Intelligent Control Systems

Image Processing (2)
— Point Operations and Local Spatial Operations —

Shingo Kagami
Graduate School of Information Sciences,
Tohoku University
swk(at)ic.is.tohoku.ac.jp

http://www.ic.is.tohoku.ac.jp/ja/swk/

Image Processing Classification

input	output	example
image	image (2-D data)	image to image processing Fourier trans., label image
	1-D data	projection, histogram
	scalar values	position, recognition
image sequence	image sequence	motion image processing
	image	
	1-D data	
	scalar	

Outline

- Image to Image Processing
  - Point Operations
  - Local Operations

Image to Image

image $\{ F_{x,y} \}$ $\longrightarrow$ image $\{ G_{x,y} \}$

point operation

$G_{i,j}$ depends only on $F_{i,j}$

local operation / neighboring operation

$G_{i,j}$ depends on pixels within some neighborhood of $F_{i,j}$

global operation

$G_{i,j}$ depends on almost all the pixels in $\{ F_{i,j} \}$

Point Operation Examples

pixel value conversion, color conversion

- e.g.: binarization, pixel value inversion, gamma correction

```
cv::Mat input, output1, output2;

...

cv::threshold(input, output1, 128, 255, THRESH_BINARY);
cv::equalizeHist(input, output2);
```

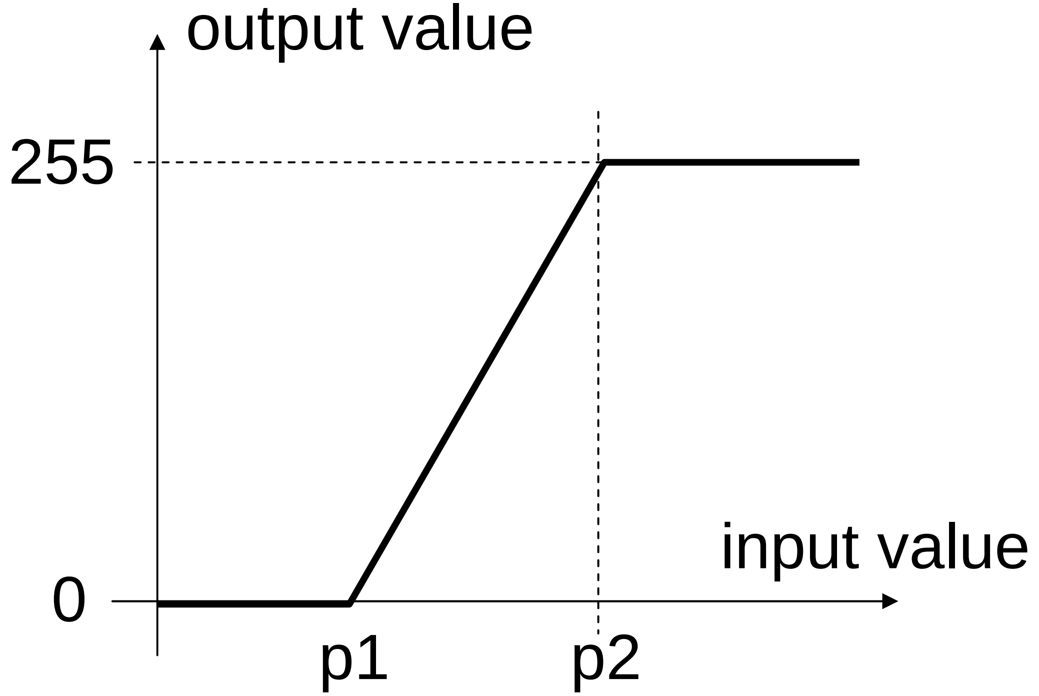
(Handling of color images will be explained next week)

Implementation

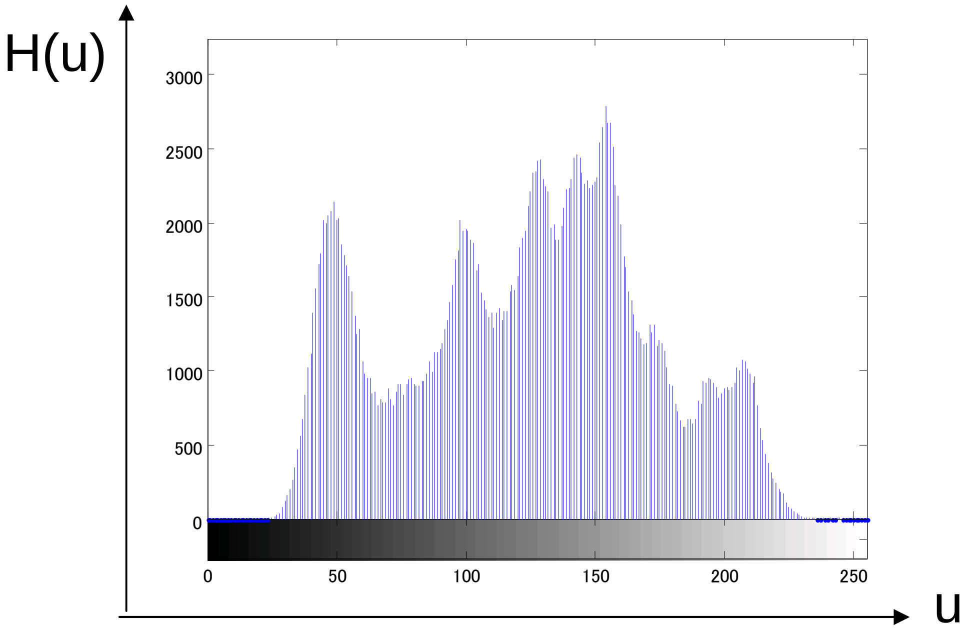
```
 for (j = 0; j < height; j++) {
     for (i = 0; i < width; i++) {
         output.at<uchar>(j, i)
              = some_func(input.at<uchar>(j, i));
     }
 }
 …

uchar some_func(uchar val) {
     …
}
```

Note again: `img.at<uchar>(j, i)` is an 8-bit value of the pixel at x = i, y = j

Pixel value conversion example



Histogram (of pixel values)



$$\boldsymbol{H} = \{H_u\}_{u=1,2,\cdots,m}, \;\; H_u = \sum_{\boldsymbol{x} \in S(u)} 1$$

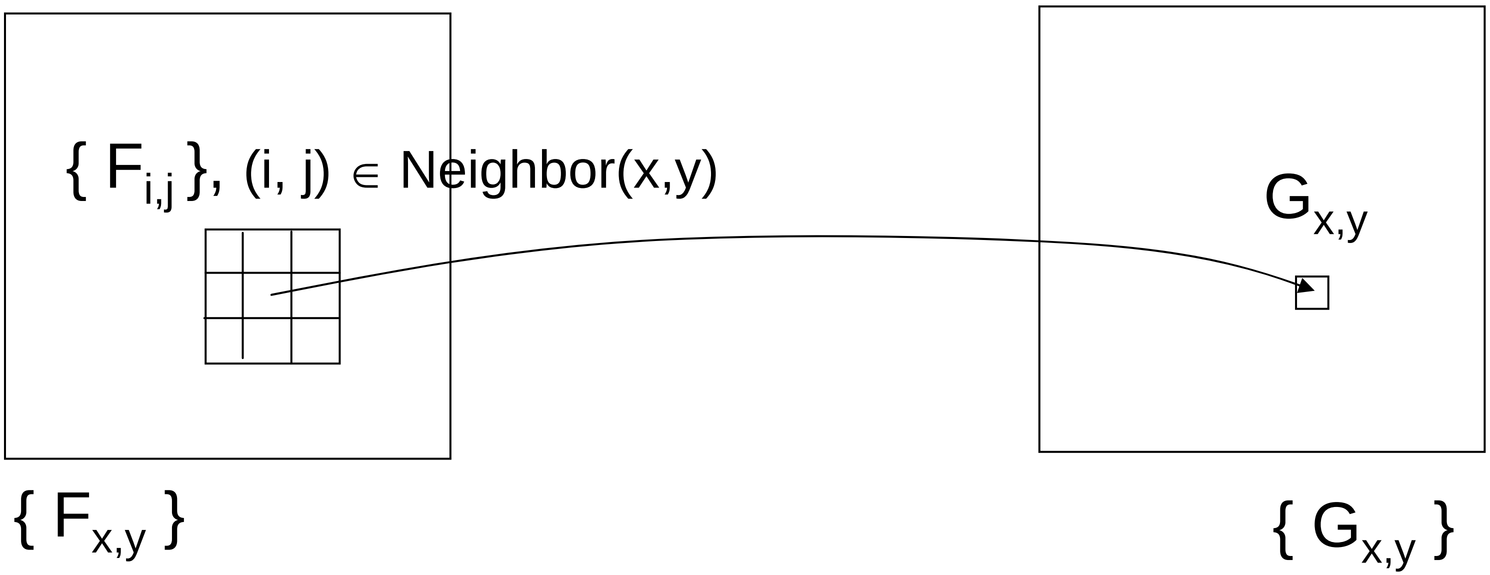
where S(u) is a set of pixels having values belonging to the bin u

Outline

- Image to Image Processing
    - Point Operations
    - Local Operations

Local operation example: Spatial Filter

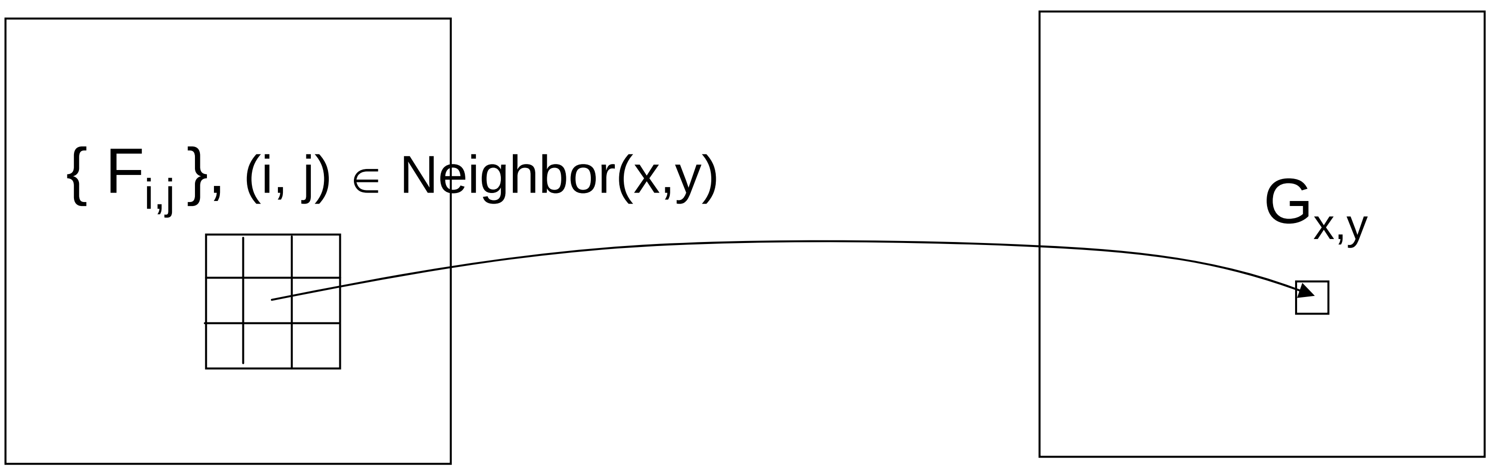
$G_{x,y}$ depends on some neighborhood (e.g. 3×3, 5×5 pixels, etc.) of the point of interest (x,y)



Typical examples: smoothing, edge detection

Important Example: Smoothing

- Output at (x, y): some representative value of the set of neighbor pixels around (x, y), e.g. mean, weighted mean, median
- Used for: e.g. noise reduction, scale-space processing



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

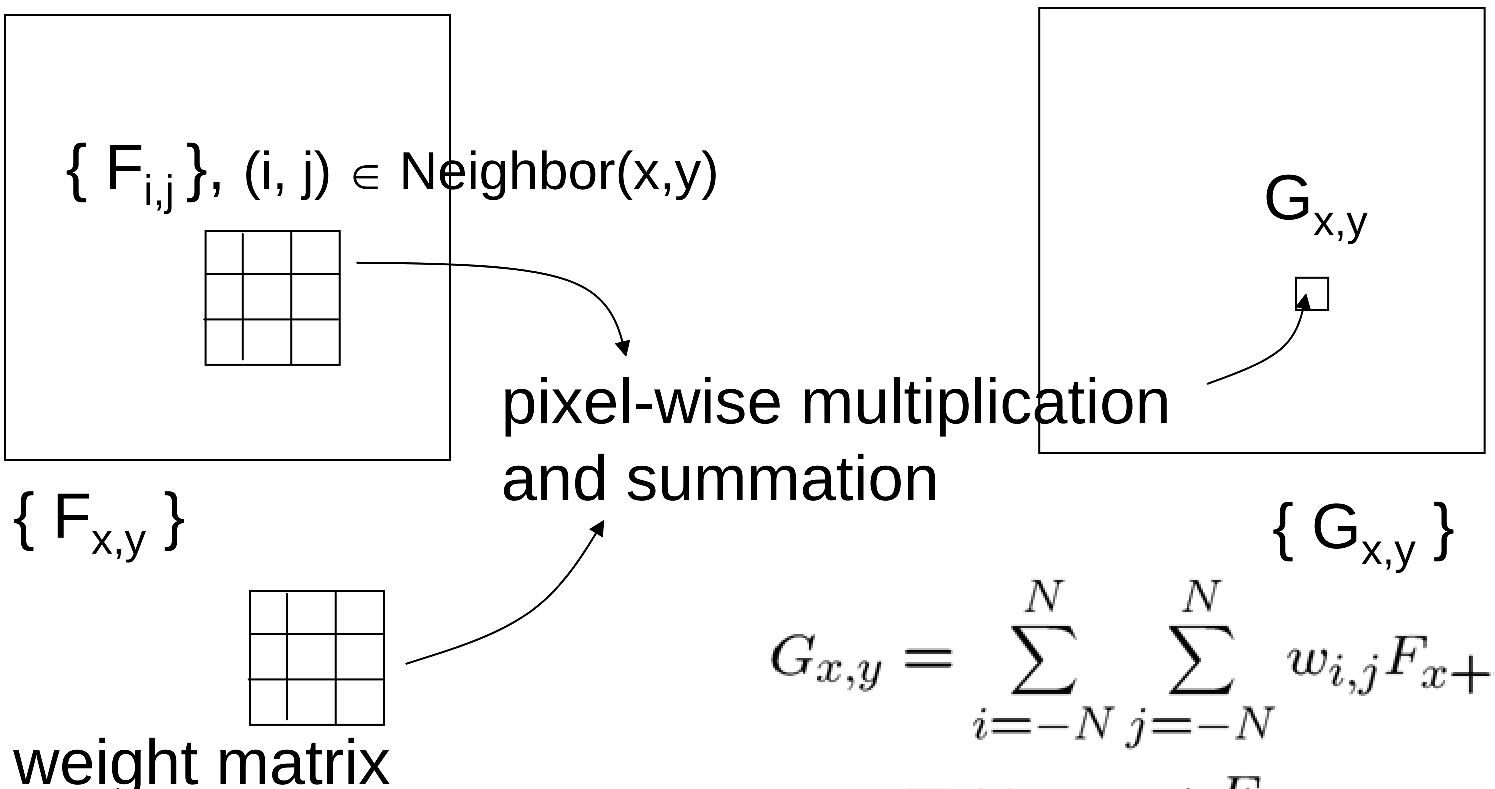
(mean)

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

(weighted mean)

Linear Spatial Filtering

- Smoothing with (weighted) mean is an example of linear spatial filtering (while smoothing with median is nonlinear)
- Computed by convolving a weight matrix (filter coefficients, filter kernel, or mask) to input image



$$G_{x,y} = \sum_{i=-N}^{N} \sum_{j=-N}^{N} w_{i,j} F_{x+i,y+j}$$
$$= w_{-x,-y} * F_{x,y}$$

Examples of 3x3 smoothing weight matrices

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

=

1/9

1	1	1
1	1	1
1	1	1

1/10	1/10	1/10
1/10	1/5	1/10
1/10	1/10	1/10

=

1/10

1	1	1
1	2	1
1	1	1

0	1/8	0
1/8	1/2	1/8
0	1/8	0

=

1/8

0	1	0
1	4	1
0	1	0

Linear filtering example

```
cv::Mat weight = (cv::Mat_<float>(3,3) <<
                      0, 1, 0,
                      1, 4, 1,
                      0, 1, 0);
weight = weight / 8.0f;

…
cv::filter2D(input, output, CV_8U, weight);
```

convenient way of matrix initialization

Or, very common filters are readily available

```
cv::GaussianBlur(input, output, cv::Size(0, 0), 10.0);
cv::Sobel(input, output, CV_8U, 1, 0);
cv::Laplacian(input, output, CV_8U);
```

Implementation of 3x3 linear filtering

```
cv::Mat input, output;
cv::Mat weight = (cv::Mat_<int>(3,3) <<
    0, 1, 0,
    1, 4, 1,
    0, 1, 0);
int normalizer = 8;

for (j = 1; j < height - 1; j++) {
    for (i = 1; i < width - 1; i++) {
        int sum = 0;
        for (n = 0; n < 3; n++) {
            for (m = 0; m < 3; m++) {
                sum += weight.at<int>(n, m)
                       * input.at<uchar>(j - 1 + n, i - 1 + m);
            }
        }

        output.at<uchar>(j, i) = saturate(sum / normalizer);
    }
}
```

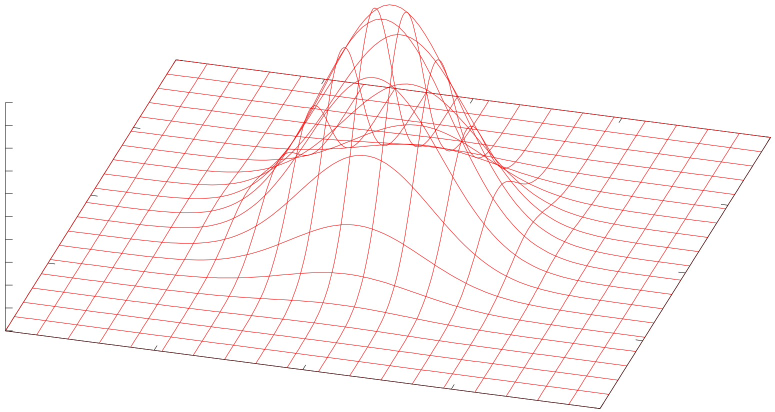
boundary handling
(just forced to be zero)

Note: center coordinate of weight is not (0, 0) but (1, 1)

make sure the pixel value is in [0, 255]

Gaussian: most widely used smoothing kernel

$$g_\sigma(x,y) = \frac{1}{\sqrt{2\pi\sigma}}\exp\{-\frac{x^2}{2\sigma^2}\} \cdot \frac{1}{\sqrt{2\pi\sigma}}\exp\{-\frac{y^2}{2\sigma^2}\}$$
$$= \frac{1}{2\pi\sigma^2}\exp\{-\frac{x^2+y^2}{2\sigma^2}\}$$



- Discretized in space for computation
- Coefficient values are sometimes rounded to integer (for efficiency)
- Amount of smoothing can be controlled by parameter σ (large σ requires large matrix size)

Edge Detection

- Spatial differentiation (approximated by finite difference)

0	0	0
-1	0	1
0	0	0

1st order diff. in x direction

0	-1	0
0	0	0
0	1	0

1st order diff. in y direction

- Often combined with smoothing:

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

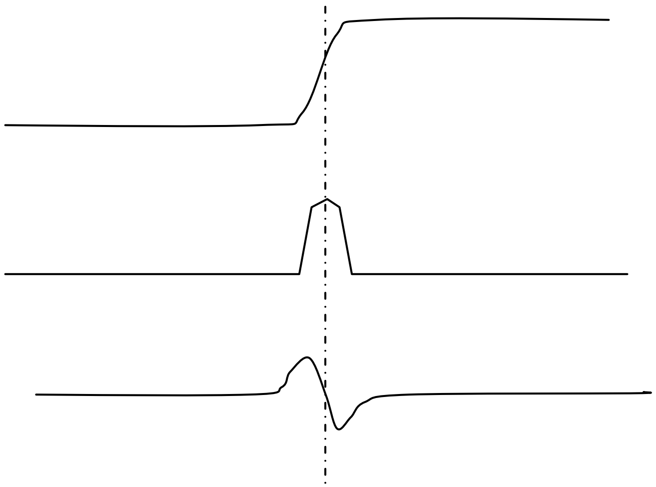
Sobel filter in x direction

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

Sobel filter in y direction

Edge detection by 2nd order derivative

- Edge = zero crossing of 2nd order derivative
- Laplacian $\partial^2/\partial x^2 + \partial^2/\partial y^2$ is the lowest-order isotropic differential operator
  - does not depend on direction of edges

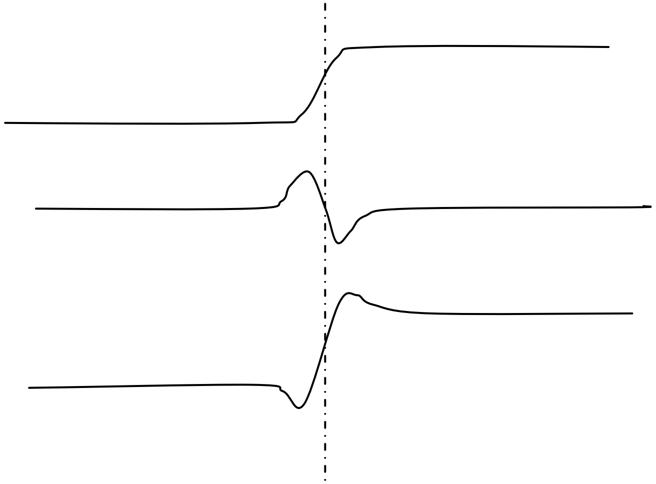


- Laplacian operator is realized by adding 2nd order differentials $f_{i+1} - 2 f_i + f_{i-1}$ of x and y directions

0	1	0
1	–4	1
0	1	0

Sharpening

Subtract the Laplacian image from the original image to yield an edge-enhanced image



$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \;-\; \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \;=\; \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

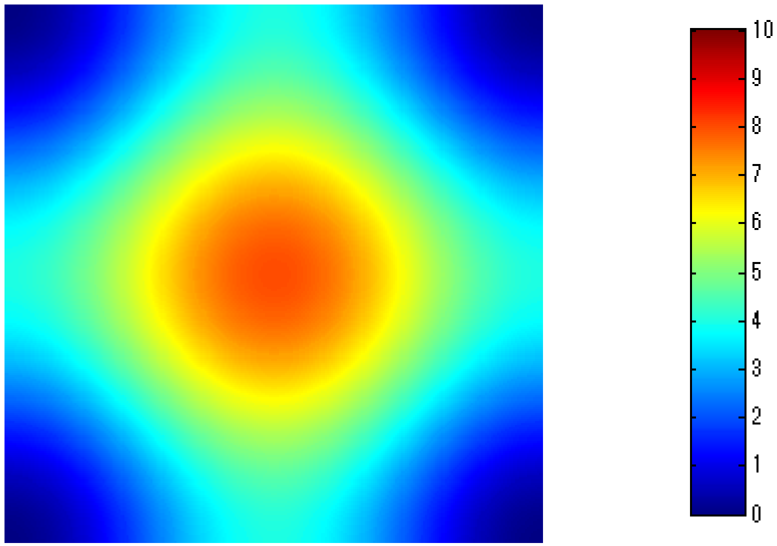
Frequency-domain understanding

$$G_{x,y} = \sum_{i=-N}^{N} \sum_{j=-N}^{N} w_{i,j} F_{x+i,y+j}$$
$$= w_{-x,-y} * F_{x,y} \xrightarrow{\mathcal{F}} \mathcal{F}\left[w_{-x,-y}\right] \cdot \mathcal{F}\left[F_{x,y}\right]$$

$\mathcal{F}[\cdot]$: 2-D discrete Fourier transform

0	1	0
1	4	1
0	1	0

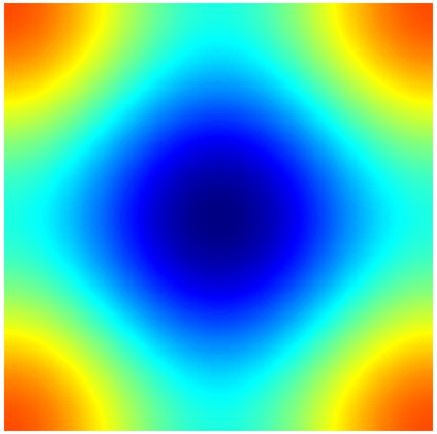
$\xrightarrow{\mathcal{F}}$



Recall: Fourier transform of Gaussian function is Gaussian

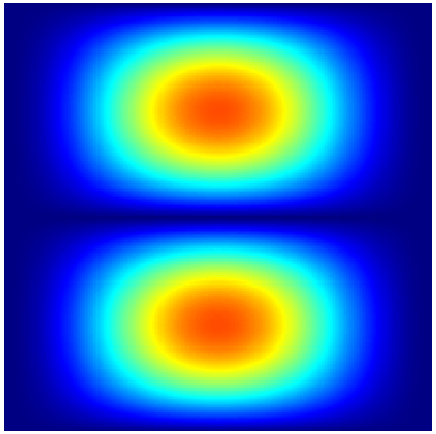
0	1	0
1	-4	1
0	1	0

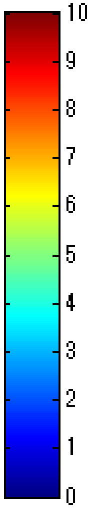
$\xrightarrow{\mathcal{F}}$



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

$\xrightarrow{\mathcal{F}}$



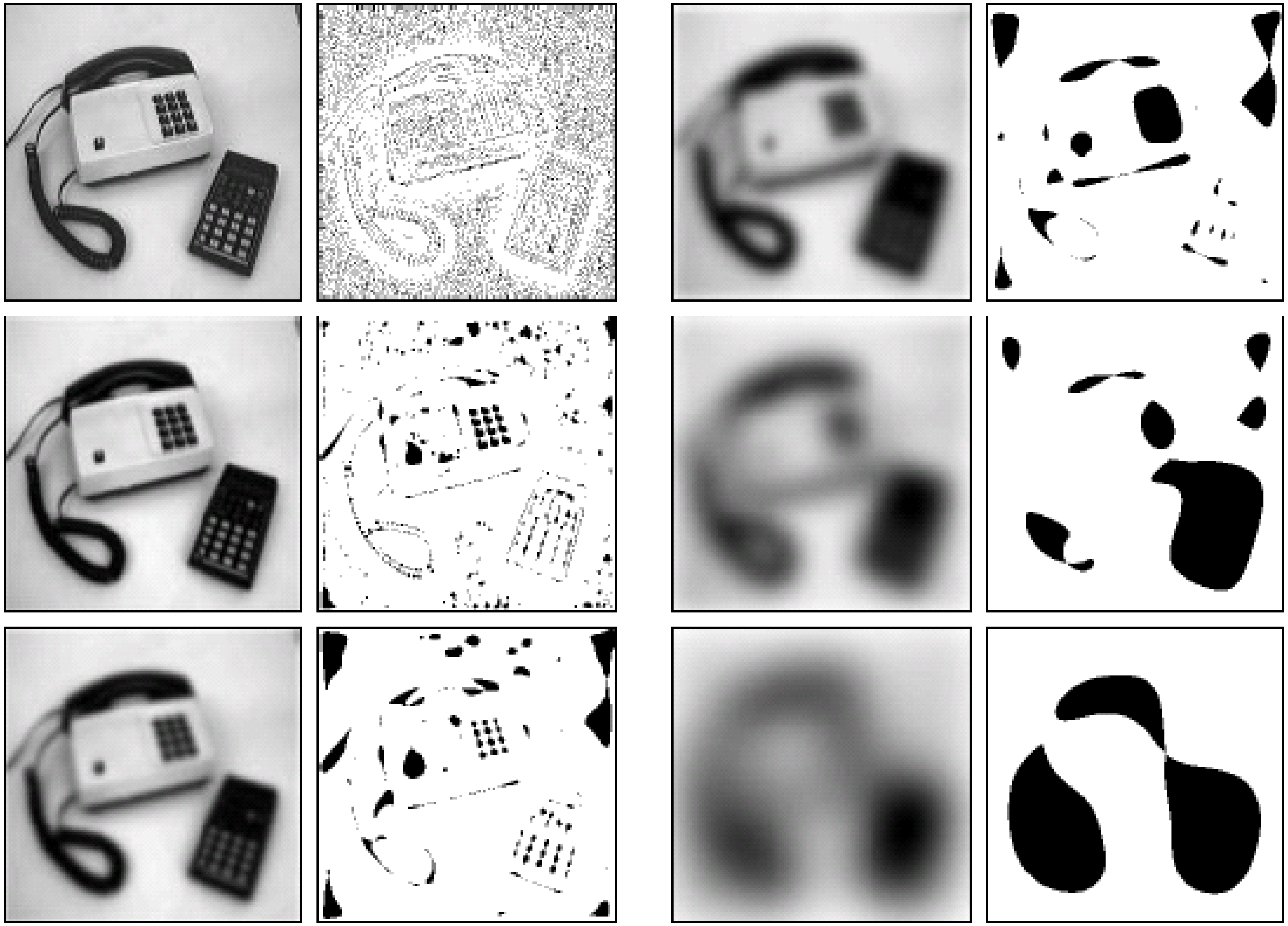


Why Gaussian is preferred for smoothing

Several explanations are possible. For example:

- When a scale space $I(x, y; t)$ in which t is a measure of smoothing amount (scale parameter) is considered, imposing some reasonable assumptions singles out Gaussian as the unique smoothing operator: e.g. (Florack et al., 1992),
  - linear and shift-invariant
  - scale-invariant
  - isotropic
  - forms a commutative semigroup, i.e. with $G(t)$ being a smoothing operator with smoothing amount t, $G(t_1) \circ G(t_2)$ is also a smoothing operator and $G(t_1) \circ G(t_2) = G(t_2) \circ G(t_1)$

Scale Space Example



[Lindeberg 1996]

Summary

- Image to Image processing
  - Point operations
    - e.g.: pixel value conversion
    - concept of histogram

  - Local operations
    - linear spatial filters (cf. nonlinear filters)
      - smoothing
      - edge detection
      - sharpening
    - frequency domain understanding
      - concept of scale space

References

- R. Szeliski: Computer Vision: Algorithms and Applications, Springer, 2010.
- A. Hornberg eds.: Handbook of Machine Vision, Wiley-VCH, 2006.
- G. Bradski and A. Kaebler: Learning OpenCV, O‘Reilly, 2008.
- OpenCV Documentation: http://docs.opencv.org/index.html

- T. Lindeberg: Scale-space: A framework for handling image structures at multiple scales, Proc. CERN School of Computing, 1996.
- L. M. J. Florack, B. M. T. Romeny, J. J. Koenderink and M. A. Viergever: Scale and the Differential Structure of Images, Image and Vision Computing, vol.10, no.6, pp.376-388, 1992.

(in Japanese)

- ディジタル画像処理編集委員会, ディジタル画像処理, CG-ARTS協会, 2004.
- 田村: コンピュータ画像処理, オーム社, 2002.

Sample codes are in sample20140617.zip available at
http://www.ic.is.tohoku.ac.jp/~swk/lecture/