COMP 478/6771 (FALL 2020)

Digital Image Processing

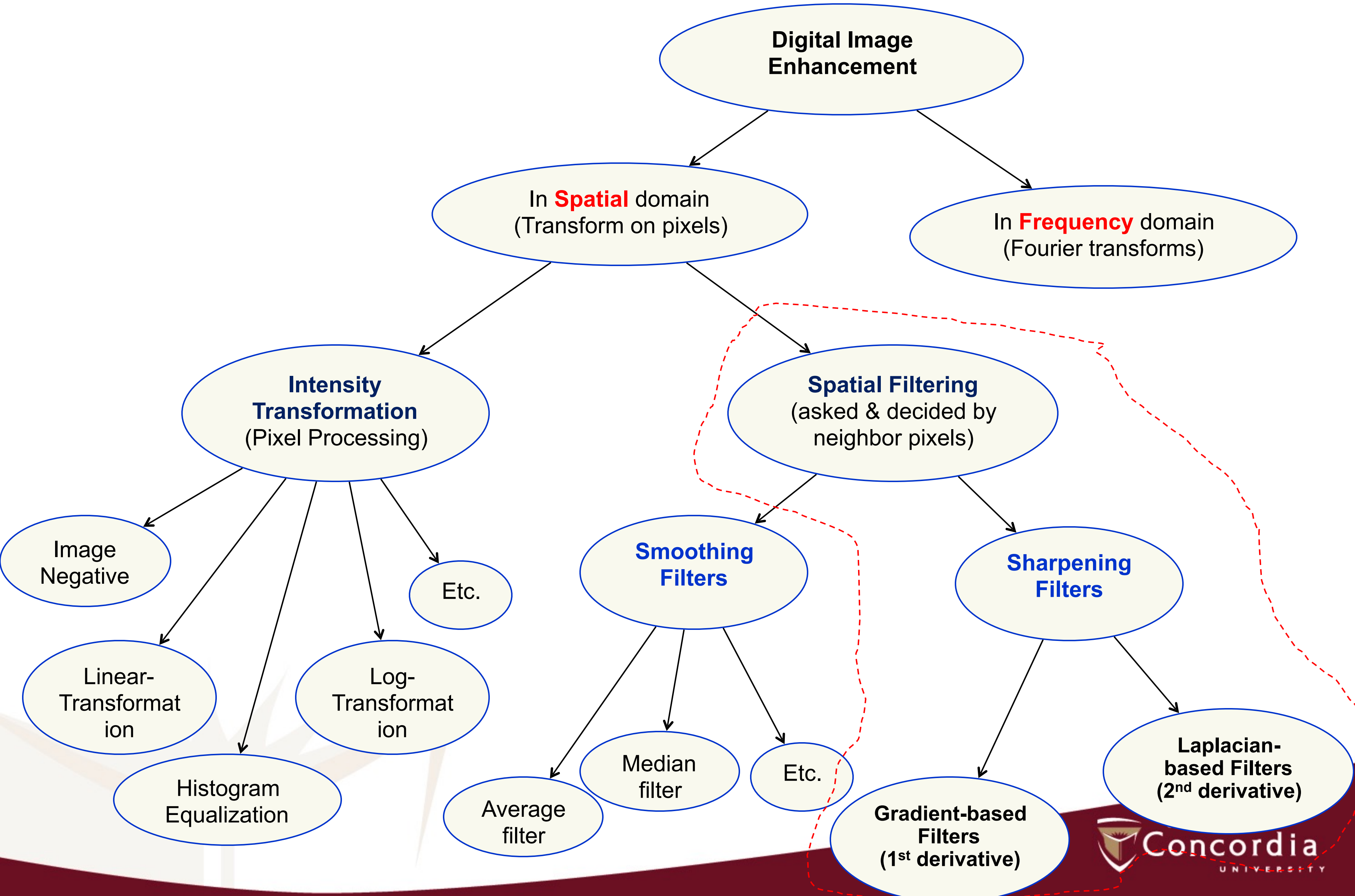
Digital Image Enhancement in Spatial Domain (cont.)

Instructor: Prof. Yiming Xiao

Tutors:

Materials provided by Dr. T. D. Bui

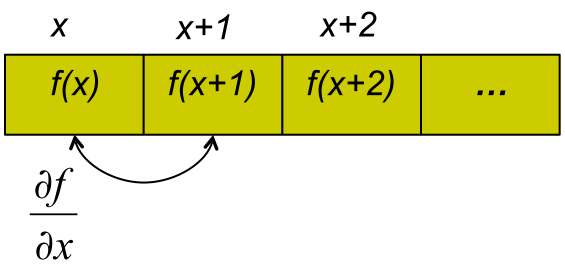
Introduction



The first-order Derivatives – The Gradient

In one-dimensional, the first-order derivative of the function *f(x)*

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$



In two-dimensional, the gradient of the function *f(x, y)*

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$g_x = \frac{\partial f(x,y)}{\partial x} = f(x+1, y) - f(x,y)$$

$$g_y = \frac{\partial f(x,y)}{\partial y} = f(x, y+1) - f(x,y)$$

The magnitude (length):

$$M(x,y) = mag(\nabla f) = \sqrt{g_x^2 + g_y^2} \approx |g_x| + |g_y|$$

The first-order Derivatives – The Gradient

$$M(x,y) = mag(\nabla f) = \sqrt{g_x^2 + g_y^2} \approx |g_x| + |g_y|$$

How to apply the first-order derivative filters to the image *I*(*x,y*) → *G*(*x,y*) ?

Step 1:
- Convolve the 1st mask g_x with the image $I(x,y) \rightarrow G_1(x,y)$
- Convolve the 2nd mask g_y with the image $I(x,y) \rightarrow G_2(x,y)$

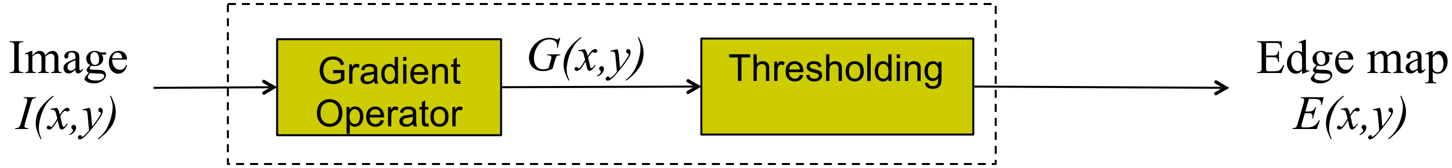
Step 2:
- Get the absolute values of $|G_1(x,y)|$, $|G_2(x,y)|$.

Step 3:
- Get the filtered image: $G(x,y) = |G_1(x,y)| + |G_2(x,y)|$

Edge detection

Note:

Gradient – based methods can be used to detect edges in images (detect sudden changes in image intensity) by adding one more step **thresholding**.



Edge detection: *I(x,y)* → *E(x,y)*

$$E(x,y) = \begin{cases} 1 & G(x,y) > th \\ 0 & otherwise \end{cases}$$

Step 1:

- Convolve the 1st mask g_x with the image $I(x,y) \rightarrow G_1(x,y)$
- Convolve the 2nd mask g_y with the image $I(x,y) \rightarrow G_2(x,y)$

Step 2:

- Get the absolute values of $|G_1(x,y)|, |G_2(x,y)|$.

Step 3:

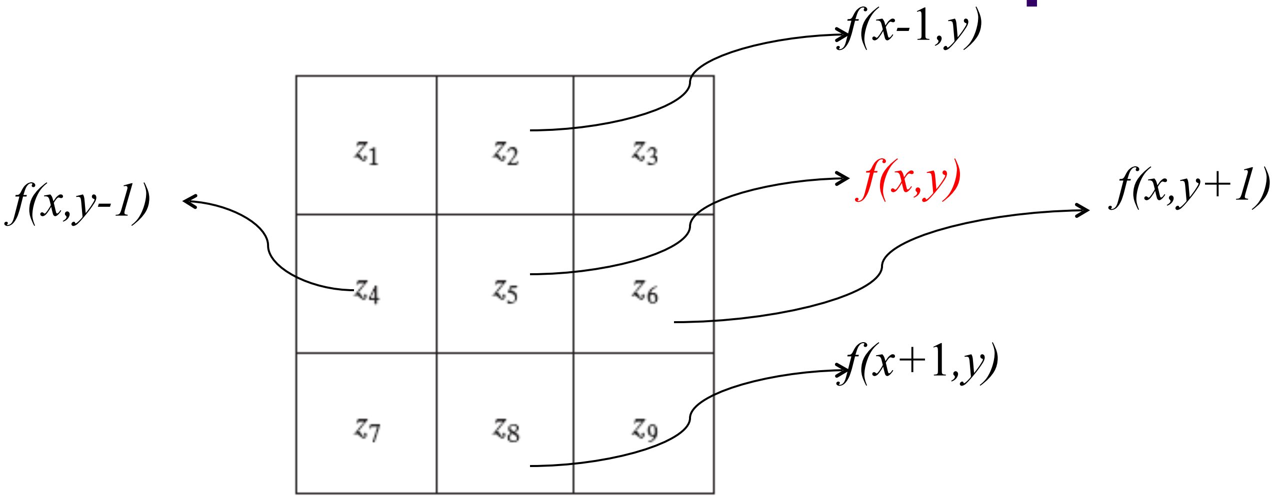
- Get the filtered image: $G(x,y) = |G_1(x,y)| + |G_2(x,y)|$

Step 4:

- Apply the thresholding at **th (we choose)**:
  + if pixel value $G(x,y)$ < **th** then $E(x,y) = 0$
  + if pixel value $G(x,y)$ ≥ **th** then $E(x,y) = 1$

How to set-up the filtering masks

The first-order Derivatives – Roberts operators



Remember, the first-order of the function ***f(x, y)***

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$g_x = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) = z_8 - z_5$$

$$g_y = \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y) = z_6 - z_5$$

Roberts suggested using the cross differences:

$$g_x = f(x+1,y+1) - f(x,y) = z_9 - z_5$$

$$g_y = f(x+1,y) - f(x,y+1) = z_8 - z_6$$

Roberts cross-gradient operators

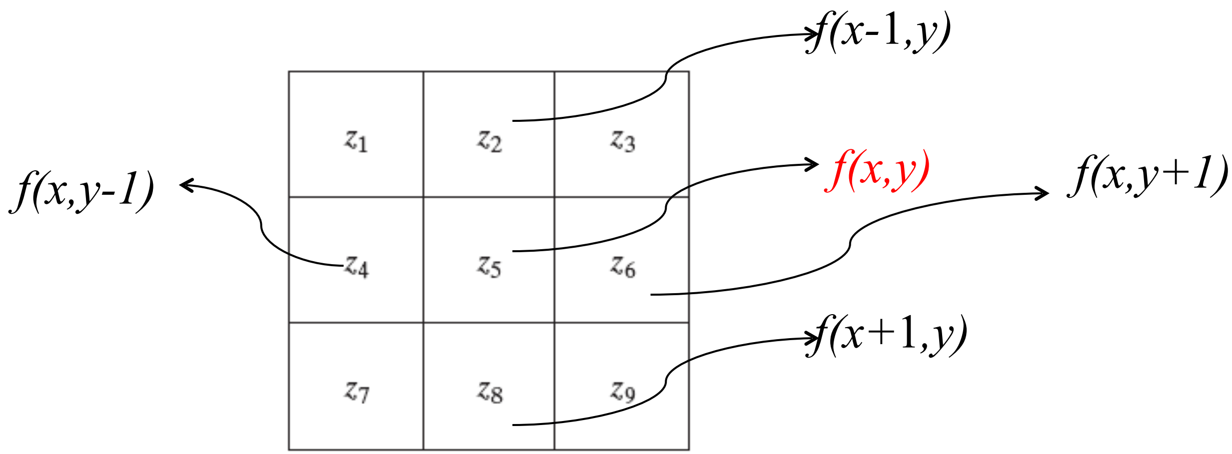
-1	0
0	1

g_x

0	-1
1	0

g_y

The first-order Derivatives – Prewitt operators



Using 3×3 neighbors:

$$g_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

$\xrightarrow{\text{Prewitt operators}}$

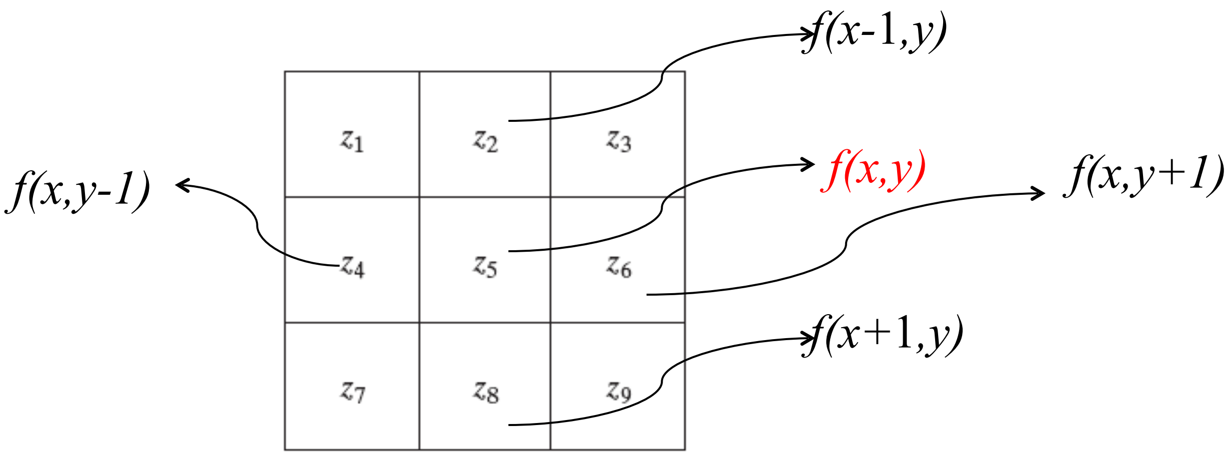
-1	-1	-1
0	0	0
1	1	1

g_x

-1	0	1
-1	0	1
-1	0	1

g_y

The first-order Derivatives – Sobel operators



Using 3×3 neighbors:

$$g_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

$\xrightarrow{\text{Sobel operators}}$

-1	-2	-1
0	0	0
1	2	1

g_x

-1	0	1
-2	0	2
-1	0	1

g_y

Notes:

- Using a weigh value of 2 in the center coefficient is to achieve some smoothing by giving more importance to the center point.
- Sum of a mask is zero

Example

Using the **Sobel gradient operators** to find the **edges** of the following image *I(x,y)*, with the threshold **th = 20** (use matlab to check the result).

$$I(x,y) = \begin{matrix} 9&9&9&9&9&9&9&2&2 \\ 9&8&9&9&9&9&2&2&2 \\ 9&9&9&9&9&9&3&2&2 \\ 9&9&9&9&9&2&2&2&2 \\ 7&9&9&9&9&2&2&2&2 \\ 9&9&9&9&2&2&2&2&2 \\ 9&9&9&9&2&2&2&4&2 \\ 9&9&9&2&2&2&2&2&2 \\ 9&9&2&9&2&2&1&2&2 \end{matrix}$$

9×9

Example (cont.)

Step 1 & 2:

- Convolve I(x,y) with the g_x:

$$G_1(x,y) = I(x,y) \, \Theta \, g_x =$$

26	34	35	36	36	29	15	8	6
0	0	0	0	0	-6	-12	-6	0
1	2	1	0	-7	-14	-7	0	0
-4	-2	0	0	-7	-15	-9	-1	0
0	0	0	-7	-14	-7	0	0	0
4	2	0	-7	-14	-7	2	4	2
0	0	-7	-14	-7	0	0	0	0
0	-7	-14	-7	0	-1	-4	-5	-2
-27	-36	-29	-15	-8	-8	-8	-8	-6

g_x

-1	-2	-1
0	0	0
1	2	1

$$|G_1(x,y)| =$$

26	34	35	36	36	29	15	8	6
0	0	0	0	0	6	12	6	0
1	2	1	0	7	14	7	0	0
4.	2	0	0	7	15	9	1	0
0	0	0	7	14	7	0	0	0
4	2	0	7	14	7	2	4	2
0	0	7	14	7	0	0	0	0
0	7	14	7	0	1	4	5	2
27	36	29	15	8	8	8	8	6

Example (cont.)

Step 1 & 2 (cont.) :

- Convolve $I(x,y)$ with the g_y:

$$G_2(x,y) = I(x,y) \, \Theta \, g_y = \begin{matrix} 26 & 0 & 1 & 0 & 0 & -7 & -21 & -14 & -6 \\ 34 & 0 & 2 & 0 & 0 & -20 & -28 & -8 & -8 \\ 35 & 0 & 1 & 0 & -7 & -26 & -21 & -2 & -8 \\ 36 & 2 & 0 & 0 & -21 & -27 & -7 & -1 & -8 \\ 36 & 4 & 0 & -7 & -28 & -21 & 0 & 0 & -8 \\ 36 & 2 & 0 & -21 & -28 & -7 & 2 & 0 & -10 \\ 36 & 0 & -7 & -28 & -21 & 0 & 4 & 0 & -12 \\ 36 & -7 & -14 & -21 & -14 & -1 & 2 & 1 & -10 \\ 27 & -14 & -7 & -7 & -14 & -2 & 0 & 2 & -6 \end{matrix}$$

g_y

-1	0	1
-2	0	2
-1	0	1

$$|G_2(x,y)| = \begin{matrix} 26 & 0 & 1 & 0 & 0 & 7 & 21 & 14 & 6 \\ 34 & 0 & 2 & 0 & 0 & 20 & 28 & 8 & 8 \\ 35 & 0 & 1 & 0 & 7 & 26 & 21 & 2 & 8 \\ 36 & 2 & 0 & 0 & 21 & 27 & 7 & 1 & 8 \\ 36 & 4 & 0 & 7 & 28 & 21 & 0 & 0 & 8 \\ 36 & 2 & 0 & 21 & 28 & 7 & 2 & 0 & 10 \\ 36 & 0 & 7 & 28 & 21 & 0 & 4 & 0 & 12 \\ 36 & 7 & 14 & 21 & 14 & 1 & 2 & 1 & 10 \\ 27 & 14 & 7 & 7 & 14 & 2 & 0 & 2 & 6 \end{matrix}$$

Example (cont.)

Step 3:

$$G(x,y) = |G_1(x,y)| + |G_2(x,y)| = \begin{matrix} 52 & 34 & 36 & 36 & 36 & 36 & 36 & 22 & 12 \\ 34 & 0 & 2 & 0 & 0 & 26 & 40 & 14 & 8 \\ 36 & 2 & 2 & 0 & 14 & 40 & 28 & 2 & 8 \\ 40 & 4 & 0 & 0 & 28 & 42 & 16 & 2 & 8 \\ 36 & 4 & 0 & 14 & 42 & 28 & 0 & 0 & 8 \\ 40 & 4 & 0 & 28 & 42 & 14 & 4 & 4 & 12 \\ 36 & 0 & 14 & 42 & 28 & 0 & 4 & 0 & 12 \\ 36 & 14 & 28 & 28 & 14 & 2 & 6 & 6 & 12 \\ 54 & 50 & 36 & 22 & 22 & 10 & 8 & 10 & 12 \end{matrix}$$

Example (cont.)

Step 4:

- Apply the thresholding **th** = 20:
  - + if pixel value $G(x,y) <$ **th** then $E(x,y) = 0$
  - + if pixel value $G(x,y) \geq$ **th** then $E(x,y) = 1$

$$E(x,y) = \begin{matrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

The second-order Derivatives – The Laplacian

In one-dimensional, the second-order derivative of the function *f(x)*

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

In two-dimensional, the Laplacian of the function *f(x, y)*

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

$$\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

The discrete Laplacian of two variable:

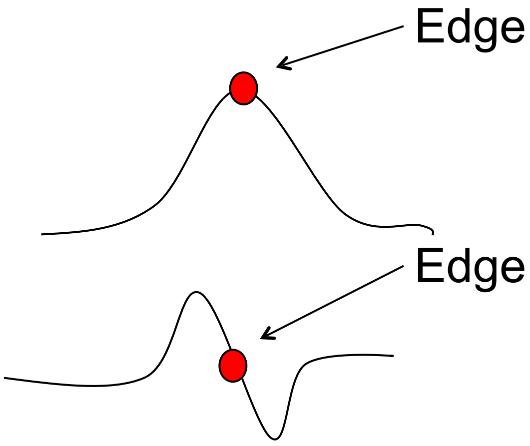
$$\partial^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

The second-order Derivatives – The Laplacian

Note:

- *In the Gradient-based method: check for the high values*

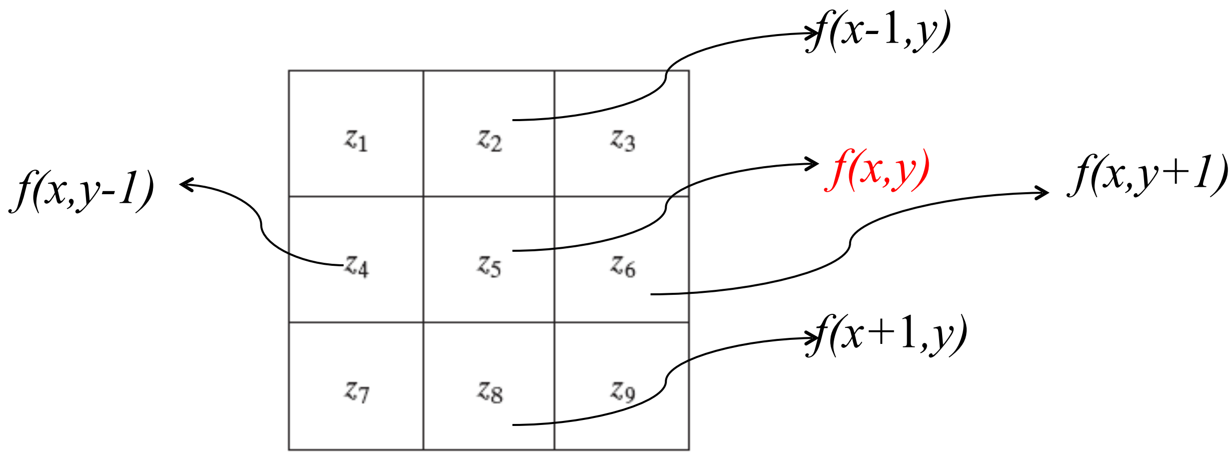
- *In the Laplacian-based method: check for the zero values*



The Laplacian-based methods can be divided into:

- *Standard Laplacian method*
- *Laplacian of Gaussian (LoG)*

The standard Laplacian



$$\partial^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

Laplacian operators:

g

0	1	0
1	-4	1
0	1	0

or

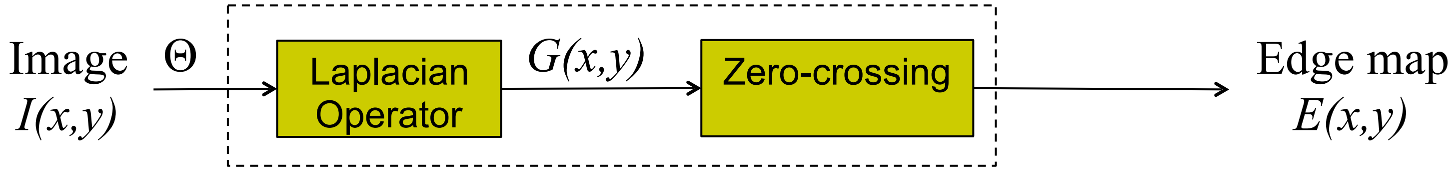
1	1	1
1	-8	1
1	1	1

or

-1	-1	-1
-1	8	-1
-1	-1	-1

Edge detection with the standard Laplacian

Note:
Detect the sudden changes in intensity images. This method is sensitive to abrupt changes, but not slow changes.



$$E(x,y) = \begin{cases} 1 & G(x,y) = 0 \\ 0 & otherwise \end{cases}$$

Edge detection: *I*(*x*,*y*) → *E*(*x*,*y*)

Step 1:
- Convolve the mask *g* with the image *I(x,y)* → *G*(*x*,*y*)

Θ: convolve

Step 2:
- Apply the Zero-crossing:
  + if neighbor pixels value *G(x,y) sign change then E(x,y)* = 1
  + if neighbor pixels value *G(x,y) sign does not change then* then *E(x,y)* = 0

The Laplacian of Gaussian (LoG)

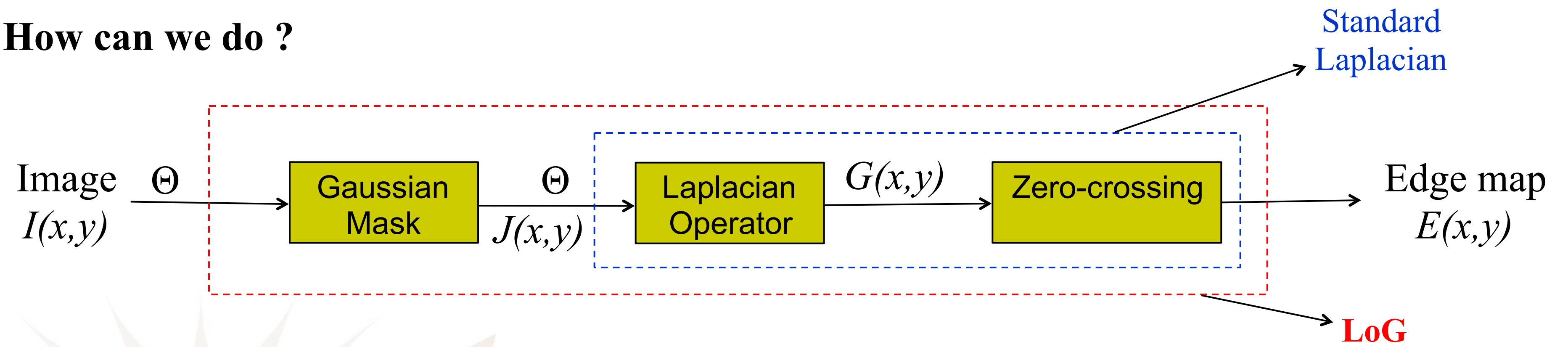
Why do we use the LoG ?

- The effect of noise is exaggerate when we get the first derivative.
- The second derivative will exaggerate the noise one more time.
- Additionally, the first derivative doesn't give us any information about the direction of edges.

Solutions:

- We use the second derivatives to get information about the direction of edges.
- However, to reduce the noise, we use the Gaussian smoothing to remove the noise.

How can we do ?



$$E(x,y)=\begin{cases}1 & G(x,y)=0\\ 0 & otherwise\end{cases}$$

The Laplacian of Gaussian (LoG)

Notes:

- The convolution is a time consuming operation. Here, we have used this operation 2 times !

=> We can combine both together, apply mask and smooth at the same time.

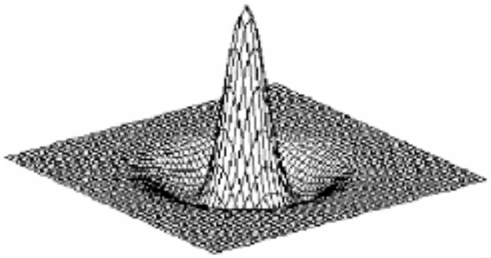
$$(f \ \Theta \ g)'' = f \ \Theta \ g''$$

How can we do ? $g(r) = e^{-r^2/2\sigma^2}$

- The Gaussian: $g'(r) = \dfrac{\partial g(r)}{\partial r} = \dfrac{-r}{\sigma^2} e^{-r^2/2\sigma^2}$

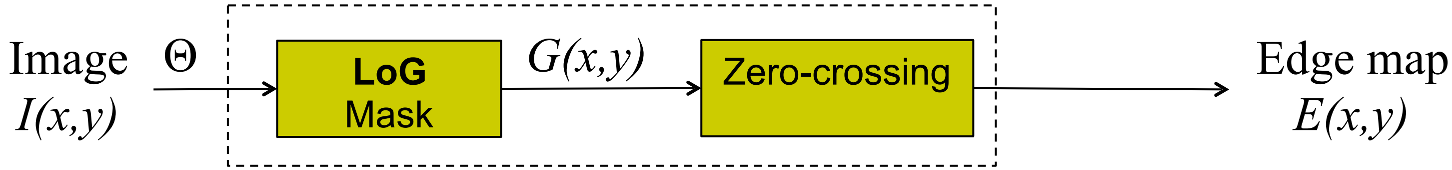
$$g''(r) = \frac{\partial^2 g(r)}{\partial r^2} = \frac{-1}{\sigma^2}[1 - \frac{r^2}{\sigma^2}]e^{-r^2/2\sigma^2}$$

From this function, we can get a LoG mask: $\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$



Then, we can apply directly on the input image *I(x,y)* .

Edge detection with LoG



$$E(x,y) = \begin{cases} 1 & G(x,y) = 0 \\ 0 & otherwise \end{cases}$$

Edge detection: *I*(*x,y*) → *E*(*x,y*)

Θ: convolve

Step 1:

- Convolve the LoG mask *g* with the image *I(x,y)* → *G*(*x,y*)

Step 2:

- Apply the Zero-crossing:

MATLAB Functions

- List of useful functions for spatial operations:
  - conv2
  - imnoise, rand, randn
  - imfilter
  - edge