知能制御システム学

画像処理の基礎 (2)
— OpenCV による基本的な例 —

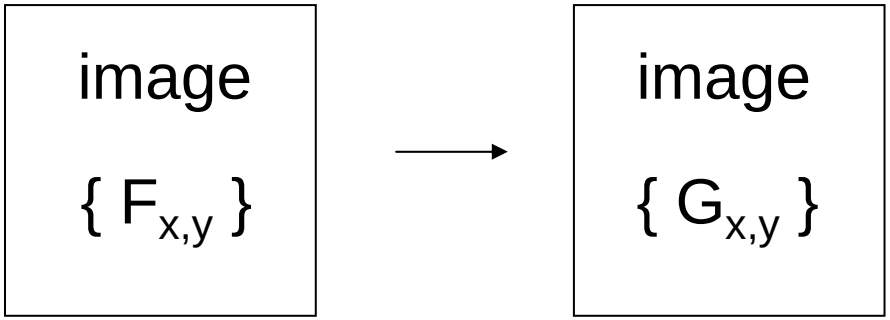
東北大学 大学院情報科学研究科
鏡 慎吾
swk(at)ic.is.tohoku.ac.jp

2008.07.01

画像処理の分類

input	output	
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 (2-D data)	motion image processing
	...	

画像から画像への変換



point operation (点処理)

$G_{i,j}$ が $F_{i,j}$ にのみ依存

local operation / neighboring operation (局所処理)

$G_{i,j}$ が $\{F_{i,j}\}$ のうち(i, j)の近傍の画素のみに依存

global operation (大域処理)

$G_{i,j}$ が $\{F_{i,j}\}$ の(ほぼ)すべての画素に依存

点処理の例

2値化 (binarization) [前回の授業でも紹介]

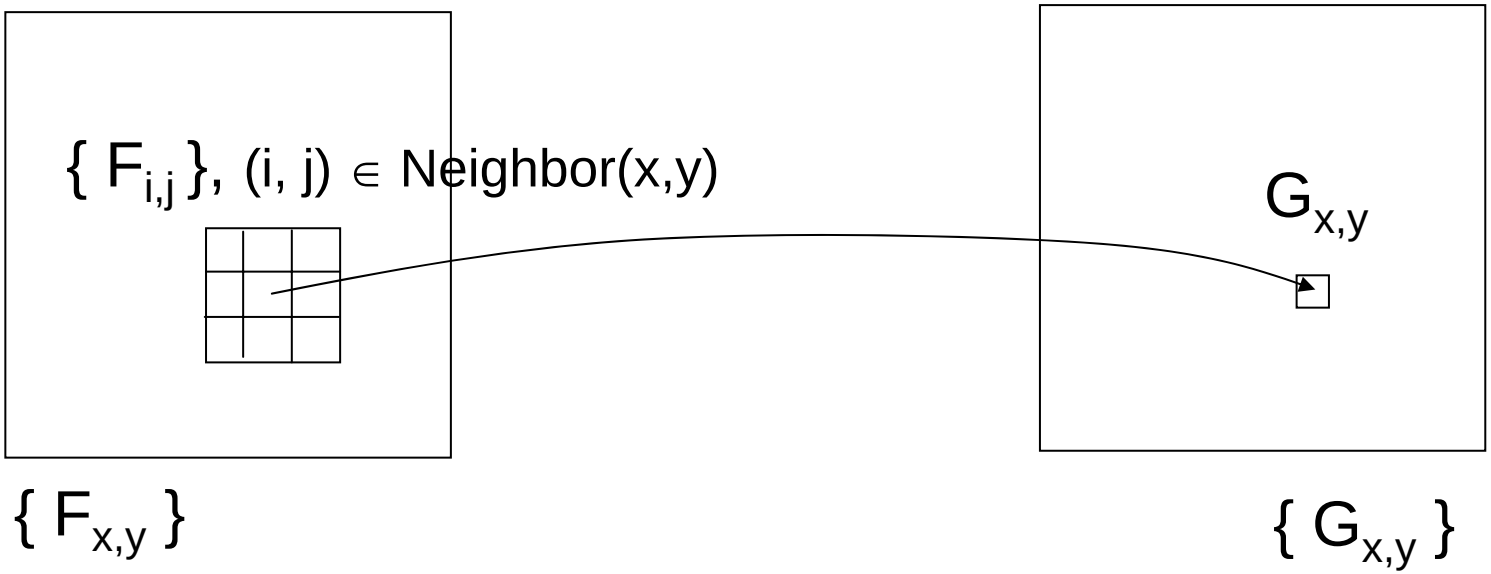
サンプルプログラム: binarize.cpp
(実用上は，cvThreshold() を使うとよい)

```
for (j = 0; j < N; j++) {
    for (i = 0; i < M; i++) {
        if (image[M * j + i] >= 128) {
            image[M * j + i] = 255;
        } else {
            image[M * j + i] = 0;
        }
    }
}
```

他の例: 濃度値変換（ヒストグラム平坦化，ガンマ補正など）

局所処理の例 — 空間フィルタリング

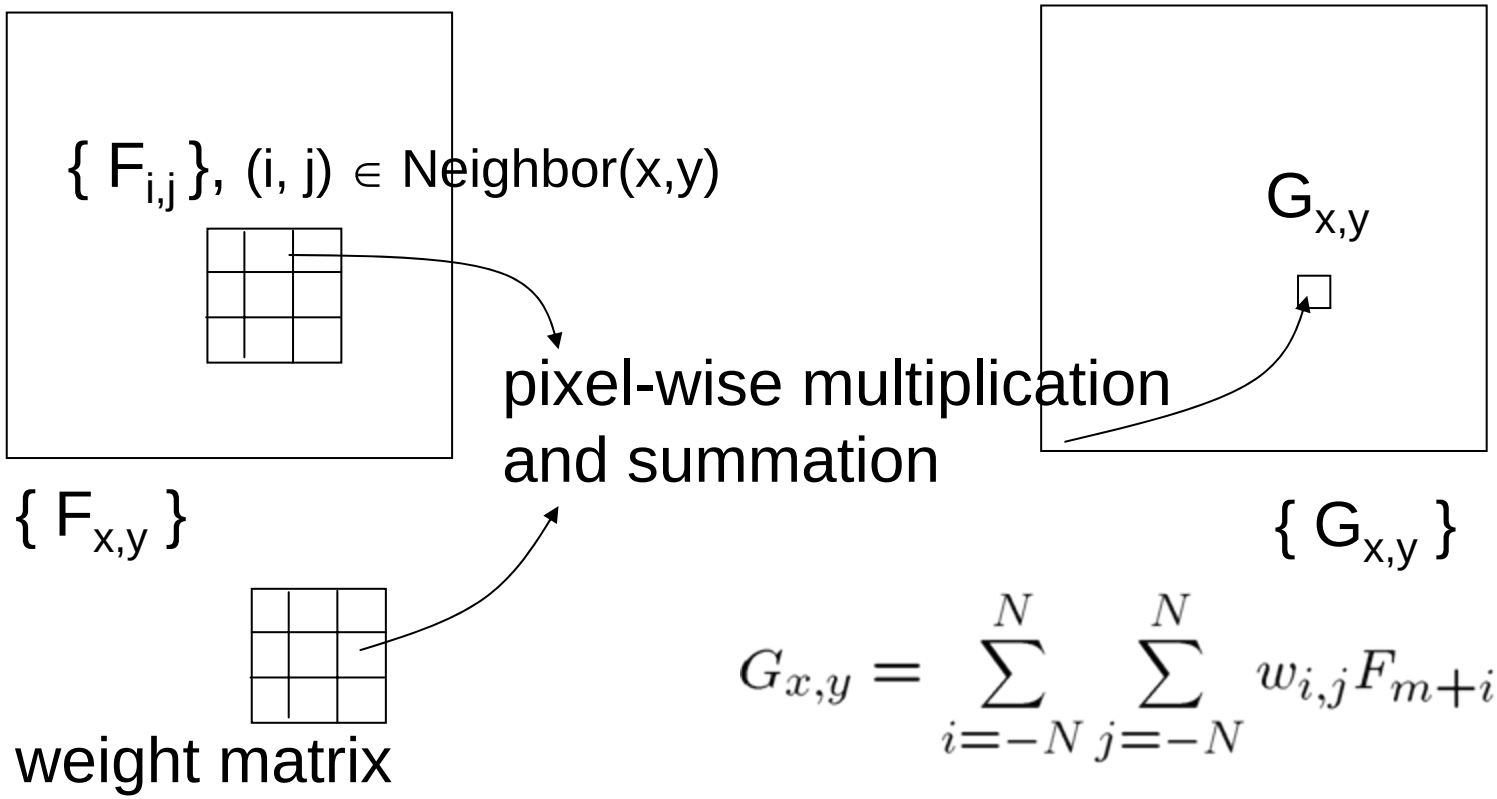
注目点の近傍（典型的には3x3画素，5x5画素, ... など）の画素値から，出力 $G_{x,y}$ を定める



典型例: 平滑化 (smoothing)，エッジ検出 (edge detection)

線形空間フィルタリング

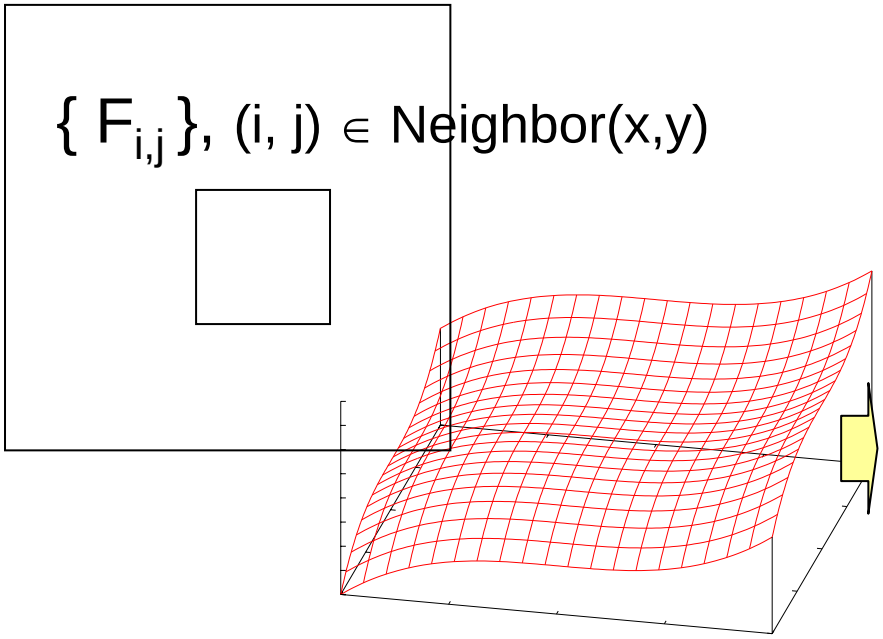
- 空間フィルタのうち，線形でシフト不変なもの線形空間フィルタ，あるいは単に線形フィルタと呼ぶ
- 加重マトリックス（マスク，カーネル，フィルタ係数，あるいは単にフィルタとも呼ばれる）のたたみこみとして表すことができる



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

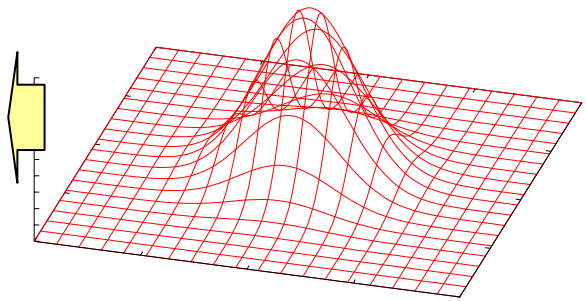
典型かつ重要な例: 平滑化 (smoothing)

- ノイズを除去したいときや，微細な構造を無視したいときに適用
- 近傍の画素値集合の代表値（平均など）を出力とする
- 適当な仮定のもとで，最良な平滑化関数は「ガウス関数のたたみこみ」であることが示される

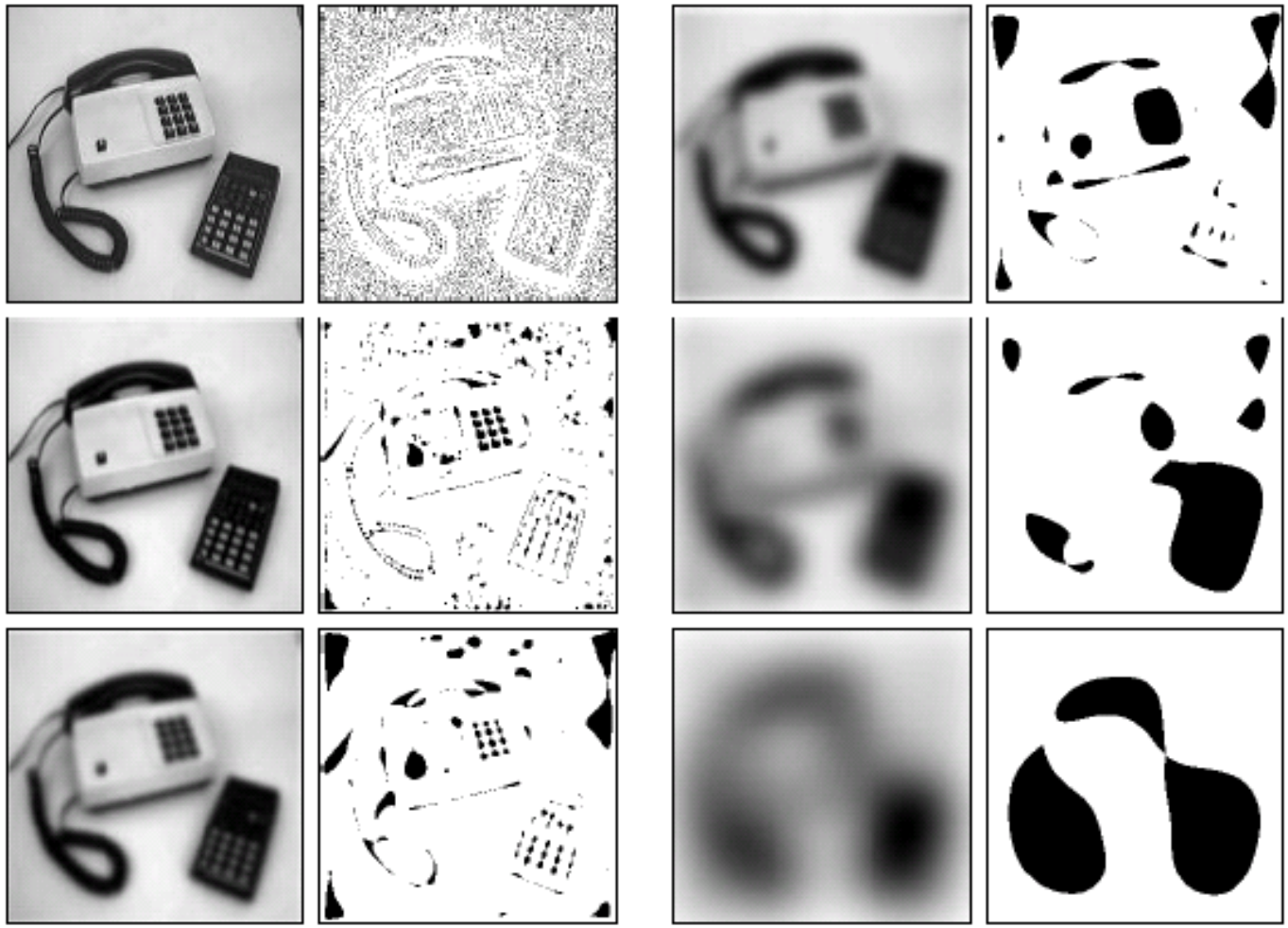


$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2+y^2}{2\sigma^2}$$

画素ごとの積を総和



ガウス関数のたたみこみの例



[Lindeberg 1996]

3x3線形フィルタのコード例

```
double w[3 * 3] = { 0.0,  1.0,  0.0,
                    1.0, -4.0,  1.0,
                    0.0,  1.0,  0.0 };
double scaling = 1.0;

for (j = 1; j < N - 1; j++) {
    for (i = 1; i < M - 1; i++) {
        double sum = 0.0;
        for (n = 0; n < 3; n++) {
            for (m = 0; m < 3; m++) {
                sum += PIXVAL(img, i - 1 + m, j - 1 + n)
                       * w[m + 3 * n];
            }
        }
        PIXVAL(result, i, j) = REAL2UCHAR(scaling * sum);
    }
}
```

サンプルプログラム: filter3x3.cpp
(実用上は，cvSobel(), cvLaplace(), cvSmooth(),
cvFilter2D() などを使い分けるとよい)

- 画像の読み出しの際に，座標範囲を越えてしまわないように注意する必要がある
- 加重は一般には実数．計算後の結果が格納先の画素値の型に収まるように注意する必要がある

加重マトリックスの例

Smoothing (平滑化)

1	1	1
1	1	1
1	1	1

1	1	1
1	2	1
1	1	1

0	1	0
1	4	1
0	1	0

Sharpening (先鋭化)

0	-1	0
-1	5	-1
0	-1	0

-1	-1	-1
-1	9	-1
-1	-1	-1

加重マトリックスの例

Edge Detection

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

単純な水平方向
1次微分

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

単純な垂直方向
1次微分

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

注目画素の近くに重み
(Sobel フィルタ)

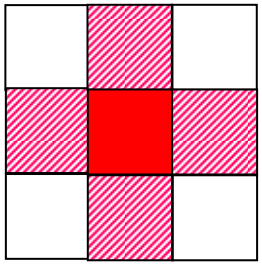
2値画像処理

2値化された後の画像 (binary image) の処理は，それだけで一分野をなすほど独特に発展している.

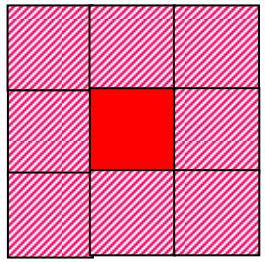
- 実用上重要であった
- 幾何学的に明確な議論がしやすく，体系的な議論が進んだ

連結性

近傍 (neighbor): ある画素の近くにある画素の集合. いろいろな定義が可能.



4-neighbor



8-neighbor

互いに n-近傍の関係にある 2 つの画素は「n-隣接している」(n-adjacent) と呼ばれる.

同じ画素値を持つ2つの画素 a, b に対して画素の系列 p_0 (= a), p_1, p_2, ..., p_{n-1}, p_n (= b) が存在し, p_i はすべて同じ画素値を持ち, p_i と p_{i-1} が n-隣接するとき, 画素 a と b は「n-連結している」(n-neighbor connected) という

数学的モルフォロジ

dilation: 近傍の誰かが 1 だったら自分も 1 になる

$$G_{i,j} = F_{i,j} \mid F_{i-1,j} \mid F_{i+1,j} \mid F_{i,j-1} \mid F_{i,j+1}$$ (4-近傍の場合)

erosion: 近傍の誰かが 0 だったら自分も 0 になる

$$G_{i,j} = F_{i,j} \,\&\, F_{i-1,j} \,\&\, F_{i+1,j} \,\&\, F_{i,j-1} \,\&\, F_{i,j+1}$$ (4-近傍の場合)

opening: erosion + dilation

closing: dilation + erosion

サンプルプログラム: morph.cpp
(実用上は, cvErode(), cvDilate() などを使うとよい)

画像から1次元データへの変換

projection (射影)

```
for (i = 0; i < img->width; i++) { hx[i] = 0; }
for (j = 0; j < img->height; j++) { hy[j] = 0; }

for (j = 0; j < img->height; j++) {
    for (i = 0; i < img->width; i++) {
        if (PIXVAL(img, i, j) > 128) {
            hx[i]++;
            hy[j]++;
        }
    }
}
```

サンプルプログラム: xyproj.cpp

画像からスカラ値への変換

モーメント特徴

$$m_{p,q} = \sum_i \sum_j i^p j^q F_{i,j}$$

$$m_{0,0} = \sum_i \sum_j F_{i,j}$$ 0次モーメント: 2値画像の場合，面積

$$m_{1,0} = \sum_i \sum_j i F_{i,j}$$ x方向の1次モーメント

$$m_{0,1} = \sum_i \sum_j j F_{i,j}$$ y方向の1次モーメント

0～1次モーメントまでの情報から，面積と重心が分かる．

$(g_x, g_y) = (m_{1,0}/m_{0,0}, m_{0,1}/m_{0,0})$

さらに高次のモーメントから，形状に関するより詳細な情報が得られる．

サンプルプログラム: moment.cpp

(実用上は，cvMoments() を用いるとよい)

References

[1] 田村: コンピュータ画像処理, オーム社, 2002.

[2] http://sourceforge.net/projects/opencvlibrary/

[3] http://opencvlibrary.sourceforge.net/

[4] T. Lindeberg: Scale-space: A framework for handling image structures at multiple scales, Proc. CERN School of Computing, 1996.