Image Features

Edges

Formal Design of an Optimal Edge Detector

- Edge detection involves 3 steps:
  - Noise smoothing
  - Edge enhancement
  - Edge localization
- J. Canny formalized these steps to design an *optimal* edge detector

Canny Edge Detector

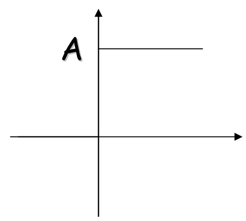
- Experiments consistently show that it performs very well
- Probably, the most used by C.V. practitioners

Canny Edge Detector

- Uses a mathematical model of the edge and the noise
- Formalizes a performance criteria
- Synthesizes the best filter

Edge Model (1D)

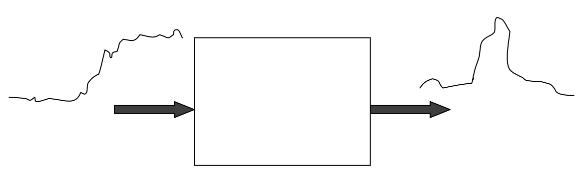
- An ideal edge can be modeled as an step



$$G(x) = \begin{cases} 0 & \text{if } x < 0 \\ A & \text{if } x \leq 0 \end{cases}$$

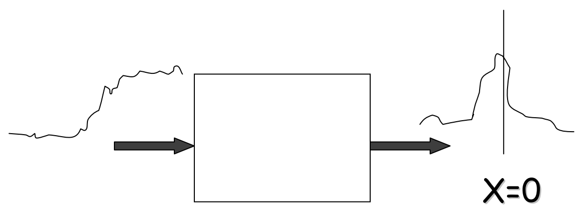
Performance Criteria (1)

- Good detection
  - The filter must have a stronger response at the edge location (x=0) than to noise



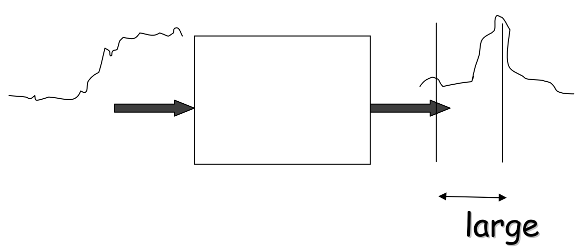
Performance Criteria (2)

- Good Localization
  - The filter response must be maximum very close to x=0



Performance Criteria (3)

- Low False Positives
  - There should be only one maximum in a reasonable neighborhood of x=0



Canny Edge Detector

- Canny found a linear, continuous filter that maximized the three given criteria.
- There is no close-form solution for the optimal filter.
- However, it looks VERY SIMILAR to the derivative of a Gaussian.

Algorithm CANNY_ENHANCER

- The input is image I; G is a zero mean Gaussian filter (std = σ)

  1. J = I * G (smoothing)
  2. For each pixel (i,j): (edge enhancement)
     - Compute the image gradient
       - » $\nabla J(i,j) = (J_x(i,j), J_y(i,j))'$
     - Estimate edge strength
       - » $e_s(i,j) = (J_x^2(i,j) + J_y^2(i,j))^{1/2}$
     - Estimate edge orientation
       - » $e_o(i,j) = \arctan(J_x(i,j)/J_y(i,j))$
- The output are images E_s and E_o

CANNY_ENHANCER

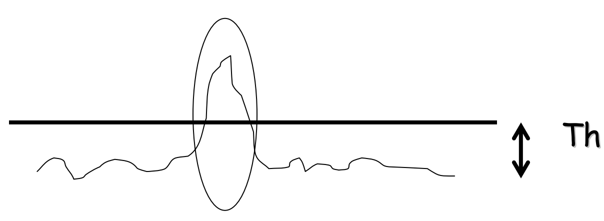
Edge strength – gradient magnitude

- The output image E_s has the magnitudes of the smoothed gradient.
- Sigma determines the amount of smoothing.
- E_s has large values at edges

$\Longrightarrow$ Edge *ENHANCER*

How do we “detect” edges?

- E_s has large values at edges:
  - Find local maxima



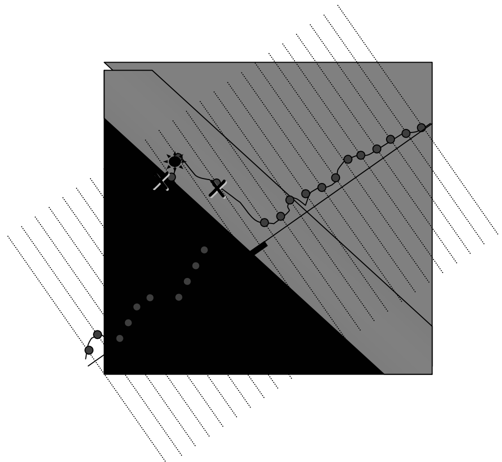
- … but it also may have wide ridges around the local maxima (large values *around* the edges)

NONMAX_SUPRESSION

Edge orientation

- The inputs are E_s & E_o (outputs of CANNY_ENHANCER)
- Consider 4 directions D={ 0,45,90,135} wrt x
- For each pixel (i,j) do:
  1. Find the direction $d \in D$ s.t. $d \cong E_o(i,j)$ (normal to the edge)
  2. If {$E_s(i,j)$ is smaller than at least one of its neigh. along d}
     - $I_N(i,j)=0$
     - Otherwise, $I_N(i,j)= E_s(i,j)$
- The output is the thinned edge image I_N

Graphical Interpretation



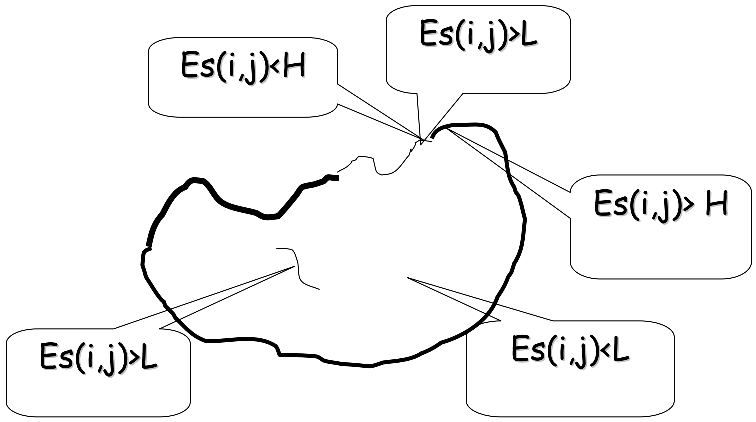
2/11/2004 Octavia I. Camps 15

Thresholding

- Edges are found by thresholding the output of NONMAX_SUPRESSION
- 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

2/11/2004 Octavia I. Camps 16

SOLUTION: Hysteresis Thresholding



Strong edges reinforce adjacent weak edges

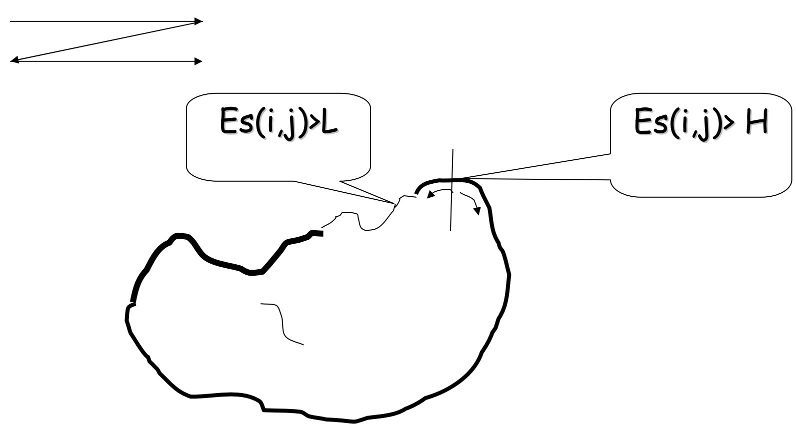
HYSTERESIS_THRESH

Inputs:
- I_N (output of NONMAX_SUPRESSION),
- E_o (output of CANNY_ENHANCER),
- thresholds L and H.

- For all pixels in I_N and scanning in a fixed order:
  1. Locate the next *unvisited* pixel s.t. $I_N(i,j)>H$
  2. Starting from $I_N(i,j)$, follow the chains of connected local maxima, in both directions perpendicular to the edge normal, as long as $I_N>L$.
     - Mark all visited points, and save the location of the contour points.

Output: a set of lists describing the contours.

Hysteresis Thresholding

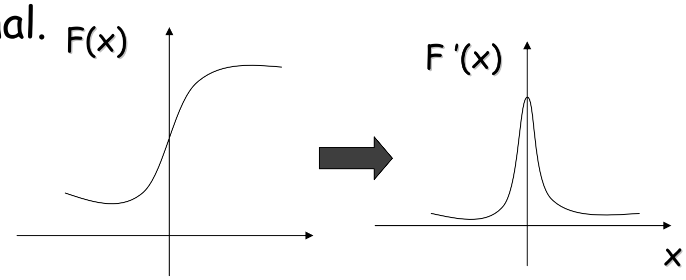


Other Edge Detectors

(2nd order derivative filters)

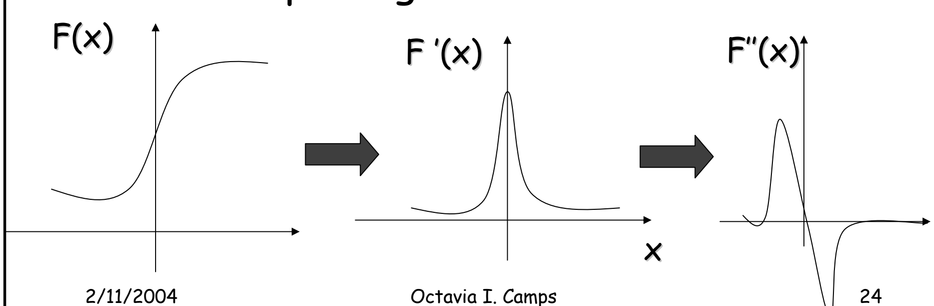
First-order derivative filters (1D)

- Sharp changes in gray level of the input image correspond to “peaks” of the first-derivative of the input signal.



Second-order derivative filters (1D)

- Peaks of the first-derivative of the input signal, correspond to “zero-crossings” of the second-derivative of the input signal.

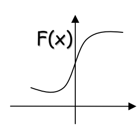


NOTE:

- $F''(x)=0$ is not enough!
  - $F'(x) = c$ has $F''(x) = 0$, but there is no edge
- The second-derivative must change sign, -- i.e. from (+) to (-) or from (-) to (+)
- The sign transition depends on the intensity change of the image – i.e. from dark to bright or vice versa.

Edge Detection (2D)

1D

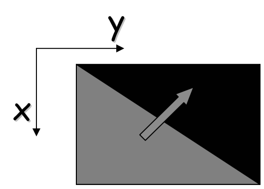


$I(x)$

$$\left|\frac{dI(x)}{dx}\right| > Th$$

$$\frac{d^2I(x)}{dx^2} = 0$$

2D



$I(x,y)$

$$|\nabla I(x,y)| = (I_x{}^2(x,y) + I_y{}^2(x,y))^{1/2} > Th$$

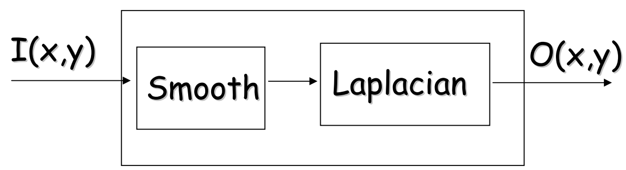
$$\tan\theta = I_x(x,y) / I_y(x,y)$$

$$\nabla^2 I(x,y) = I_{xx}(x,y) + I_{yy}(x,y) = 0$$

Laplacian

Notes about the Laplacian:

- $\nabla^2 I(x,y)$ is a SCALAR
  - ↑ Can be found using a SINGLE mask
  - ↓ Orientation information is lost
- $\nabla^2 I(x,y)$ is the sum of SECOND-order derivatives
  - But taking derivatives increases noise
  - Very noise sensitive!
- It is always combined with a smoothing operation:

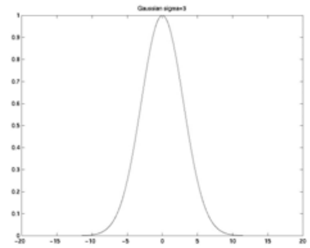


LOG Filter

- First smooth (Gaussian filter),
- Then, find zero-crossings (Laplacian filter):
  - $O(x,y) = \nabla^2(I(x,y) * G(x,y))$
- Using linearity:
  - $O(x,y) = \nabla^2 G(x,y) * I(x,y)$
  - This filter is called: "Laplacian of the Gaussian" (LOG)

1D Gaussian

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

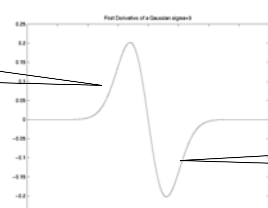


$$g'(x) = -\frac{1}{2\sigma^2} 2xe^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

First Derivative of a Gaussian

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

Positive

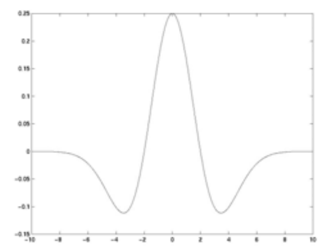


Negative

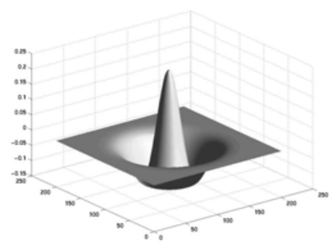
As a mask, it is also computing a difference (derivative)

Second Derivative of a Gaussian

$$g''(x) = \left(\frac{x^2}{\sigma^3} - \frac{1}{\sigma}\right)e^{-\frac{x^2}{2\sigma^2}}$$



2D



"Mexican Hat"