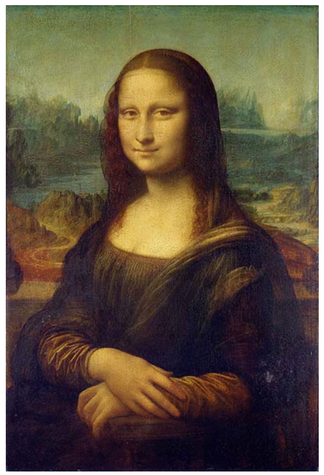
マルチメディア処理

Multimedia Information Processing

第8回：画像処理 (2) 空間フィルタリング



髙木一幸

髙木一幸 takagi@uec.ac.jp

講義概要

- マルチメディアデータ（文書、音声、画像、映像など）の表現方法と処理技術について、基礎的な内容を紹介・解説する。
- 授業時間中および宿題として、計算機を使った演習を行うことにより実際のデータの扱い方を学び、理解を深める。

第1回：授業の概要説明、
人間の感覚とマルチメディア処理（4/11）

髙木

第2回：文字・テキストの表現と処理（4/18）

第3回：音声のディジタル表現と処理（4/25）

第4回：画像・映像のディジタル表現（5/2）

第5回：マルチメディアデータの符号化とファイル形式（5/9）

第6回：2次元図形の表現と描画（5/16）

第7回：画像処理(1)画素ごとの濃淡変換（5/23）

第8回：画像処理(2)空間フィルタリング（5/30）

廣田

第9回：カメラと写真撮影（6/13）

第10回：3次元コンピュータグラフィックス（6/20）
(1)形状表現と透視投影

第11回：3次元コンピュータグラフィックス（6/27）
(2)照明効果とシェーディング

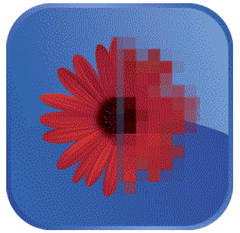
第12回：アニメーションと映像制作（7/4）

第13回：シミュレーションと可視化（7/11）

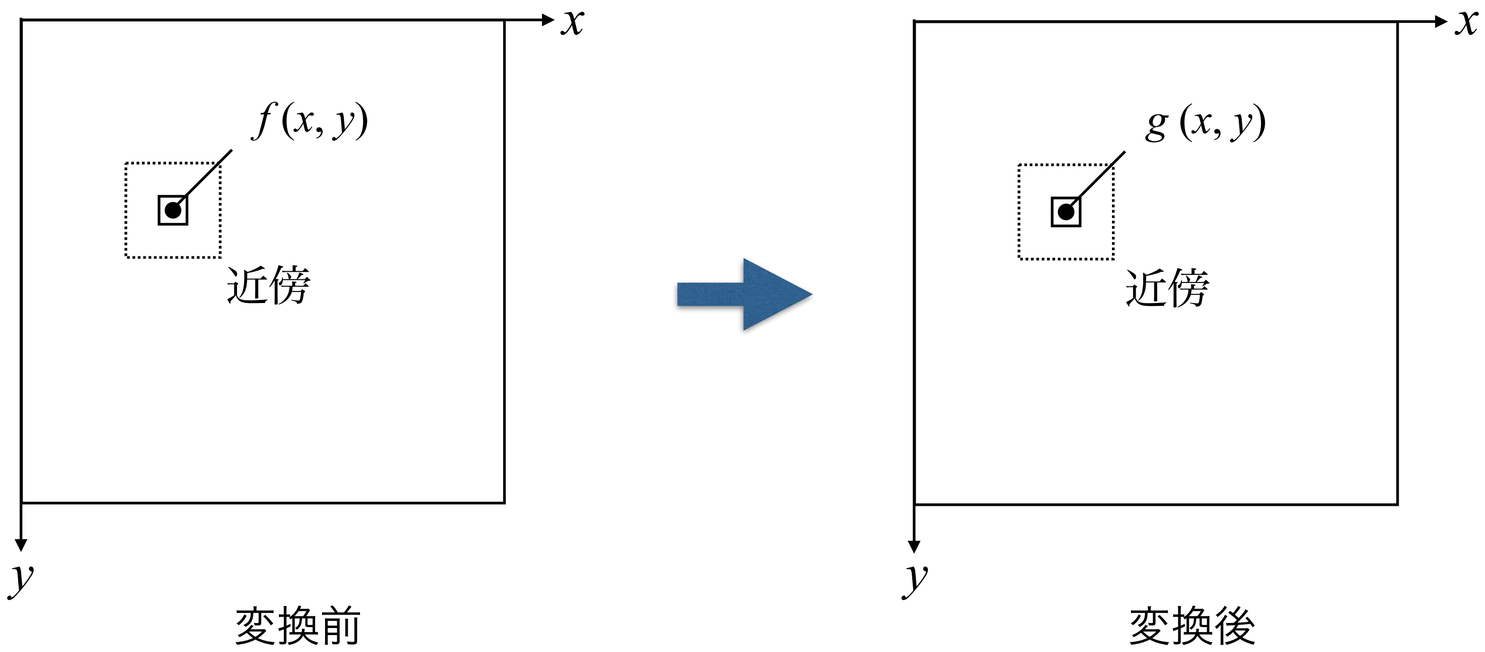
第14回：グラフィックパイプラインとシェーダ（7/18）

第15回：マルチメディアの応用例I（7/25）

髙木一幸 takagi@uec.ac.jp



空間フィルタリング

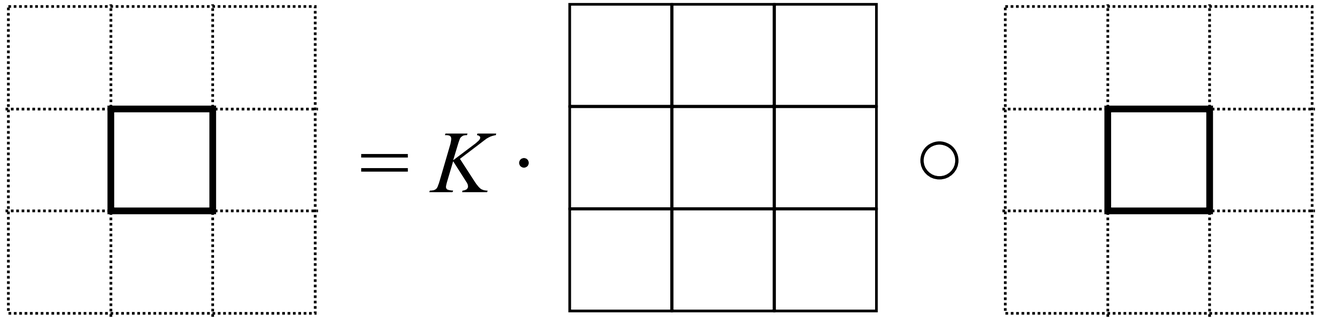


- エッジや図形の強調処理（微分）
- 平滑化やノイズ除去処理（積分）
- パターン抽出処理

空間フィルタリングでは、原画像 $f(x, y)$ の任意の注目画素の変換後の階調 $g(x, y)$ を、注目画素およびその近傍の画素の階調値を用いた演算によって求める。

髙木一幸 takagi@uec.ac.jp

エッジ強調（微分）処理



$g(x, y)$ $= K \cdot$ C_{ij} $\circ$ $f(x, y)$

○ : Hadamard product

$$g(x, y) = K \cdot \sum_{i=-1}^{1} \sum_{j=-1}^{1} C_{ij} \cdot f(x+i, y+j)$$

線形微分（差分フィルタ）

髙木一幸 takagi@uec.ac.jp

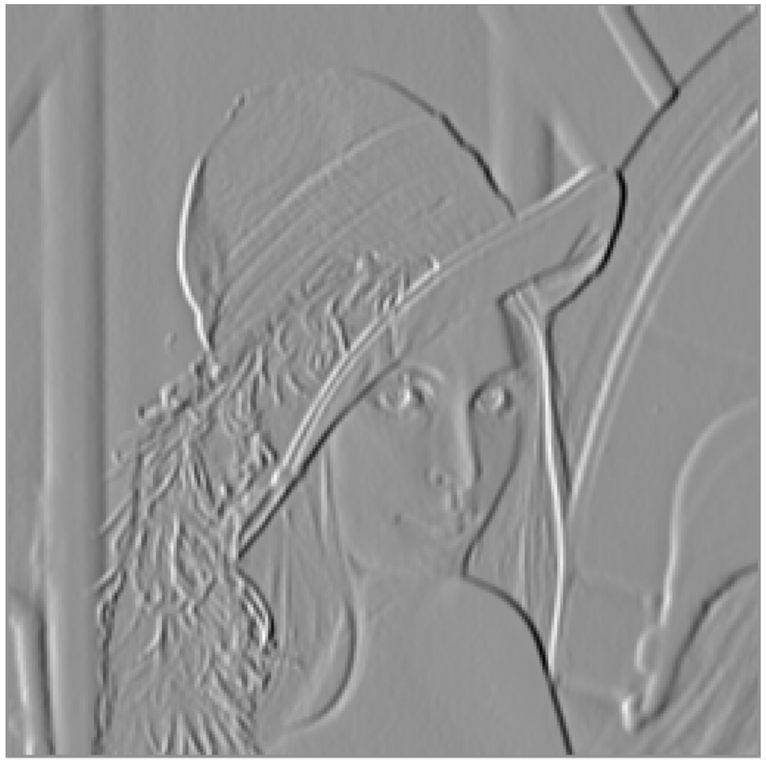
基礎的な線形微分フィルタ

$K=1$

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

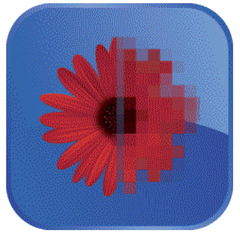
C_{ij}

1次微分フィルタ
3×3近傍



差分の方向が横なので、横方向に変化の大きい場所の値が大きくなる（グレースケール画像では明るくなる）

髙木一幸 takagi@uec.ac.jp



基礎的な線形微分フィルタ

アルゴリズム（擬似コード風）

−1	0	1
−1	0	1
−1	0	1

C_{ij}

```
c=[-1 0 1; -1 0 1; -1 0 1];
for i=2:255
  for j=2:255
    s=0;
    for k=1:3
      for l=1:3
        s=s+c(k,l)*a(i+k-2,j+l-2);
      end
    end
    b(i,j)=round(s);
  end
end
b(1,1) = b(2,2);
b(256,1) = b(255,2);
b(1,256) = b(2,255);
b(256,256) = b(255,255);
```

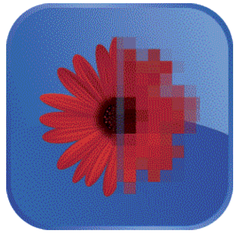
s：積和計算値

a(*,*)：処理対象画像

round()：小数点以下四捨五入

画像外周部は１ピクセル内側の
階調値をコピーする場合のコード
※後で説明

髙木一幸 takagi@uec.ac.jp



Processingで行列を扱う

−1	0	1
−1	0	1
−1	0	1

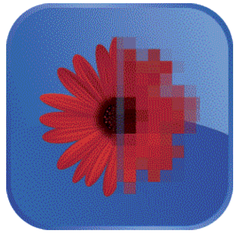
C_{ij}

```
float[][] c = { { -1, 0, -1 },
                { -1, 0, -1 },
                { -1, 0, -1 } };
```

- 初期値は行毎に値をカンマで区切って列挙して " { } " で囲い、
  行をカンマで区切って順番に並べたものを " { } " で囲う。

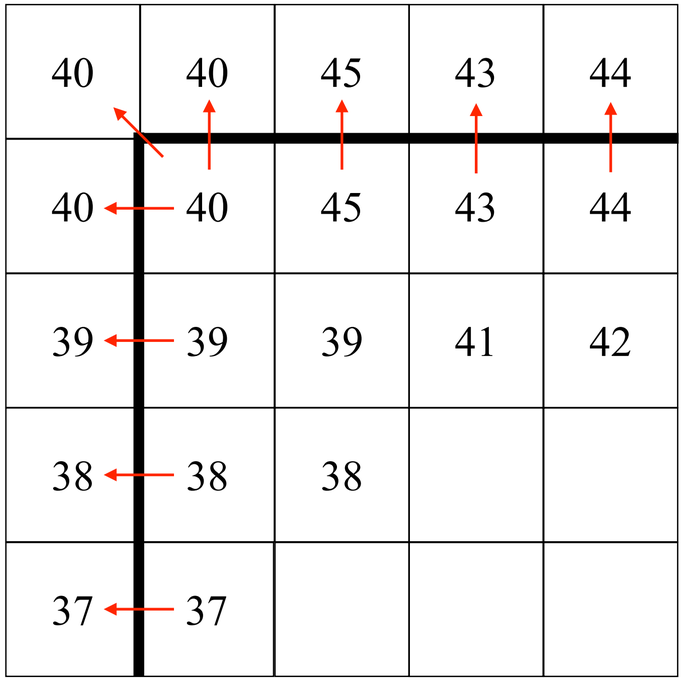
- 行毎に改行する必要はないが、この例のように記述すると、
  人間にとってわかりやすい。

髙木一幸 takagi@uec.ac.jp

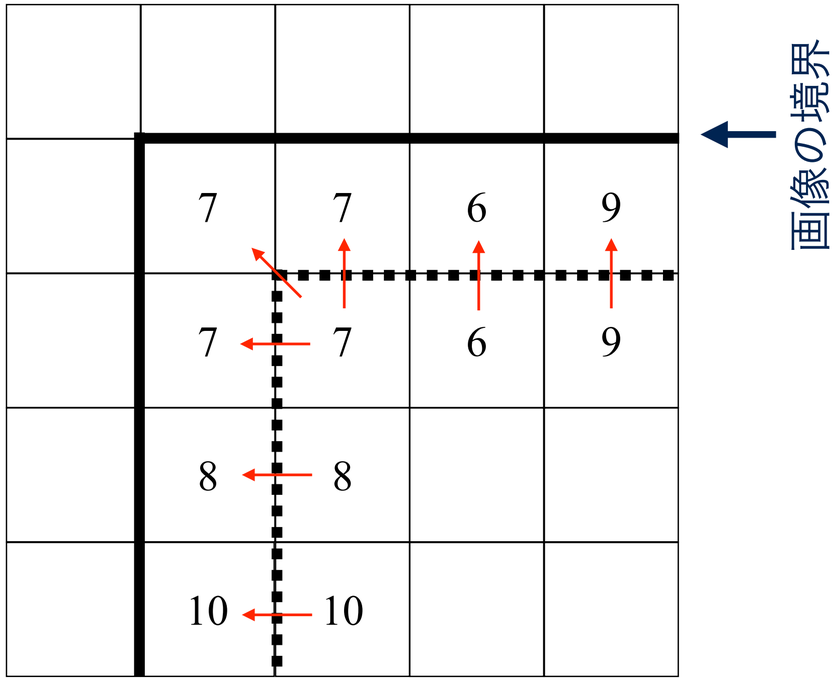


画像外周部分の処理

隣接画素がない場合は例外処理をする他ない
値をコピーする２種類の方法：計算前、計算後



画像の外側に階調値をコピー
計算前



1つ内側の計算結果を外周にコピー
計算後

髙木一幸 takagi@uec.ac.jp

基礎的な線形微分フィルタ

$K=1$

−1	−1	−1
0	0	0
1	1	1

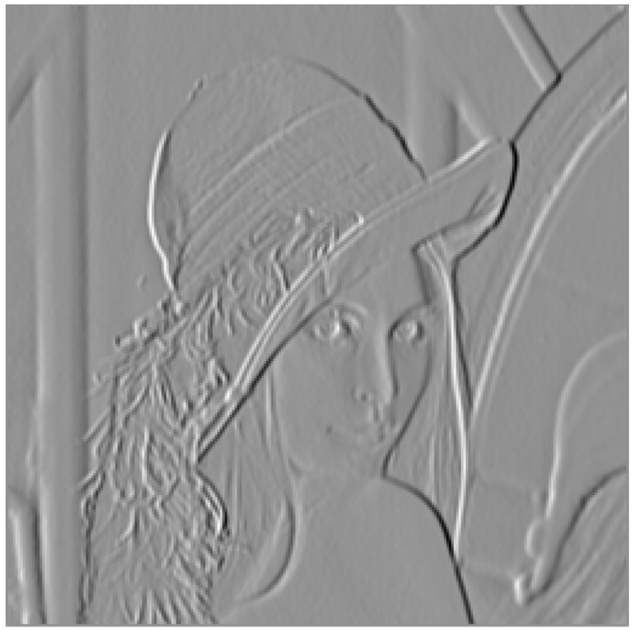
C_{ij}

1次微分フィルタ
3×3近傍

差分の方向が縦なので、縦方向に変化の大きい場所の値が大きくなる（グレースケール画像では明るくなる）

髙木一幸 takagi@uec.ac.jp

線形微分フィルタ：横方向と縦方向



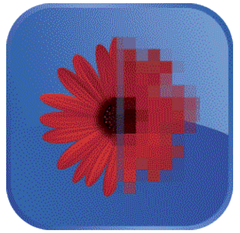
−1	0	1
−1	0	1
−1	0	1

横方向

−1	−1	−1
0	0	0
1	1	1

縦方向

髙木一幸 takagi@uec.ac.jp



基礎的な線形微分(差分)フィルタの例

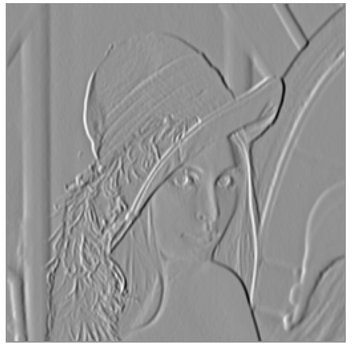
1次微分フィルタ
3×3近傍，K=1
※ 既出

−1	0	1
−1	0	1
−1	0	1

横

−1	−1	−1
0	0	0
1	1	1

縦



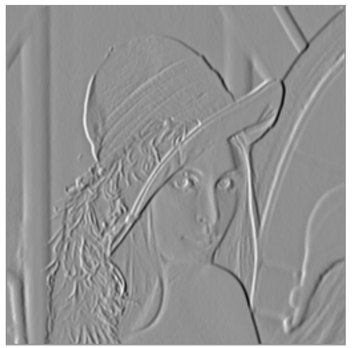
1次微分フィルタ
Sobel
3×3近傍，K=1

−1	0	1
−2	0	2
−1	0	1

横

−1	−2	−1
0	0	0
1	2	1

縦



2次微分フィルタ
Laplacian
3×3近傍，K=1

0	1	0
1	−4	1
0	1	0

4近傍

1	1	1
1	−8	1
1	1	1

8近傍

髙木一幸 takagi@uec.ac.jp

Sobelフィルタ

−1	0	1
−2	0	2
−1	0	1

横

−1	−2	−1
0	0	0
1	2	1

縦

髙木一幸 takagi@uec.ac.jp

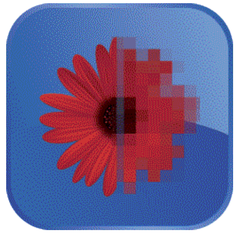
Laplacianフィルタ

2次微分フィルタ
Laplacian
3×3近傍，K=1

1	1	1
1	−8	1
1	1	1

8近傍

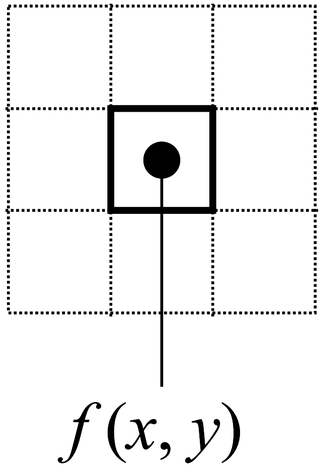
髙木一幸 takagi@uec.ac.jp



Laplacianフィルタの係数の導出

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

横方向(x)と縦方向(y)の２次微分



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

離散化した画像では微分は差分になる

$$= \{(f(x,y) - f(x-1,y))\} - \{(f(x-1,y) - f(x-2,y))\}$$

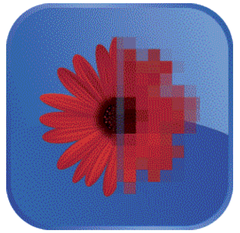
$$= f(x-2,y) - 2f(x-1,y) + f(x,y)$$

$x \leftarrow x - 1$ と置き換えると　基準位置をxに戻す

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

0	0	0
1	−2	1
0	0	0

髙木一幸 takagi@uec.ac.jp



Laplacianフィルタの係数の導出

同様にして

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

0	1	0
0	−2	0
0	1	0

以上２式より

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

$$= f(x, y-1) + f(x-1, y) - 4f(x, y)$$

$$+ f(x+1, y) + f(x, y+1)$$

0	0	0
1	−2	1
0	0	0

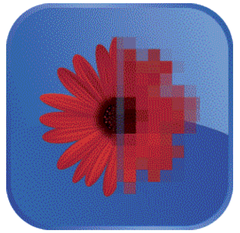
＋

0	1	0
0	−2	0
0	1	0

＝

0	1	0
1	−4	1
0	1	0

髙木一幸 takagi@uec.ac.jp



８近傍ラプラシアンフィルタ

1	1	1
1	−8	1
1	1	1

0	0	0
1	−2	1
0	0	0

＋

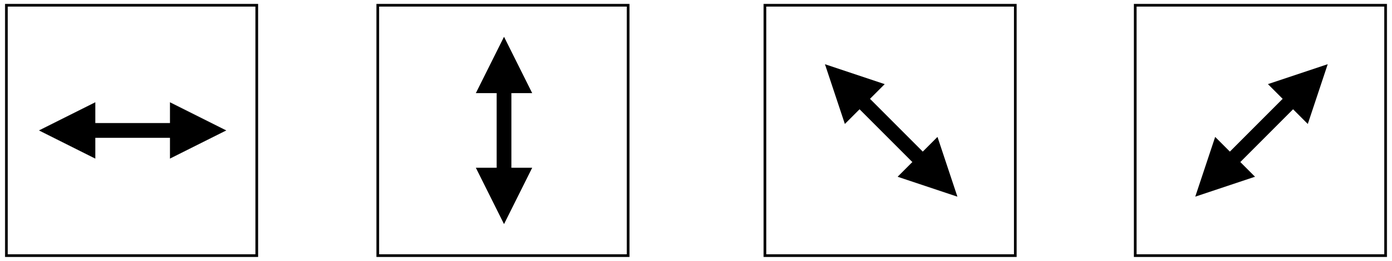
0	1	0
0	−2	0
0	1	0

＋

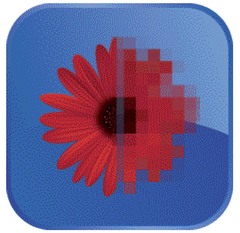
1	0	0
0	−2	0
0	0	1

＋

0	0	1
0	−2	0
1	0	0



髙木一幸 takagi@uec.ac.jp



非線形差分フィルタ

- Robertsフィルタ $$g(x,y)=\sqrt{\left(\sqrt{f(x,y)}-\sqrt{f(x+1,y+1)}\right)^2+\left(\sqrt{f(x,y+1)}-\sqrt{f(x+1,y)}\right)^2}$$
- Forsenフィルタ $$g(x,y)=|f(x,y)-f(x+1,y+1)|+|f(x,y+1)-f(x+1,y)|$$
- レンジフィルタ $$g(x,y)=\max\{f(x+m,y+n)|-1\le m,n\le 1\}-\min\{f(x+m,y+n)|-1\le m,n\le 1\}$$

Roberts

Forsen

range

髙木一幸 takagi@uec.ac.jp

平滑化（積分処理）

元画像

平滑化画像

$K = 1 / 9$

C_{ij}

1	1	1
1	1	1
1	1	1

線形平滑化フィルタ
3×3近傍

髙木一幸 takagi@uec.ac.jp

平滑化（積分処理）

ノイズ重畳画像

平滑化画像

$K = 1 / 9$　　C_{ij}

1	1	1
1	1	1
1	1	1

線形平滑化フィルタ
3×3近傍

髙木一幸 takagi@uec.ac.jp

平滑化（積分処理）

$K = 1 / 9$

C_{ij}

1	1	1
1	1	1
1	1	1

$K = 1 / 10$

C_{ij}

1	1	1
1	2	1
1	1	1

髙木一幸 takagi@uec.ac.jp

非線形平滑化フィルタ：最大値・最小値

原画像

最小値フィルタ

画像の暗い部分が膨張

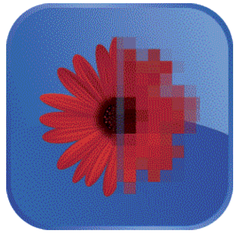
最大値フィルタ

画像の明るい部分が膨張

最小値フィルタ： $g(x, y) = \min\{f(x+m, y+n), -1 \le m, n \le 1\}$

最大値フィルタ： $g(x, y) = \max\{f(x+m, y+n), -1 \le m, n \le 1\}$

髙木一幸 takagi@uec.ac.jp



非線形平滑化フィルタ：medianフィルタ

medianフィルタ： $g(x, y) = \mathrm{median}\{f(x+m, y+n), -1 \le m, n \le 1\}$

ノイズ重畳画像

無作為に選んだ30%の画素に対し、[-50, 50]から無作為に選んだ値を画素値に加えた

線形平滑化フィルタ

3×3近傍

メディアンフィルタ

3×3近傍

髙木一幸 takagi@uec.ac.jp

パターン抽出処理（特定方向の強調）

抽出対象の形状に係数１、それ以外に−１を配置。

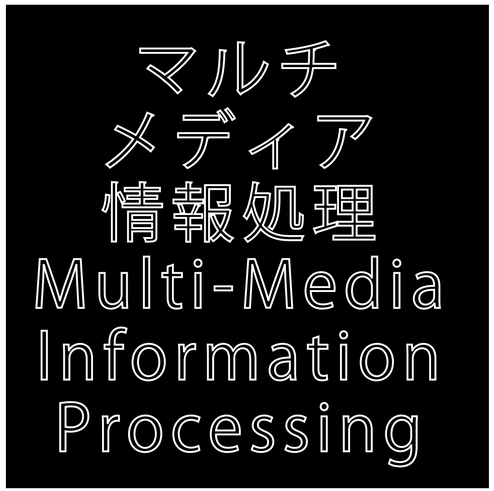
−1	−1	−1
1	1	1
−1	−1	−1

−1	1	−1
−1	1	−1
−1	1	−1

−1	−1	1
−1	1	−1
1	−1	−1

1	−1	−1
−1	1	−1
−1	−1	1

−1	−1	−1	−1	−1
−1	1	1	1	−1
−1	1	1	1	−1
−1	1	1	1	−1
−1	−1	−1	−1	−1

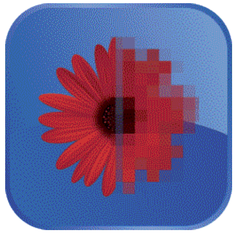


原画像

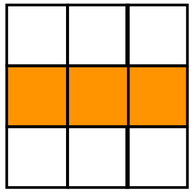
線幅1画素の水平強調画像

線幅1画素の垂直強調画像

髙木一幸 takagi@uec.ac.jp

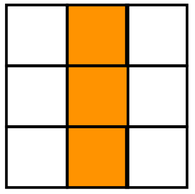


パターン抽出処理（特定方向の強調）



線幅1画素の水平強調画像

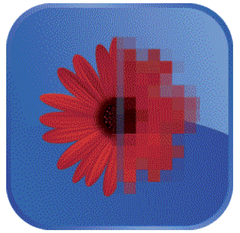
文字の横方向の縁の場所の値が大きい



線幅1画素の垂直強調画像

文字の縦方向の縁の場所の値が大きい

髙木一幸 takagi@uec.ac.jp

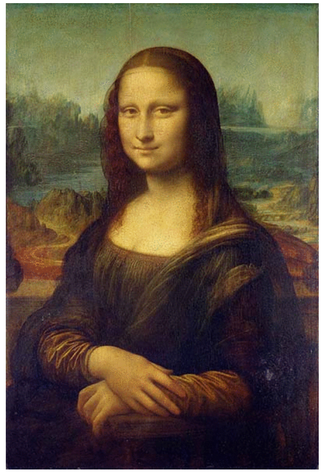


画像処理 (2) 空間フィルタリング

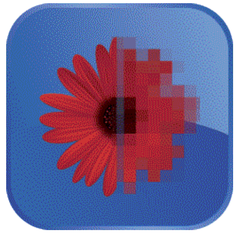
今日の要点

1. 注目画素および近傍の画素の階調値を用いて演算を行う
    1. エッジ・図形の強調処理：微分（差分）
        1. 線形微分フィルタ、Sobel、Laplacian
        2. 非線形差分フィルタ：Roberts、Forsen、レンジ
    2. 平滑化・ノイズ除去処理：積分
        1. 線形平滑化フィルタ
        2. 非線形平滑化フィルタ：最大・最小、median
    3. パターン抽出処理

        特定方向を強調（係数１と−１の配置による）
2. 画像外周部分の例外処理



髙木一幸 takagi@uec.ac.jp



第2回レポート課題

画像処理プログラム

画像処理プログラムの画像処理の
数値計算部は自分で書いてください。
Processingに対応する関数やライブラリが
備わっているものもありますが、
レポートの眼目は処理の内容を理解し、
自分でコーディングすることです。

画素ごとの濃淡変換

(1) γ補正

・スライダ操作によりインタラクティブに変化させること

・カラー画像を補正する場合は、R、G、Bを独立して変化することができること

(2) 曲線による階調変換（講義で擬似コードを示したもの以外、微分可能関数でなくても良い）

空間フィルタリング

(3) メディアンフィルタ：ノイズ入り画像を作成しフィルタを適用すること

(4) Laplacianフィルタ或はパターン抽出フィルタ

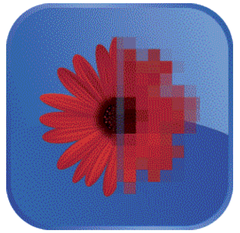
レポート課題に用いる画像データ

・Lenna以外のSIDBAの画像（カラー、白黒は問わない）、あるいは

・自分で撮影した画像など

髙木一幸 takagi@uec.ac.jp

次頁に続く



第2回レポート課題

画像処理プログラム

前頁から続く

画像処理プログラムの画像処理の
数値計算部は自分で書いてください。
Processingに対応する関数やライブラリが
備わっているものもありますが、
レポートの眼目は処理の内容を理解し、
自分でコーディングすることです。

- 提出期限：**2025年6月12日（木）**23:59:59
- 提出先：　mmip@takagi.inf.uec.ac.jp
  - メールの件名： MMIP Report #2
  - メール本文：空
  - ファイル名は "UECアカウント名.pdf"（e.g. “a2310765.pdf"）
  - メールの添付ファイル
    (1)レポート本体：PDF形式　**ファイル名は「メールの件名」と同一**
    (2)Processingのスケッチ（および必要ならば画像データファイル）

フォルダ構造を維持したまま圧縮アーカイブファイル（.zip）にすること

- **考察（≠感想）が無いものは評価の対象にしない。**

Processingのノイズ

random(), randomSeed(), noise(), noiseSeed(), noiseDetail()

random()

乱数を生成（ホワイトノイズ）。呼び出すたびに異なる値。

```
random(high), random(low, high)  low, high: float
```

randomSeed()

乱数の種を設定。乱数列を決定する。同じ値を設定すれば、同じ乱数列が得られる。`randomSeed(value)  value: int`

noise()

パーリンノイズの生成（CGで自然物の表現に使われる）[0, 1.0]

```
noise(x), noise(x, y), noise(x, y, z)
                                  x, y, z: float
```

noiseSeed()

パーリンノイズの種を設定。同じ値を設定すれば、同じ乱数列が得られる。

```
noiseSeed(vale)  value: int
```

noiseDetail()

パーリンノイズの性質はオクターブ数と減衰係数という２つのパラメータを使って調整。

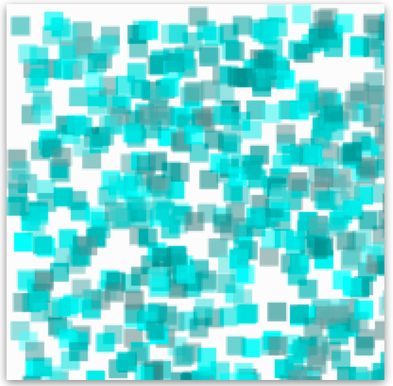
```
noiseDetail(octaves), noiseDetail(octaves, falloff)
octaves: int, falloff: float
```

髙木一幸 takagi@uec.ac.jp

Processingのノイズ

random()を用いて生成したパターン

```
size(200, 200);
colorMode(HSB, 100);
background(99);
for(int x=0; x<width; x++) {
  float color1 = random(100);
  stroke(color1, 60, 100);
  line(x, 0, x, height);
}
```



```
size(200, 200);
colorMode(RGB, 100);
background(99);
noStroke();
for(int i=0; i<600; i++) {
  float color1 = random(0, 50);
  float color2 = random(50, 100);
  fill(color1, color2, color2, 60);
  float x = random(200);
  float y = random(200);
  rect(x, y, 10, 10);
}
```

© 田中孝太郎、前川峻志、

Built with Processing

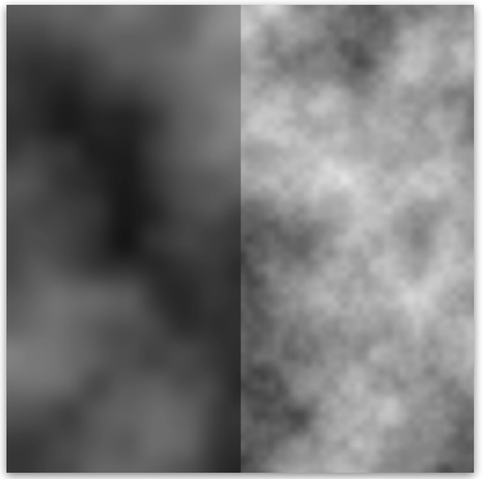
—デザイン/アートのためのプログラミング入門、

2010年、BNN による

髙木一幸 takagi@uec.ac.jp

Processingのノイズ

Perlinノイズを用いて質感が異なる２種類のテクスチャを生成



```
float val;
float s = 0.02;

void setup(){
  size(200, 200);
}

void draw() {
  for (int y=0; y<height; y++) {
    for (int x=0; x<width/2; x++) {
      noiseDetail(3, 0.4);
      val = noise((mouseX+x)*s, (mouseY+y)*s);
      stroke(val*255);
      point(x, y);
      noiseDetail(5, 0.6);
      val = noise((mouseX+x+width/2)*s, (mouseY+y)*s);
      stroke(val*255);
      point(x+width/2,y);
    }
  }
}
```

変化の緩やかなノイズ

細かいノイズ

髙木一幸 takagi@uec.ac.jp

ジェネラティブ・アート
Generative Art

コンピュータソフトウェアのアルゴリズムや

数学的/機械的/無作為的自律過程によって

アルゴリズム的に生成・合成・構築される芸術作品

Wikipedia

髙木一幸 takagi@uec.ac.jp

ジェネラティブ・アート
Generative Art

ケーススタディ：Wave Clock

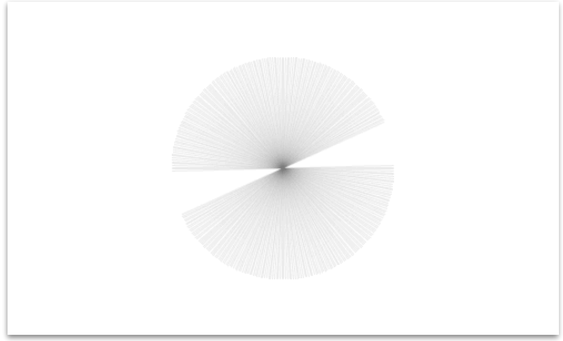
© マット・ピアソン著、久保田晃弘監修, 沖啓介翻訳、ジェネラティブ・アート ―Processingによる実践ガイド 、2014年、BNN　の作例を一部変更した

step.1　直線を回転させて円を描く

```
float _angle = -PI/2;
float _radius = 100;

void setup() {
  size(500, 300);
  smooth();
  frameRate(30);
  background(255);
  stroke(128, 32);
  strokeWeight(1);
  noFill();
}
```

```
void draw() {
  float centerX = width/2;
  float centerY = height/2;
  float rad = radians(_angle);
  float x1 = centerX + (_radius * cos(rad));
  float y1 = centerY + (_radius * sin(rad));
  float opprad = rad + PI;
  float x2 = centerX + (_radius * cos(opprad));
  float y2 = centerY + (_radius * sin(opprad));
  line(x1, y1, x2, y2);
  _angle += 1;
}
```



髙木一幸 takagi@uec.ac.jp

ジェネラティブ・アート
Generative Art

ケーススタディ：Wave Clock

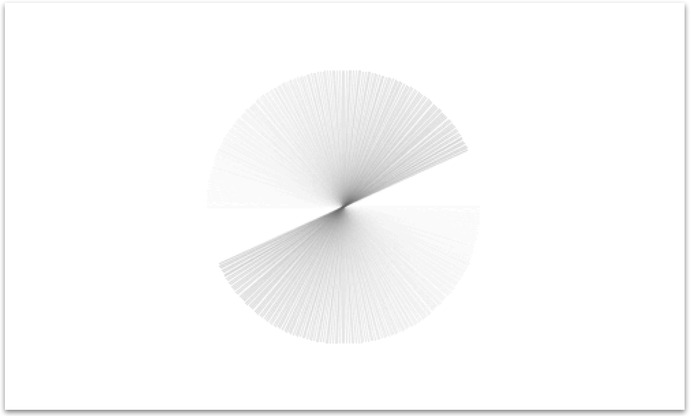
step.2

線の色を255（白）から始めてフレーム毎に１つづ0（黒）に達するまで減少させ、
また逆に255に戻るようにして、フェード効果を与える。

```
float _angle = -PI/2;
float _radius = 100;
float _strokeCol = 254;
int _strokeChange = -1;

void setup() {
  size(500, 300);
  smooth();
  frameRate(30);
  background(255);
  noFill();
}
```

```
void draw() {
   (省略)
  _strokeCol += _strokeChange;
  if(_strokeCol > 254) { _strokeChange = -1; }
  if(_strokeCol < 0) { _strokeChange = 1; }
  stroke(_strokeCol, 60);
  line(x1, y1, x2, y2);
  _angle += 1;
}
```



髙木一幸 takagi@uec.ac.jp

ジェネラティブ・アート
Generative Art

ケーススタディ：Wave Clock

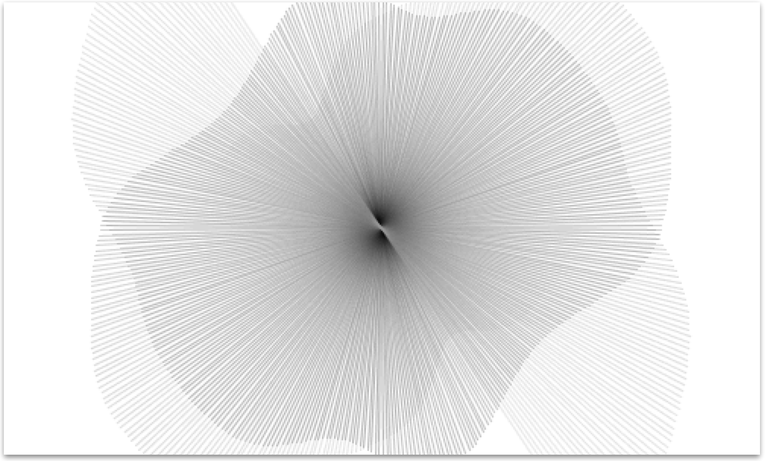
step.3　回転につれて線の長さをノイズ値によって変化するようにする。

```
（省略）
float _radiusnoise;

void setup() {
  size(500, 300);
  smooth();
  frameRate(30);
  background(255);
  noFill();
  _radiusnoise = random(10);
}
```

```
void draw() {
  _radiusnoise += 0.005;
  _radius = (noise(_radiusnoise) * 550) +1;

  （省略）
  line(x1, y1, x2, y2);
  _angle += 1;
}
```



髙木一幸 takagi@uec.ac.jp

ジェネラティブ・アート
Generative Art

ケーススタディ：Wave Clock

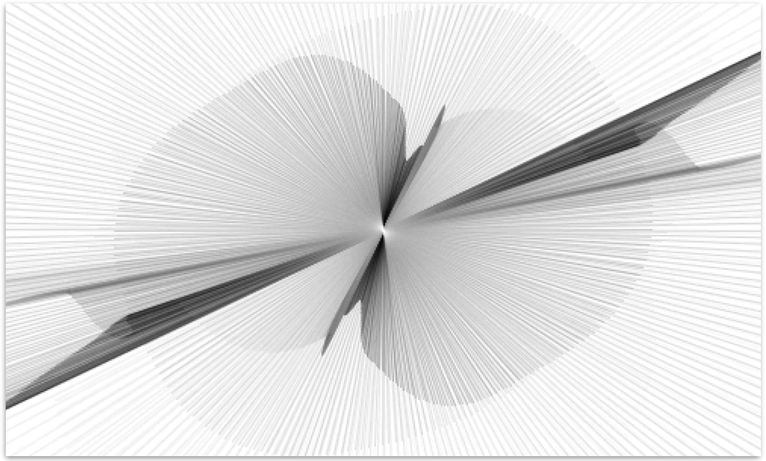
step.4　回転の角度にノイズ変動を加える。

```
(省略)
float _angnoise;

void setup() {
  size(500, 300);
  smooth();
  frameRate(30);
  background(255);
  noFill();
  _radiusnoise = random(10);
  _angnoise = random(10);
}
```

```
void draw() {
  (省略)
  _angnoise += 0.005;
  _angle += (noise(_angnoise) * 6) - 3;
  if (_angle > 360) { _angle -= 360; }
  if (_angle < 0) { _angle += 360; }

  (省略)
}
```



髙木一幸 takagi@uec.ac.jp

ジェネラティブ・アート
Generative Art

ケーススタディ：Wave Clock

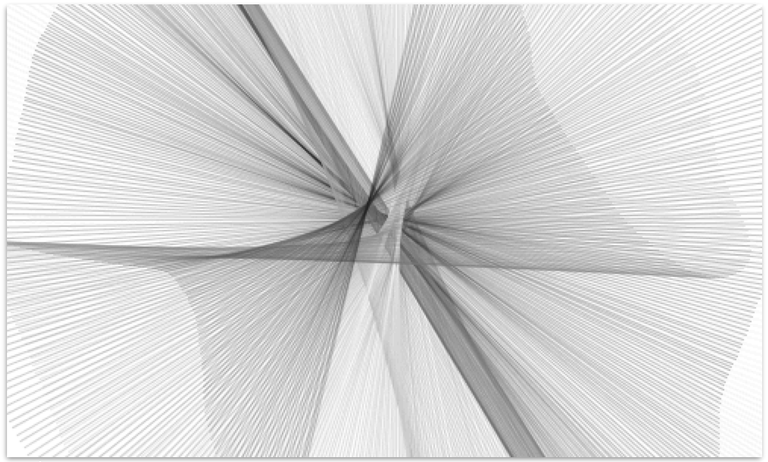
© マット・ピアソン著、久保田晃弘監修, 沖啓介翻訳、ジェネラティブ・アート ―Processingによる実践ガイド 、2014年、BNN　の作例を一部変更した

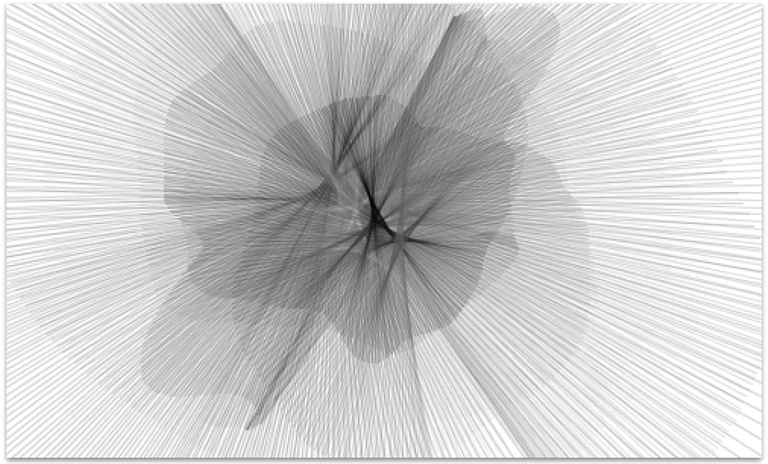
step.5　円の中心を揺らせる：ノイズ関数を使ってx軸、y軸方向に最大±50ピクセル

```
(省略)
float _xnoise, _ynoise;

void setup() {
  (省略)
  _xnoise = random(10);
  _ynoise = random(10);
}
```

```
void draw() {
  _xnoise += 0.01;
  _ynoise += 0.01;
  float centreX = width/2+(noise(_xnoise)*100)-50;
  float centreY = height/2+(noise(_ynoise)*100)-50;
  (省略)
}
```





髙木一幸 takagi@uec.ac.jp