Additive noise

- $I = S + N$. Noise doesn’t depend on signal.
- We’ll consider:
  - $I_i = s_i + n_i$ with $E(n_i) = 0$
  - $s_i$ deterministic. $n_i$ a random var.
  - $n_i, n_j$ independent for $i \neq j$
  - $n_i, n_j$ identically distributed

Box vs. Gaussian: Smoothing by Averaging

Box vs. Gaussian: Smoothing with a Gaussian
The effects of smoothing
Each row shows smoothing
With Gaussians of different width; each column shows
different realizations of
an image of Gaussian noise.

Gaussian Smoothing

Gaussian Smoothing

by Charles Allen Gillbert

by Harmon & Julesz

http://www.michaelbach.de/ot/cog_blureffects/index.html

Efficient Implementations

Both, the BOX filter and the Gaussian
filter are separable:
– First convolve each row with a 1-D filter
– Then convolve each column with a 1-D
filter.

For Gaussian kernels \( g_1(x) \) and \( g_2(x) \),
• If \( g_1 \) & \( g_2 \) respectively have variance \( \sigma_1^2 \) & \( \sigma_2^2 \)
• Then \( g_1 * g_2 \) has variance \( \sigma_1^2 + \sigma_2^2 \)

Edge Detection

and

Corner Detection
**Edges**

What is an edge?
A discontinuity in image intensity.

Physical causes of edges
1. Object boundaries
2. Surface normal discontinuities
3. Reflectance (albedo) discontinuities
4. Lighting discontinuities

**Object Boundaries**

**Surface normal discontinuities**

**Boundaries of materials properties**

**Boundaries of lighting**

**Noisy Step Edge**
- Derivative is high everywhere.
- Must smooth before taking gradient.
Edge is Where Change Occurs: 1-D
• Change is measured by derivative in 1D
  - Ideal Edge
  - Smoothed Edge
  - First Derivative
  - Second Derivative
• Biggest change, derivative has maximum magnitude
• Or 2nd derivative is zero.

Implementing 1-D Edge Detection
1. Filter out noise: convolve with Gaussian
2. Take a derivative: convolve with [-1 0 1]
   - We can combine 1 and 2.
3. Find the peak: Two issues:
   - Should be a local maximum.
   - Should be sufficiently high.

Numerical Derivatives
Take Taylor series expansion of f(x) about x₀:
\[ f(x) = f(x₀) + f'(x₀)(x-x₀) + \frac{1}{2} f''(x₀)(x-x₀)^2 + \ldots \]
Consider samples taken at increments of h and first two terms of the expansion, we have:
\[ f(x₀+h) = f(x₀) + f'(x₀)h + \frac{1}{2} f''(x₀)h^2 \]
\[ f(x₀-h) = f(x₀) - f'(x₀)h + \frac{1}{2} f''(x₀)h^2 \]
Subtracting and adding f(x₀+h) and f(x₀-h) respectively yields:
\[ f'(x₀) = \frac{f(x₀+h) - f(x₀-h)}{2h} \]
\[ f''(x₀) = \frac{f(x₀+h) - 2f(x₀) + f(x₀-h)}{h^2} \]
Convolve with
First Derivative: [-1 0 1]
Second Derivative: [1 -2 1]

2D Edge Detection: Canny
1. Filter out noise
   - Use a 2D Gaussian Filter.
2. Take a derivative \( J = I \otimes G \)
   - Compute the magnitude of the gradient:
\[ \nabla J = (J_x, J_y) = \left( \frac{\partial J}{\partial x}, \frac{\partial J}{\partial y} \right) \]
\[ \|\nabla J\| = \sqrt{J_x^2 + J_y^2} \]

Next step
Can next step, be the same for 2-D edge detector as in 1-D detector??

NO!!

Smoothing and Differentiation
• Need two derivatives, in x and y direction.
• Filter with Gaussian and then compute Gradient, OR
• Use a derivative of Gaussian filter
  • because differentiation is convolution, and convolution is associative
Directional Derivatives

\[
\frac{\partial G_x}{\partial x} \quad \frac{\partial G_y}{\partial y}
\]

\[
\cos \theta \frac{\partial G_x}{\partial x} + \sin \theta \frac{\partial G_y}{\partial y}
\]

Finding derivatives

Is this \( dI/dx \) or \( dI/dy \)?

There are three major issues:
1. The gradient magnitude at different scales is different; which scale should we choose?
2. The gradient magnitude is large along a thick trail; how do we identify the significant points?
3. How do we link the relevant points up into curves?

There is ALWAYS a tradeoff between smoothing and good edge localization!

Image with Edge (No Noise)

Edge Location

Image + Noise

Derivatives detect edge and noise

Smoothed derivative removes noise, but blurs edge

1 pixel

3 pixels

7 pixels

The scale of the smoothing filter affects derivative estimates

We wish to mark points along the curve where the magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: which point is the maximum, and where is the next point on the curve?
Non-maximum suppression

For every pixel in the image (e.g., q) we have an estimate of edge direction and edge normal (shown at q).

Using normal at q, find two points p and r on adjacent rows (or columns). We have a maximum if the value is larger than those at both p and at r. Interpolate to get values.

Predicting the next edge point

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).

Hysteresis Thresholding

- Track edge points by starting at point where gradient magnitude > $\tau_{\text{high}}$.
- Follow edge in direction orthogonal to gradient.
- Stop when gradient magnitude < $\tau_{\text{low}}$.
  - i.e., use a high threshold to start edge curves and a low threshold to continue them.

Input image

Single Threshold

$T=15$

$T=5$

Hysteresis Thresholding

$T_{h}=15$ $T_{l}=5$
Why is Canny so Dominant

- Still widely used after 20 years.
  1. Theory is nice (but end result same.).
  2. Details good (magnitude of gradient, non-max supression).
  3. Hysteresis an important heuristic.
  4. Code was distributed.

Boundary Detection

- Brightness
- Color
- Texture
Corner Detection

Why extract features?
• Motivation: panorama stitching
  – We have two images – how do we combine them?

Corners contain more info than lines.
• A point on a line is hard to match.

Feature extraction: Corners and blobs

Step 1: extract features
Step 2: match features
Step 3: align images

Why extract features?
• Motivation: panorama stitching
  – We have two images – how do we combine them?
Corners contain more info than lines.

- A corner is easier to match

![Image](image1.png)

The Basic Idea

- We should easily recognize the point by looking through a small window
- Shifting a window in any direction should give a large change in intensity

Edge Detectors Tend to Fail at Corners

Finding Corners

Intuition:
- Right at corner, gradient is ill-defined.
- Near corner, gradient has two different values.

Distribution of gradients for different image patches

Formula for Finding Corners

Shi-Tomasi Detector

We look at matrix:

$$C(x,y) = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Matrix is symmetric

**WHY THIS?**
General Case:

Because C is a symmetric positive definite matrix, it can be factored as follows:

$$C = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

Where R is a 2x2 rotation matrix and \( \lambda \) is non-negative.

What is region like if:

1. \( \lambda_1 = 0 \)
2. \( \lambda_2 = 0 \)
3. \( \lambda_1 = 0 \) and \( \lambda_2 = 0 \)
4. \( \lambda_1 > 0 \) and \( \lambda_2 > 0 \)

So, to detect corners

- Filter image with a Gaussian.
- Compute the gradient everywhere.
- Move window over image and construct C over the window.
- Use linear algebra to find \( \lambda_1 \) and \( \lambda_2 \).
- If they are both big, we have a corner.

1. Let \( e(x,y) = \min(\lambda_1(x,y), \lambda_2(x,y)) \)
2. \((x,y)\) is a corner if it’s local maximum of \( e(x,y) \) and \( e(x,y) > \tau \)

Parameters: Gaussian std. dev, window size, threshold

Corner Detection Sample Results

Threshold=25,000

Threshold=10,000

Threshold=5,000