画像信号処理特論

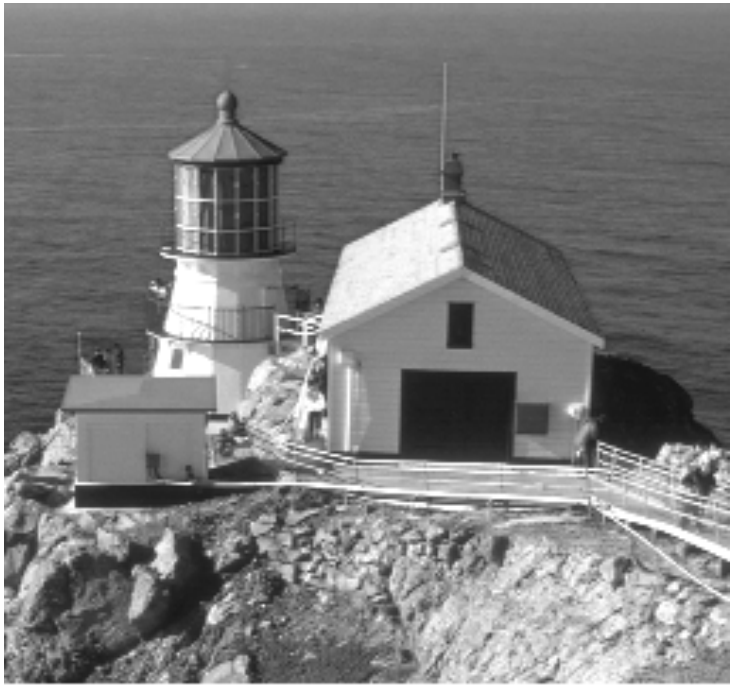
2020年度

高橋桂太

Today’s contents

Today's Issue

- Mean filtering

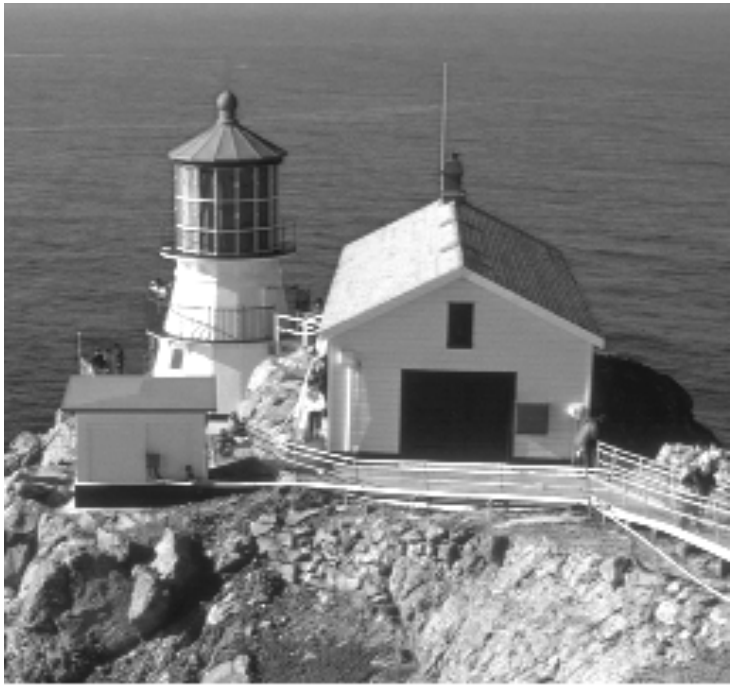


Input

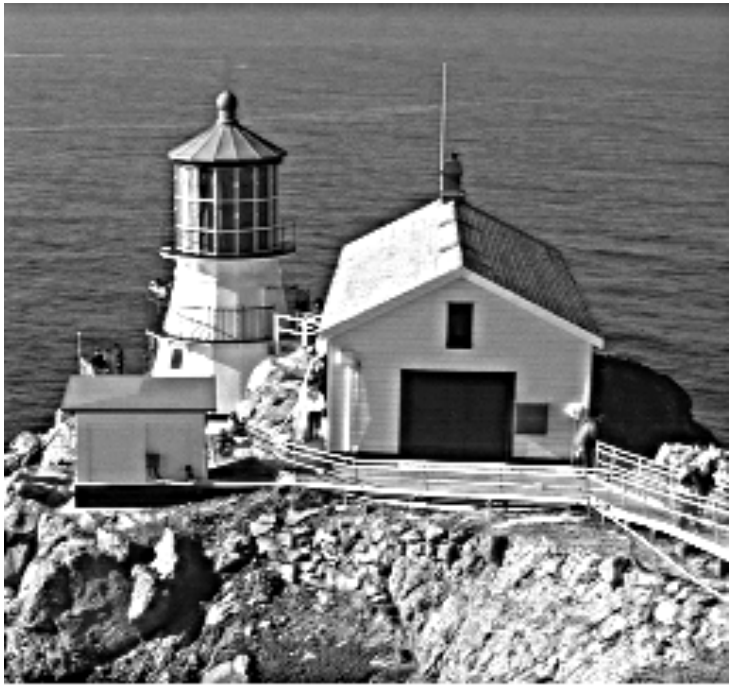
Output

Today's Issue

- Application of mean filtering

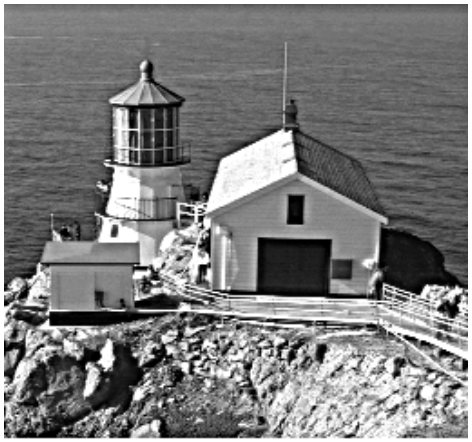


Input

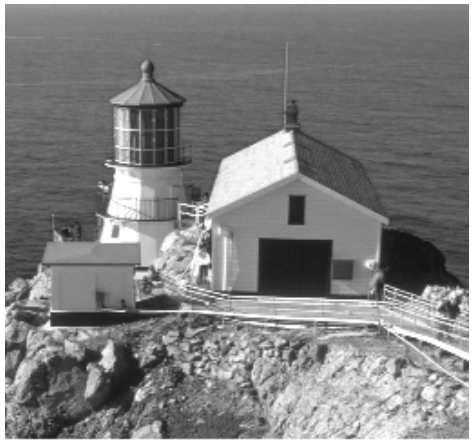


Output

Detail Enhancement (unsharp masking)



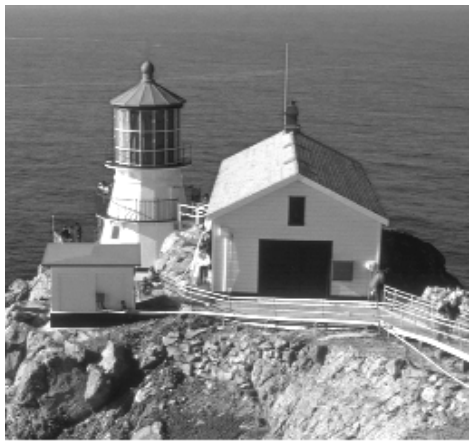
=



+

Const ×

{



−

}

Today's Issue

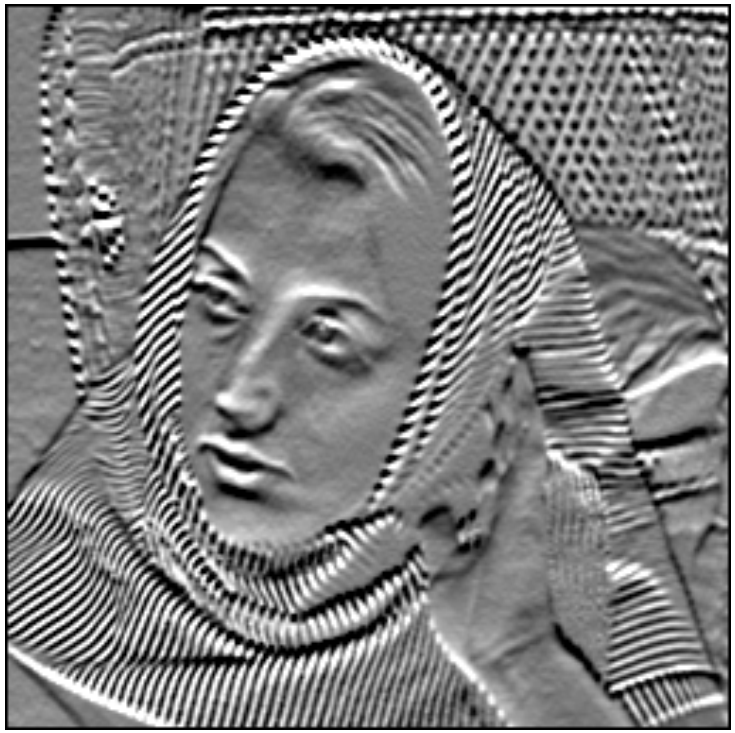
- Edge Detection

Input

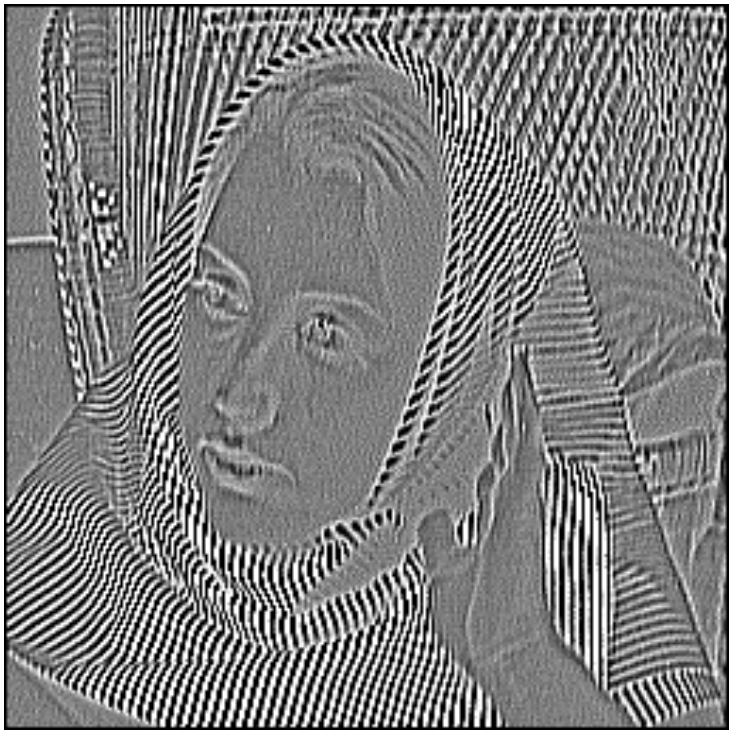
Horizontal edge

Today's Issue

- Edge Detection

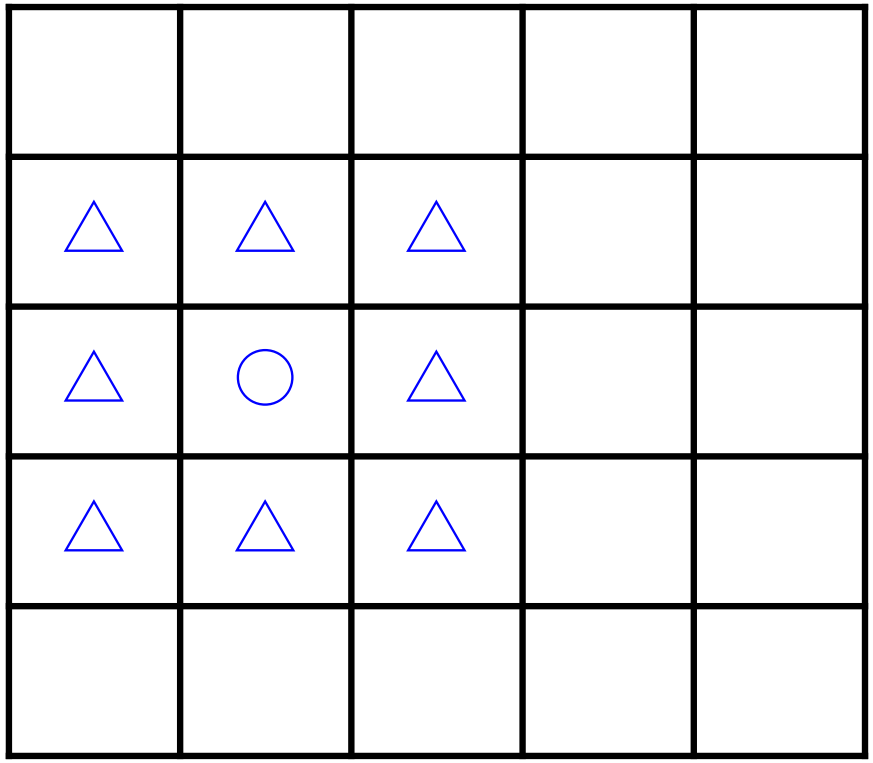


Vertical edge

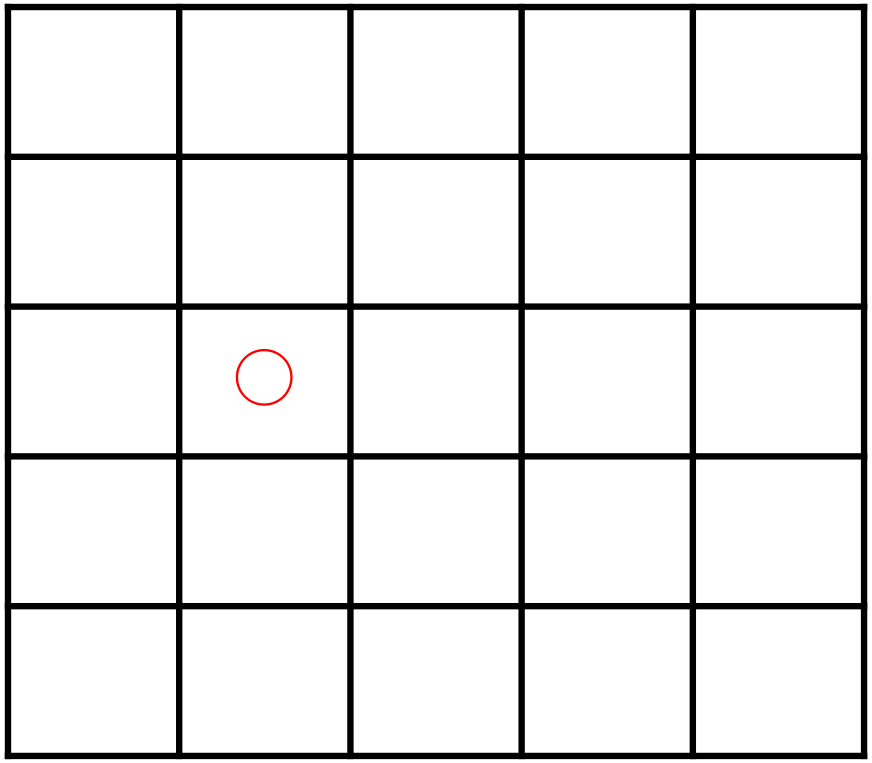


Laplacian

Principle of Mean Filtering

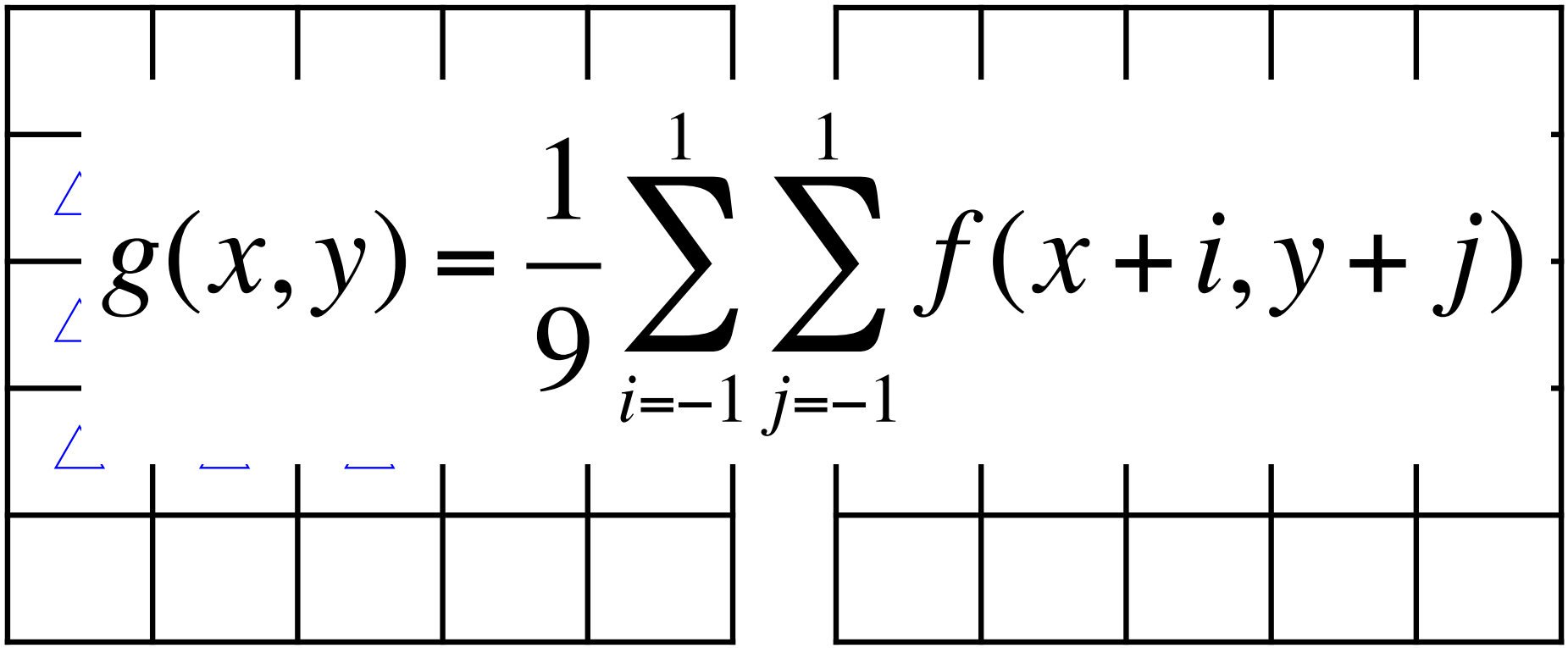


Input



Output

Principle of Mean Filtering



Implementation of image filtering

```
Loop for y (H times)
    Loop for x (W times)
        Value = 0
        Loop for i (M times)
            Loop for j (M times)
                Value += f(x+i, y+j)
        g(x,y) = Value/(M*M)
```

Take care when processing image boundaries

Why mean filtering blurs images?

- Let us consider a 1-dimentional signal.

$$g(x) = \left(f(x-1) + f(x) + f(x+1) \right) / 3$$

Applying Fourier transform to both sides…

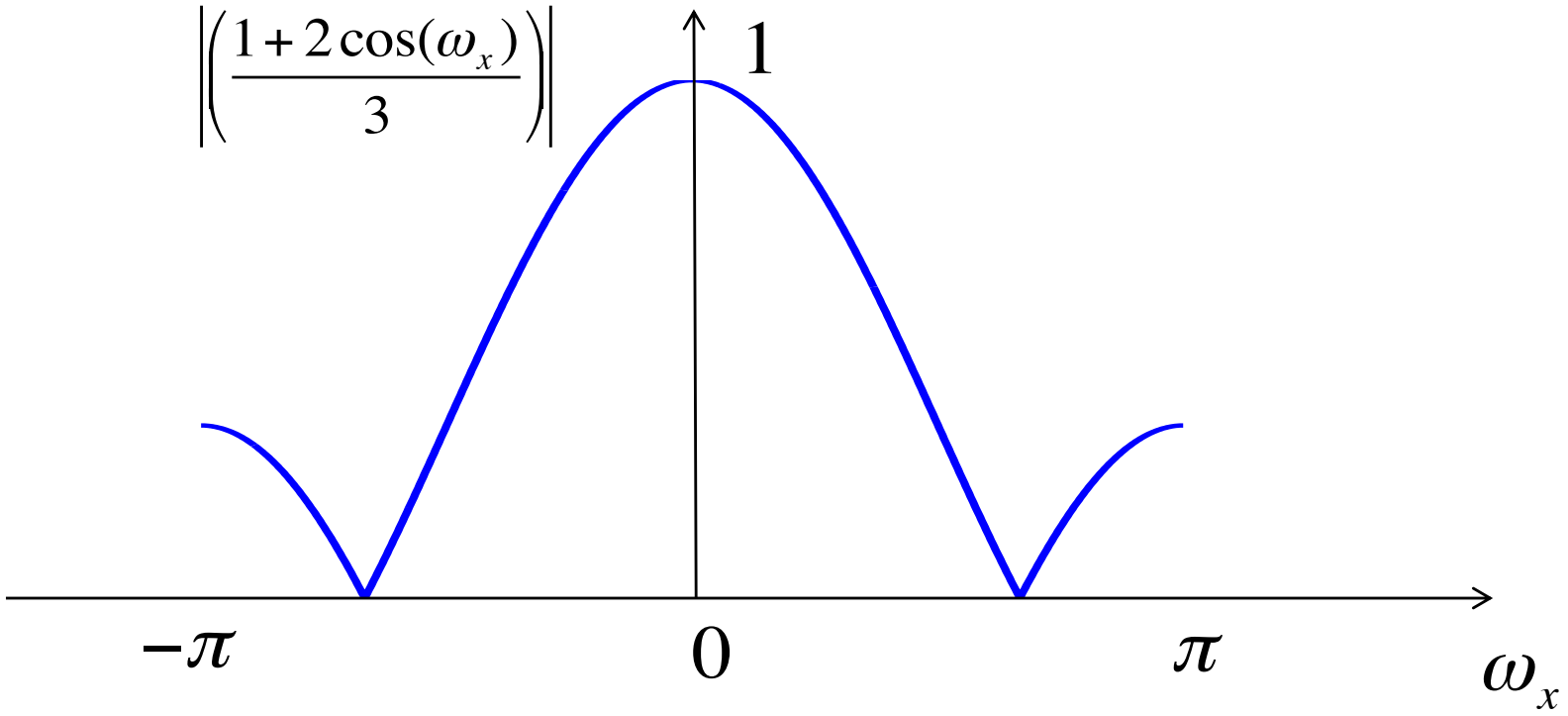
$$g(x) \Leftrightarrow G(\omega_x) \qquad f(x) \Leftrightarrow F(\omega_x)$$

$$f(x+\alpha) \Leftrightarrow F(\omega_x) \exp\left(j\omega_x \alpha \right)$$

Why mean filtering blurs images?

$$g(x) = \left(f(x-1) + f(x) + f(x+1) \right) / 3$$

$$\Rightarrow \quad G(\omega_x) = F(\omega_x) \left(\frac{1 + 2\cos(\omega_x)}{3} \right)$$

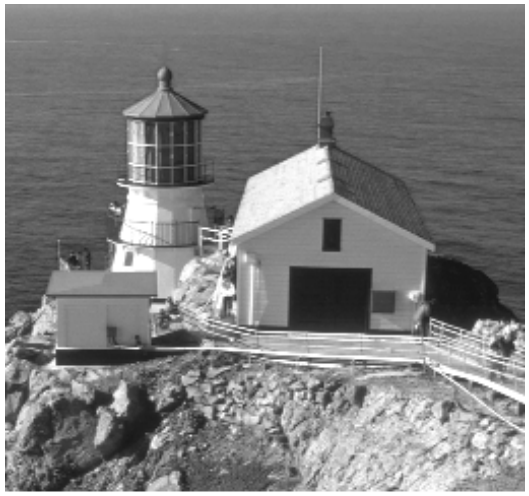


Representation of general filters

Can be written as convolution with a kernel

Kernel

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



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

Representation of other 3x3 filters

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Mean

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

Sobel

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

Sobel

1	1	1
1	-8	1
1	1	1

Laplacian

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

Any filter kernel is applicable in the same manner

Edge detection with 3x3 filters

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

Sobel

$$h * \approx \frac{\partial}{\partial x}$$

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

Sobel

$$h * \approx \frac{\partial}{\partial y}$$

1	1	1
1	-8	1
1	1	1

0	1	0
1	-4	1
0	1	0

Laplacian

$$h * \approx \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

Implementation using kernel

kernel[3][3]

Loop for y (H times)

Loop for x (W times)

Value = 0

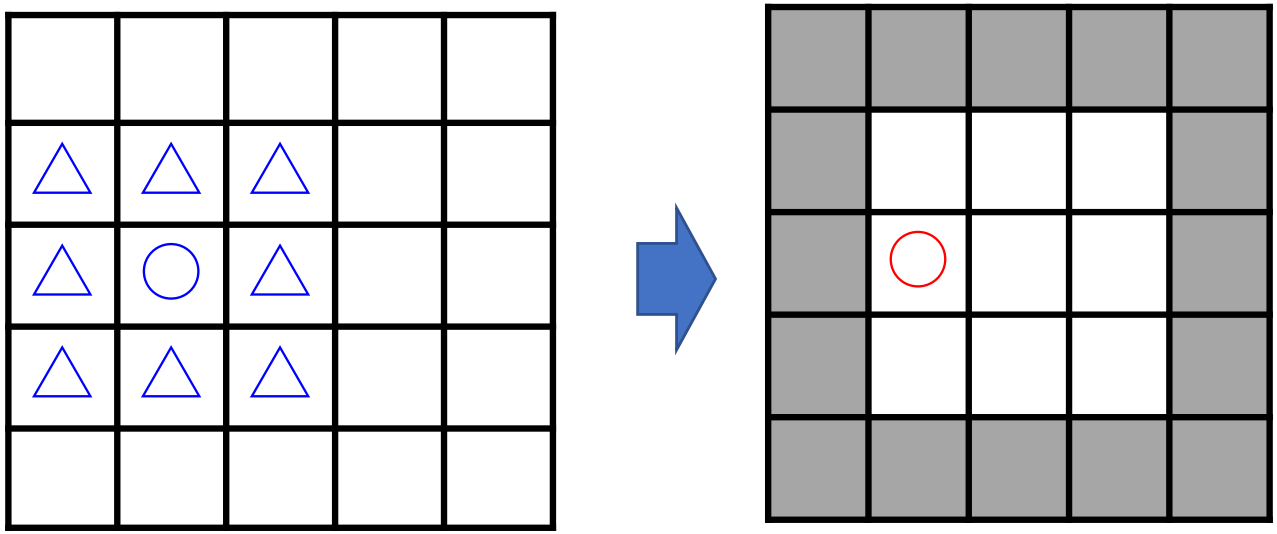
Loop for i (M times)

Loop for j (M times)

Value += f(x+i, y+j)*kernel[j+1][i+1]

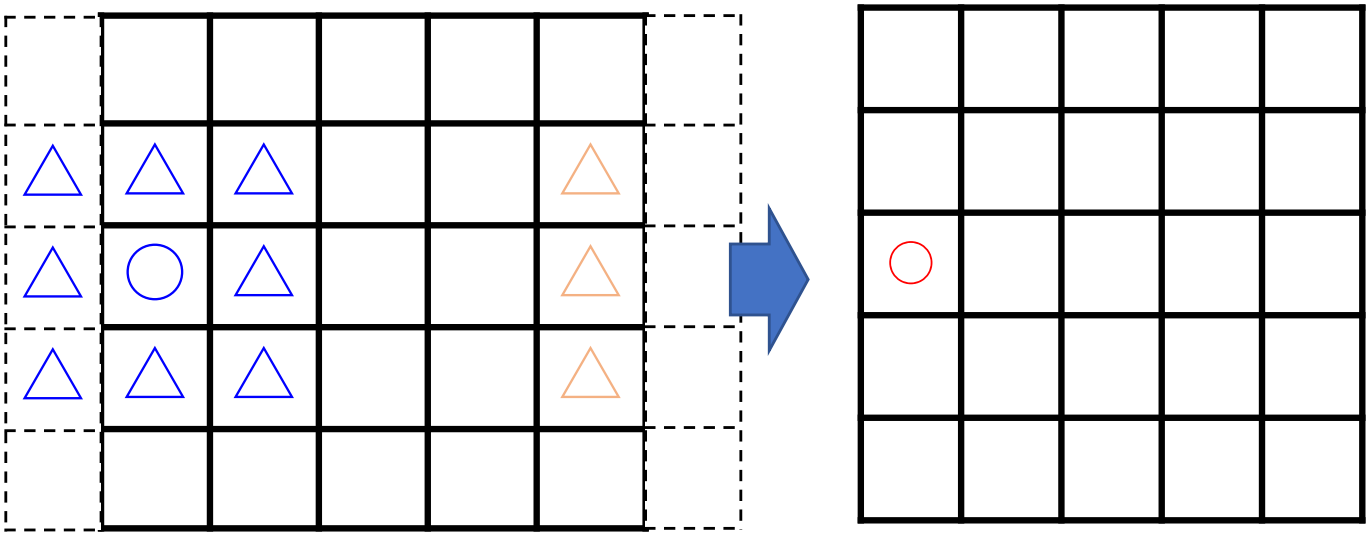
Boundary processing

- Just ignore (exclude) the boundary pixels



Boundary processing

- Extend the boundaries assuming …
  - Pixel values are zero outside the boundary
  - Pixel values are kept constant across the boundary
  - Image signal is repetitive (consistent with DFT)



Exercises

- Build and execute “sample2”
- Implement image filters
  - Mean, horizontal/vertical edge detection, Laplacian
- Implement detail enhancement
  - (original) + (original) – (mean)
  - (original) + (edge)