

Lecture-7

Edge Detection: LG, Canny

Edge Detection

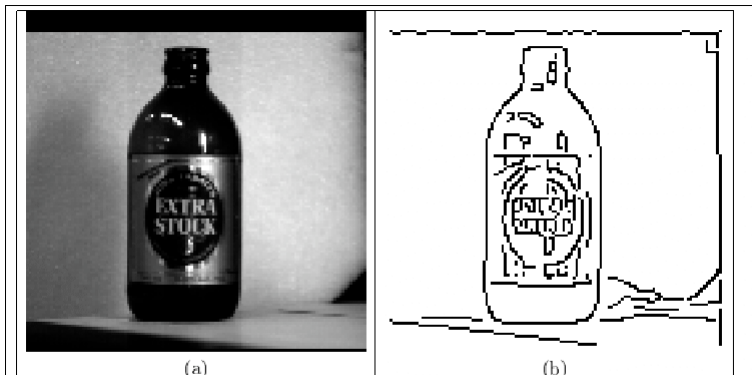


Figure 2.20: (a). The bottle image. (b). The edge map from the Canny edge detector.

Edge Detectors

- Gradient operators: Sobel, Prewit, Robert
- Laplacian of Gaussian (Marr-Hildreth)
- Gradient of Gaussian (Canny)
- Facet Model Based Edge Detector (Haralick)

Laplacian of Gaussian Edge Detector

- Generate a mask for LG for a given σ
- Apply mask to the image
- Detect zerocrossings
 - Scan along each row, record an edge point at the location of zerocrossing.
 - Repeat above step along each column

Laplacian of Gaussian

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

$$\frac{\partial}{\partial x} g(x, y) = \left(-\frac{x}{\sigma^2}\right) e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

$$\frac{\partial^2}{\partial x^2} g(x, y) = \left(-\frac{x}{\sigma^2}\right) \left(-\frac{x}{\sigma^2}\right) e^{-\frac{(x^2+y^2)}{2\sigma^2}} + \left(-\frac{1}{\sigma^2}\right) e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

$$\frac{\partial^2}{\partial x^2} g(x, y) = \frac{1}{\sigma^2} \left(1 - \frac{x^2}{\sigma^2}\right) e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Laplacian of Gaussian

$$g_{xx} = \frac{\partial^2}{\partial x^2} g(x, y) = \frac{1}{\sigma^2} \left(1 - \frac{x^2}{\sigma^2}\right) e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

$$g_{yy} = \frac{\partial^2}{\partial y^2} g(x, y) = \frac{1}{\sigma^2} \left(1 - \frac{y^2}{\sigma^2}\right) e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

$$\nabla^2 g(x, y) = \frac{\partial^2}{\partial x^2} g(x, y) + \frac{\partial^2}{\partial y^2} g(x, y)$$

$$= \frac{1}{\sigma^2} \left(1 - \frac{x^2}{\sigma^2}\right) e^{-\frac{(x^2+y^2)}{2\sigma^2}} + \frac{1}{\sigma^2} \left(1 - \frac{y^2}{\sigma^2}\right) e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

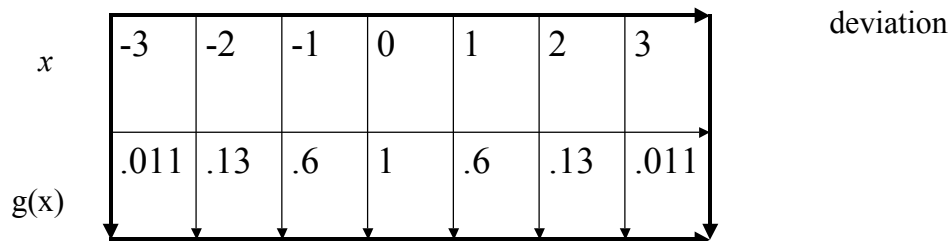
$$\nabla^2 g(x, y) = \frac{1}{\sigma^2} \left(2 - \frac{x^2 + y^2}{\sigma^2}\right) e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

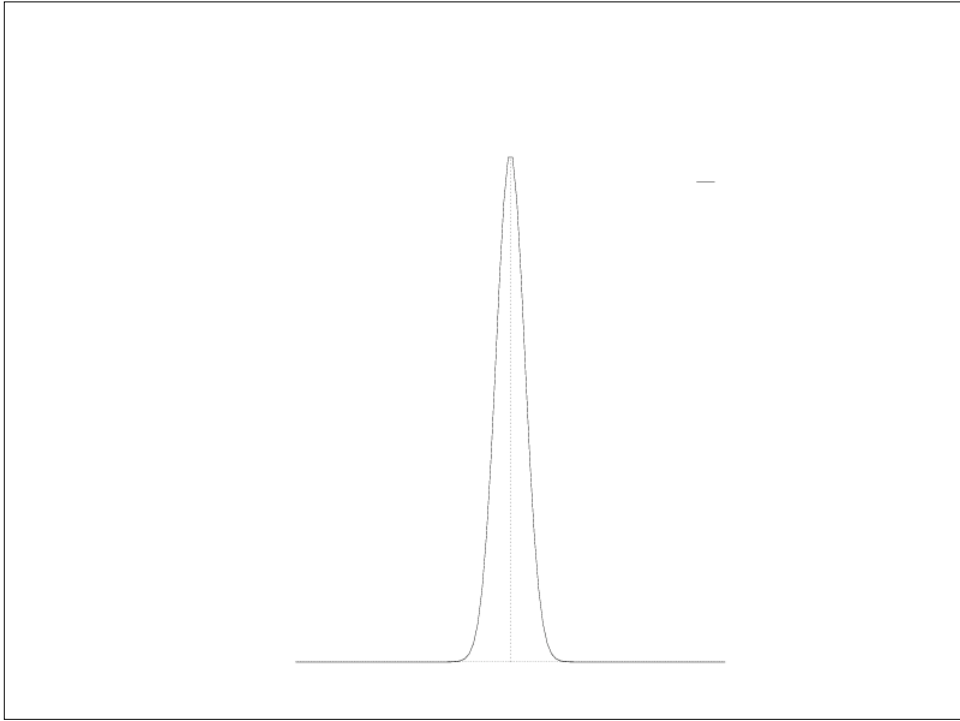
Zerocrossings

- Four cases of zerocrossings : $\{+,-\}$, $\{+,0,-\}$, $\{-,+\}$, $\{-,0,+\}$
- Slope of zerocrossing $\{a, -b\}$ is $|a+b|$.
- To detect zerocrossing apply threshold to the slope. If the slope is above some threshold, then that point is an edge point.

Gaussian

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





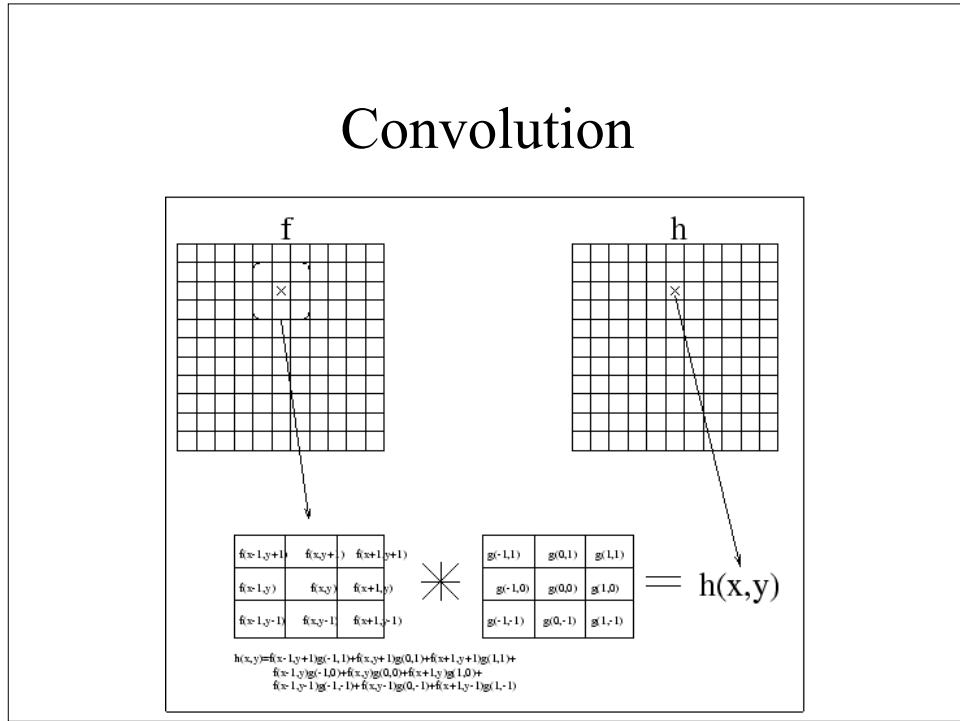
2-D Gaussian

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

0	0	0	0	1	2	2	2	1	0	0	0	0
0	0	1	3	6	9	11	9	6	3	1	0	0
0	1	4	11	20	30	34	30	20	11	4	1	0
0	3	11	26	50	73	82	73	50	26	11	3	0
1	6	20	50	93	136	154	136	93	50	20	6	1
2	9	30	73	136	198	225	198	136	73	30	9	2
2	11	34	82	154	225	255	225	154	82	34	11	2
2	9	30	73	136	198	225	198	136	73	30	9	2
1	6	20	50	93	136	154	136	93	50	20	6	1
0	3	11	26	50	73	82	73	50	26	11	3	0
0	1	4	11	20	30	34	30	20	11	4	1	0
0	0	1	3	6	9	11	9	6	3	1	0	0
0	0	0	0	1	2	2	2	1	0	0	0	0

$$\sigma = 2$$

Convolution



Convolution (contd)

$$h(x, y) = \sum_{i=-1}^1 \sum_{j=-1}^1 f(x+i, y+j)g(i, j)$$

$$h(x, y) = f(x, y) * g(x, y)$$

Separability of Gaussian

$$h(x, y) = f(x, y) * g(x, y)$$

Requires n^2 multiplications for a n by n mask, for each pixel.

$$h(x, y) = (f(x, y) * g(x)) * g(y)$$

This requires $2n$ multiplications for a n by n mask, for each pixel.

Separability of Laplacian of Gaussian

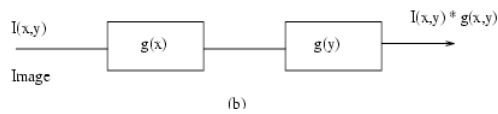
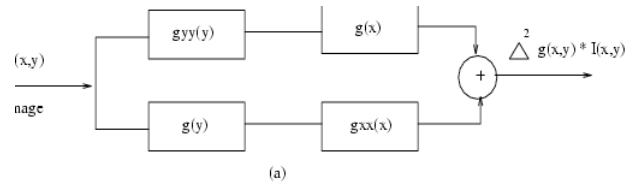
$$h(x, y) = f(x, y) * \nabla^2 g(x, y)$$

Requires n^2 multiplications for a n by n mask, for each pixel.

$$h(x, y) = (f(x, y) * g_{xx}(x)) * g(y) + (f(x, y) * g_{yy}(y)) * g(x)$$

This requires $4n$ multiplications for a n by n mask, for each pixel.

Separability



Decomposition of LG into four 1-D convolutions

- Convolve the image with a second derivative of Gaussian mask $g_{yy}(y)$ along each column
- Convolve the resultant image from step (1) by a Gaussian mask $g(x)$ along each row. Call the resultant image I^x .
- Convolve the original image with a Gaussian mask, $g(y)$ along each column
 - Convolve the resultant image from step (3) by a second derivative of Gaussian mask $g_{xx}(x)$ along each row. Call the resultant image I^y .
- Add I^x and I^y .

Canny Edge Detector

- Compute the gradient of image $f(x,y)$ by convolving it with the first derivative of Gaussian masks in x and y directions.
- Perform non-maxima suppression on the gradient magnitude.
- Apply hysteresis thresholding to the non-maxima suppressed magnitude.

Canny Edge Detector

$$f_x(x,y) = f(x,y) * g_x(x,y) = (f(x,y) * g_x(x)) * g(y)$$

$$f_y(x,y) = f(x,y) * g_y(x,y) = (f(x,y) * g_y(y)) * g(x)$$

(f_x, f_y) Gradient Vector

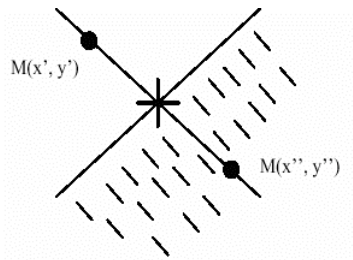
$$\text{magnitude} = \sqrt{(f_x^2 + f_y^2)}$$

$$\text{direction} = \theta = \tan^{-1} \frac{f_y}{f_x}$$

Non-maxima Suppression

- Suppress the pixels which are not local maxima.

$$M(x, y) = \begin{cases} M(x, y) & \text{if } M(x, y) > M(x-1, y) \text{ \& } \\ & \text{if } M(x, y) > M(x+1, y) \\ 0 & \text{otherwise} \end{cases}$$

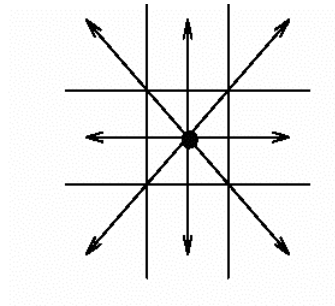


Quantization in Eight Possible Directions

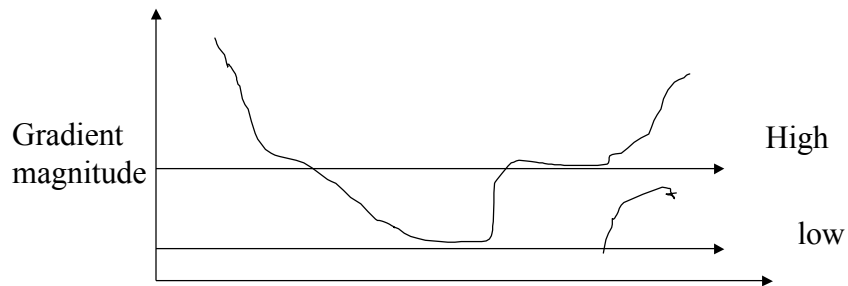
(f_x, f_y) Gradient Vector

$$\text{magnitude} = \sqrt{f_x^2 + f_y^2}$$

$$\text{direction} = \theta = \tan^{-1} \frac{f_y}{f_x}$$



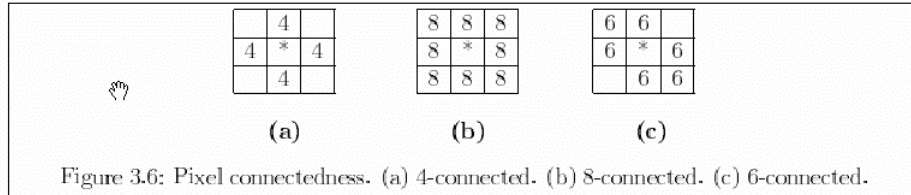
Hysteresis Thresholding



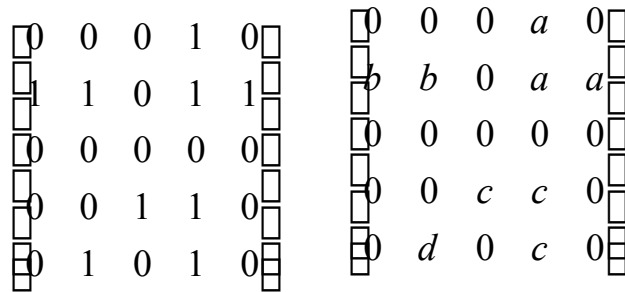
Hysteresis Thresholding

- Scan the image from left to right, top-bottom. If
 - The gradient magnitude at a pixel is above a high threshold declare that as an edge point
 - Then recursively consider the *neighbors* of this pixel.
 - If the gradient magnitude is above the low threshold declare that as an edge pixel.

Connectedness



Connected Component



4

Connected Component

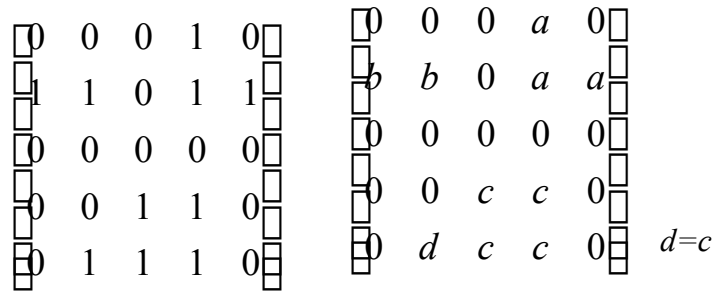
0	0	0	1	0	0	0	0	<i>a</i>	0
1	1	0	1	1	<i>b</i>	<i>b</i>	0	<i>a</i>	<i>a</i>
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	<i>c</i>	<i>c</i>	0
0	1	0	1	0	0	<i>c</i>	0	<i>c</i>	0

8

1. Scan the binary image left to right, top to bottom.
2. If there is an unlabeled pixel with a value of '1' assign a new label to it.
3. Recursively check the neighbors of the pixel in step 2 and assign the same label if they are unlabeled with a value of '1'.
4. Stop when all the pixels of value '1' have been labeled.

Figure 3.7: Recursive Connected Component Algorithm.

Sequential



1. Scan the binary image left to right, top to bottom.
2. If an unlabeled pixel has a value of '1', assign a new label to it according to the following rules:

$\begin{matrix} 0 & 0 \\ 0 & 1 \end{matrix} \rightarrow \begin{matrix} 0 & L \end{matrix}$	$\begin{matrix} 0 & 0 \\ L & 1 \end{matrix} \rightarrow \begin{matrix} L & L \end{matrix}$
$\begin{matrix} L & L \\ 0 & 1 \end{matrix} \rightarrow \begin{matrix} L & L \end{matrix}$	$\begin{matrix} L & L \\ M & 1 \end{matrix} \rightarrow \begin{matrix} L & L \end{matrix}$ (Set $L = M$).
3. Determine equivalence classes of labels.
4. In the second pass, assign the same label to all elements in an equivalence class.

Figure 3.8: Sequential Connected Component Algorithm.

Recursive

0	0	0	1	0	0	0	0	<i>a</i>	0
1	1	0	1	1	<i>b</i>	<i>b</i>	0	<i>a</i>	<i>a</i>
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	<i>c</i>	<i>c</i>	0
0	1	1	1	0	0	<i>c</i>	<i>c</i>	<i>c</i>	0