GUC
German University in Cairo

Winter 2019

DMET 901 – Computer Vision

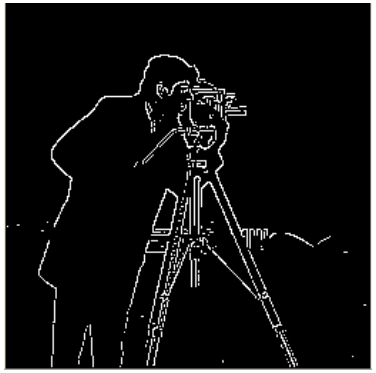
Edge Detection

Seif Eldawlatly

Edge Detection

- Edges are pixels where brightness changes abruptly
- Edges are important for image perception
- An edge is a vector variable with magnitude and direction
- Example:

Original Image

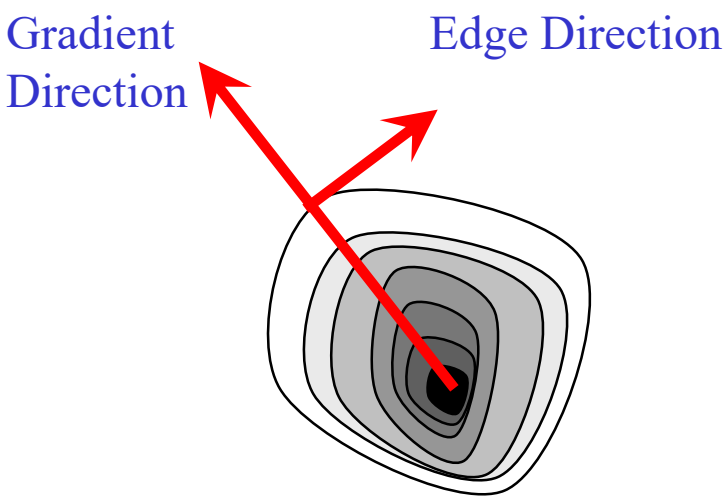


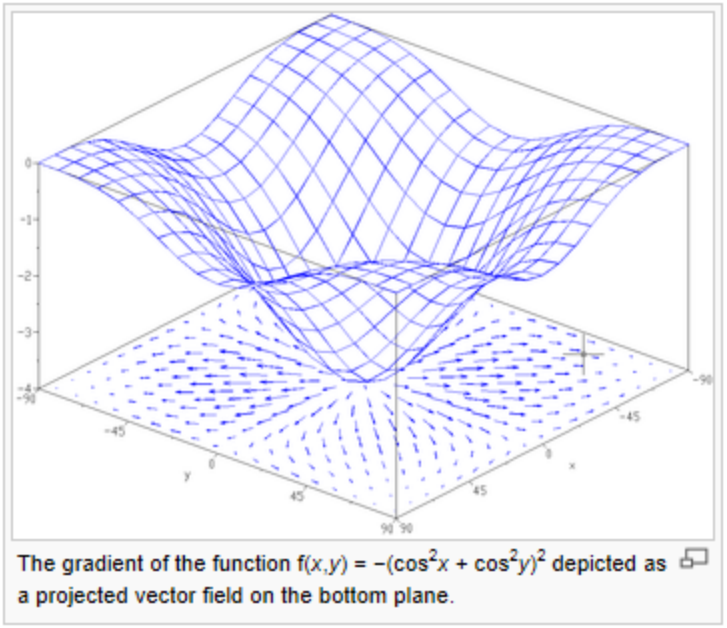
Edge Image

Edge Detection

- The direction of the edge is perpendicular to the direction of the gradient

- The gradient gives the direction of maximal growth of the image function from black to white





The gradient of the function $f(x,y) = -(\cos^2 x + \cos^2 y)^2$ depicted as a projected vector field on the bottom plane.

Image Gradient

- The gradient magnitude and direction are continuous image functions

$$\left|\nabla_g(x,y)\right| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2}$$

$$\psi = \tan^{-1}\left(\frac{\partial g}{\partial y} \Big/ \frac{\partial g}{\partial x}\right)$$

- A digital image is discrete in nature, so derivatives are approximated by differences

Image Gradient

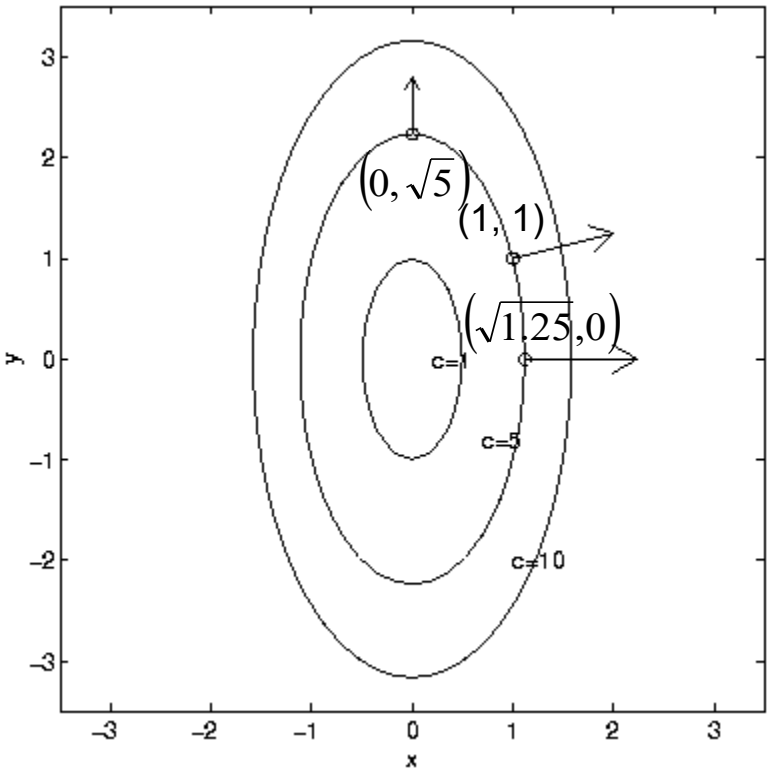
- Why does the image gradient give the direction of maximal change?
- Example:

$$f(x, y) = 4x^2 + y^2$$

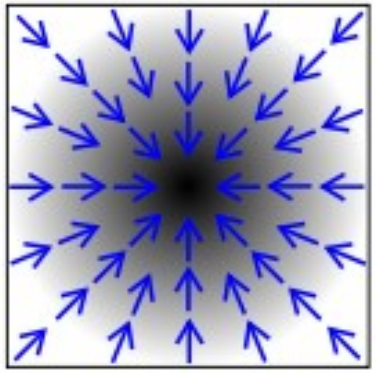
The gradient vector at any point (x, y)

$$\langle 8x, 2y \rangle$$

As the figure shows, the gradient vector at (x, y) is always perpendicular to the level curve through (x, y)



- For an image:



Edge Detection

- Operators which detect edges are represented by a collection of masks, each corresponding to a certain direction

- Two groups of gradient operators:
  - Operators approximate first derivative
  - Operators based on the zero-crossings of the image function second derivative (LoG)

- Gradient detection is based on convolution in very small neighborhood

First Derivative Operators

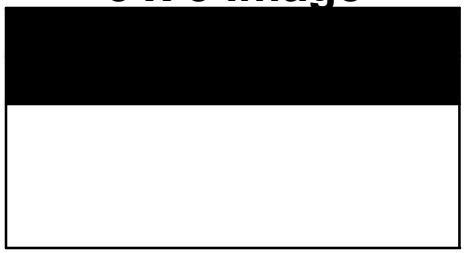
- First derivative operators
  - Roberts

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$gradient\ \ magn. = \left|g(i,j) - g(i+1,j+1)\right| + \left|g(i,j+1) - g(i+1,j)\right|$$

  - Example

5 x 5 Image



0	0	0	0	0
0	0	0	0	0
255	255	255	255	255
255	255	255	255	255
255	255	255	255	255

↓ **Roberts**

0	0	0	0	0
510	510	510	510	510
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Edge Image

First Derivative Operators

- First derivative operators
  - Roberts

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$gradient\ magn. = |g(i,j) - g(i+1,j+1)| + |g(i,j+1) - g(i+1,j)|$$

  - Prewitt

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$gradient\ magn. = \sqrt{x^2 + y^2}, gradient\ direction = \tan^{-1}\left(\frac{y}{x}\right)$$

$$where\ x\ is\ h_2\ response\ \&\ y\ is\ h_1\ response.$$

First Derivative Operators

- First derivative operators
  - Sobel

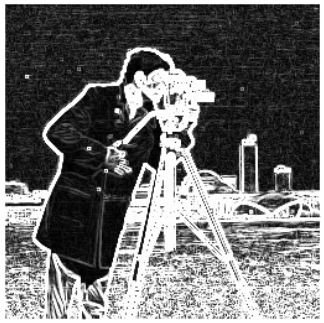
$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

  - Robinson

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$$

  - Kirsch

$$h_1 = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix}, h_2 = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix}$$



Thresholding and Thinning

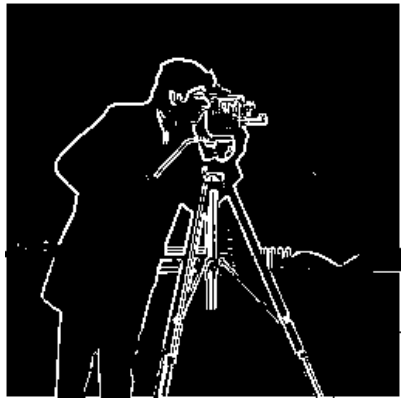
- Two more steps to get the final edge image

  1. Thresholding

Prewitt Output (Gray-level Image)

> 1

Binary Image



Pixels > 1 are set to 1, others set to 0

> 0.5

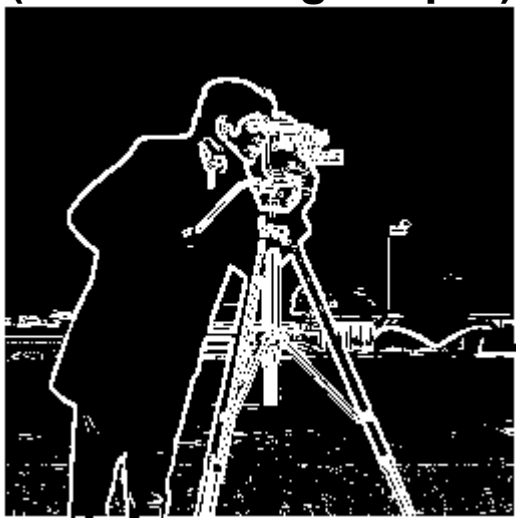
Pixels > 0.5 are set to 1, others set to 0

Thresholding and Thinning

- Two more steps to get the final edge image

  2. Thinning

Binary Image
(Thresholding Output)



Edges are more
than 1 pixel wide

Thinning

Edges are 1 pixel wide

Thresholding and Thinning

- Thinning Algorithm (Non-maximal Suppression)

1. Quantize gradient directions eight ways according to 8-connectivity

2. For each pixel with non-zero gradient magnitude, inspect the two adjacent pixels indicated by the direction of its gradient

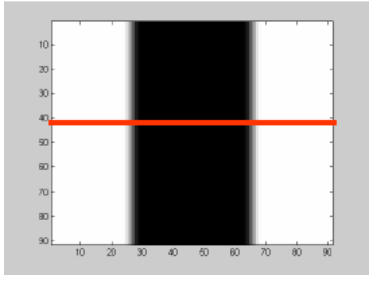
3. If the gradient magnitude of either of these two exceeds that of the pixel under inspection, mark it for deletion

4. When all pixels have been inspected, re-scan the image and erase to zero all edge pixels marked for deletion

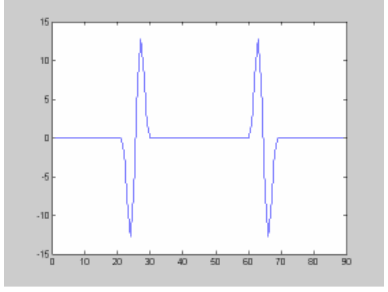
Laplacian of Gaussian (LoG)

- This method is based on the detection of zero-crossings in the second derivative of the image function

- The second derivative of the image function should be zero at an edge position

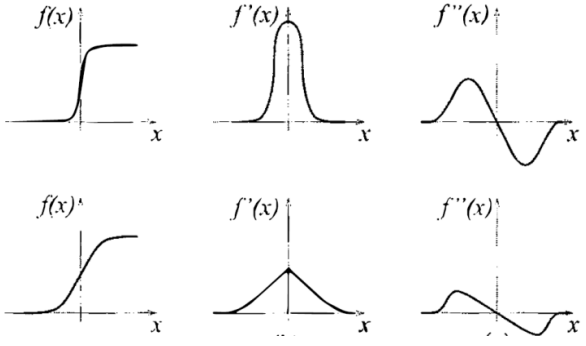


Image



2nd Derivative

- Since edges correspond to sudden change in intensity, they could be detected by finding the maximum/minimum of the derivative of the image function



Laplacian of Gaussian (LoG)

- It is much easier and more precise to find a zero crossing position than an extreme

- A popular second derivative operator is the Laplacian operator:

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

- This operator is very sensitive to noise. Therefore, the image should be smoothed before applying the filter

- To smooth the image $f(x, y)$, the image is convolved with a Gaussian filter $g(x, y)$

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

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

Laplacian of Gaussian (LoG)

- The Laplacian operator is then applied to $s(x, y)$, so it is Laplacian of Gaussian (LoG)

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

  where $h(x, y)$ is a filter which approximates the Laplacian operator.

- Since the convolution operator is associative, the Gaussian smoothing filter can be convolved first with the Laplacian filter, then convolve the Laplacian of Gaussian (LoG) with the image

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

- This requires fewer computations as the LoG mask can be computed in advance

Laplacian of Gaussian (LoG)

- The LoG function is

$$LoG(x,y) = \frac{-1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- A discrete mask that approximates this function for a Gaussian of σ = 1.2 and with appropriate scaling and rounding:

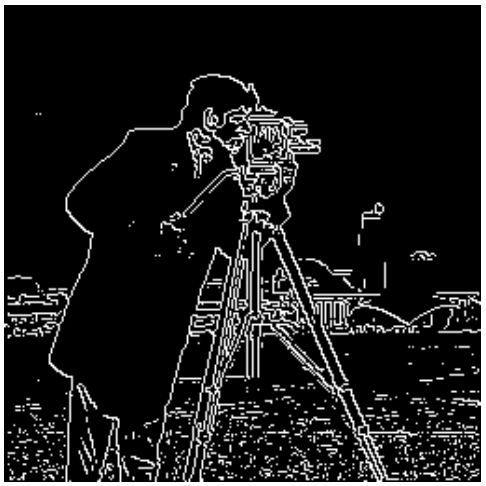
$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 3 & 4 & 3 & 2 & 0 & 0 \\ 0 & 2 & 4 & 5 & 4 & 5 & 4 & 2 & 0 \\ 1 & 3 & 5 & -6 & -18 & -6 & 5 & 3 & 1 \\ 1 & 4 & 4 & -18 & -40 & -18 & 4 & 4 & 1 \\ 1 & 3 & 5 & -6 & -18 & -6 & 5 & 3 & 1 \\ 0 & 2 & 4 & 5 & 4 & 5 & 4 & 2 & 0 \\ 0 & 0 & 2 & 3 & 4 & 3 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

- The width of a mask with σ standard deviation is approximately:
  2 x ceil(3xσ) + 1

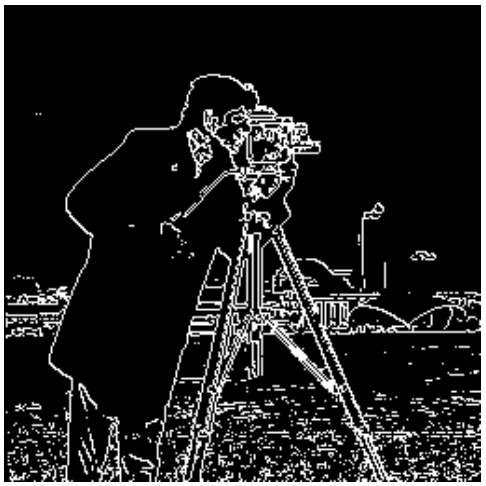
Laplacian of Gaussian (LoG)

- LoG Algorithm
  1. Determine the size of the LoG mask
  2. Form the LoG mask
  3. Convolve the LoG mask with the image
  4. Search for the zero crossings in the result of the convolution
  5. If there is sufficient edge evidence from a first derivative operator, form the edge image by setting the position of zero crossing to 1 and other positions to 0

σ = 0.5



σ = 2



σ = 5