text{high}} = 120$  $T_{\text{low}} = 1$
- $\sigma = 2$
  - $T_{\text{high}} = 120$  $T_{\text{low}} = 1$

original
Multi-scale Edge detection
Problem definition

- Our goal is to simultaneously extract edges of all lengths
- Edges are well localized across the scale-space
\[ \sigma = 0.5 \]

\[ \sigma = 1.5 \]

\[ \sigma = 1 \]

\[ \sigma = 2 \]

2-D Canny edge maps
LOG with increasing SIGMA
Motivation

• A step edge is sensed at various points by cells of the retinal array

• Real-world objects are composed of different structures at different scales

• Connectivity of an object depends on the scale at which it is observed

• In real-world images the edges may not be ideal

• Variation of the response over different scales is important
Optimal Edge Detection in Two-Dimensional Images
Compute the gradient map of Gaussian blurred image

Assign the magnitude of the gradient as edge strength to all edge pixels

Edge strength is equalized (HEQ) to full scale of intensity

Compute the histogram for the edge segment strengths

Fit a Gaussian to the low intensity part of the histogram and compute three threshold (Low, medium and high) based on mean and variance of Gaussian

Compute edge subsegments and compute NESS for each subsegment

NES > MT

NESS > LT

NESS > HT

Salient edge Map

NES: Normalized Edge Strength
NESS: Normalized Edge Strength of sub-segment
LT: Low Threshold
HT: High threshold
MT: Medium Threshold
Histogram of the normalised edge strengths and fitted Gaussian distribution

$\sigma=0.5$

$\sigma=1.5$
Salient Edge maps

2-D Canny edge map

\[ \sigma = 0.5 \quad \sigma = 1 \quad \sigma = 1.5 \quad \sigma = 2 \]
Combining different scales

- The combination procedure checks if there are new salient edges in the detection results from larger scales

**Algorithm**

1. Minmap, maxmap = edge map of smallest scale
2. Compare maxmap with second smallest scale edgemap
3. If an edge segment of minimum length from second smallest scale does not appear in maxmap, add that particular segment to minmap
4. Repeat step 2 and step 3 with various scales
5. Minmap is the final combined scale output
Scale space combination
2-D Canny edge model

Scale space combination
of Qian & Huang
edge model

Lena
256x256
Read about:
- Hysteresis based Thresholding
- Non-maximal suppression
- Edge Linking & Thinning
- Edge preserving enhancement or super-resolution
- Contour Tracing
- Level set based or differential geometry based analysis
- Edge detection with sub-pixel accuracy
- Neuro-fuzzy models for optimal edge detection
- **Phase Congruency** model (Peter Kovesi) for edge detection
- **Deriche model** for optimal/recursive filtering
- Neural model for supervised edge detection
- Physics based processing
- Optimization based (MRF, HMM) edge detection, in presence of noise and blur
- Multi-channel edge detection
- Berkeley Edge Detection/segmentation
- **Structured Forests**
REFERENCES


A Two-Dimensional Edge Detection Scheme for General Visual Processing, Qian, R.J. and Huang, T.S, ICPR-94, YEAR = "1994", "595-598".

