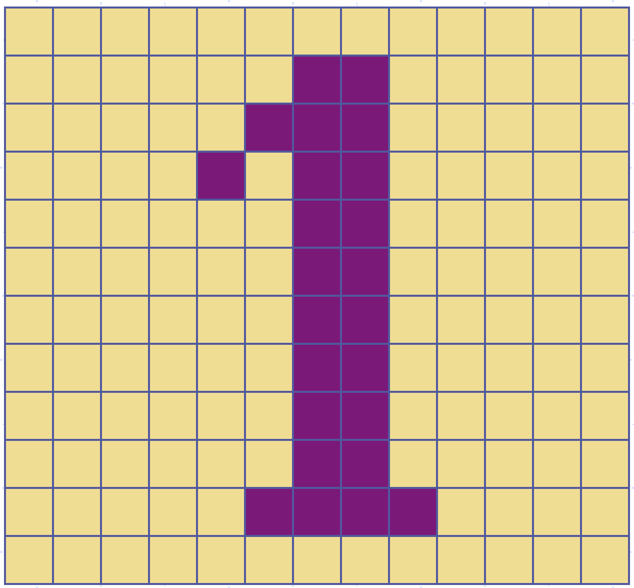
二次元画像処理

日浦慎作

パターンとシンボル

パターン



- 均質な要素の配列
- 各要素値の並びが重要

シンボル

0	青
1	赤
2	緑
3	黄

- 不均質均質な要素の配列
- 各要素が独立に意味を持つ

画像の処理と認識・理解

狭義の画像処理

- 画像処理・画像変換（パターン→パターン）
  - 画質改善
  - 画像符号化・圧縮
  - メディア変換（不可視情報の可視化）
- 画像認識・画像理解（パターン→シンボル）
  - ２次元パターン認識
  - ３次元画像計測・認識
- 画像生成（シンボル→パターン）
  - コンピュータグラフィックス

二次元画像処理

- フィルタ演算
  - 画像をぼかす・鮮明にする
  - 画像中のエッジ(輪郭)を抽出する　など
- 明度変換
  - コントラストの強調，二値化　など
- 二値画像処理
  - ノイズ除去・ラベリング・細線化　など
- 画像圧縮
  - 可逆圧縮・非可逆圧縮

画像処理が対象とするデータ

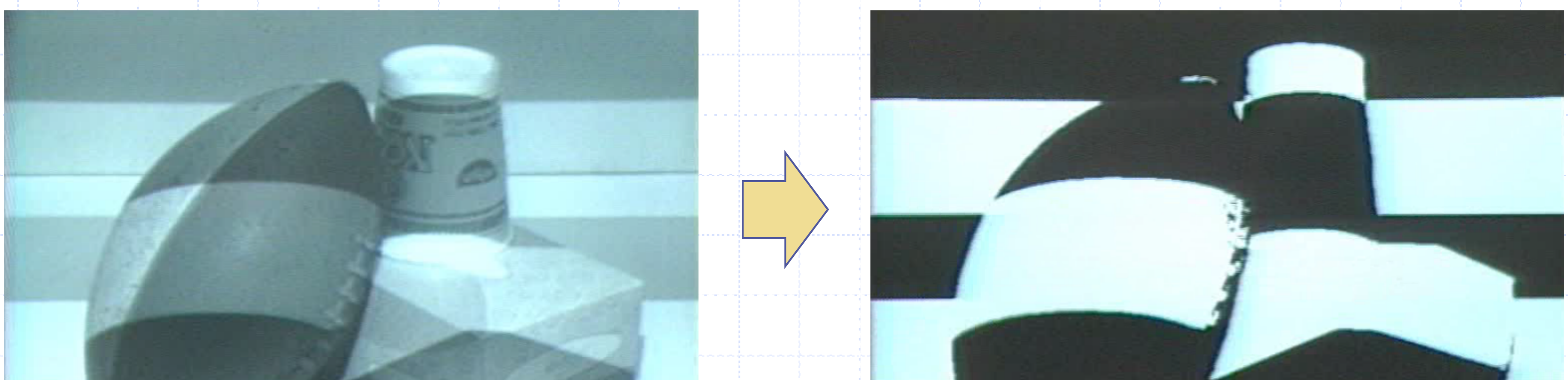
カラー画像
24bit/pixel

濃淡画像
8bit/pixel

二値画像
1bit/pixel

二値画像処理（教科書5章）

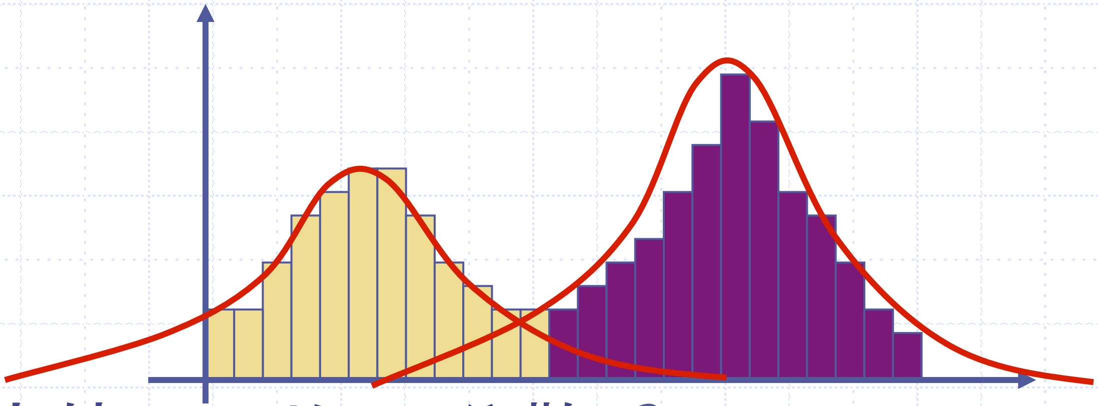
- 画像を「白」と「黒」だけで扱う処理
  - 図形の処理として，もっとも基本的
- 二値化とは
  - 画像を白と黒の領域に分ける処理
  - どのぐらいの明るさにするか？が問題



二値化と閾値決定

- 図と地の割合が予測できる場合（文書等）
  - P-タイル法
    ヒストグラムを一方から加算した割合が
    ちょうどp になる値を利用
- ヒストグラムがはっきりとした双峰性
  - ２つのピーク間の最小値をしきい値とする
- その他の場合
  - 判別分析法（次スライド）

判別分析法



- 特徴量
  - 全画素の明度値の平均μ, 分散σ^2
  - 閾値以上・以下の分布をクラス1,2に分類
  - 各クラス x の割合w_x , 平均μ_x, 分散σ_x^2

クラス内分散 $\sigma_w^2 = w_1\sigma_1^2 + w_2\sigma_2^2$

クラス間分散 $\sigma_B^2 = w_1(\mu_1 - \mu)^2 + w_2(\mu_2 - \mu)^2 = w_1 w_2(\mu_1 - \mu_2)^2$

$$\sigma^2 = \sigma_w^2 + \sigma_B^2$$

- 閾値の決定
  - クラス間の分離度σ_B^2 / σ^2を最大にする閾値

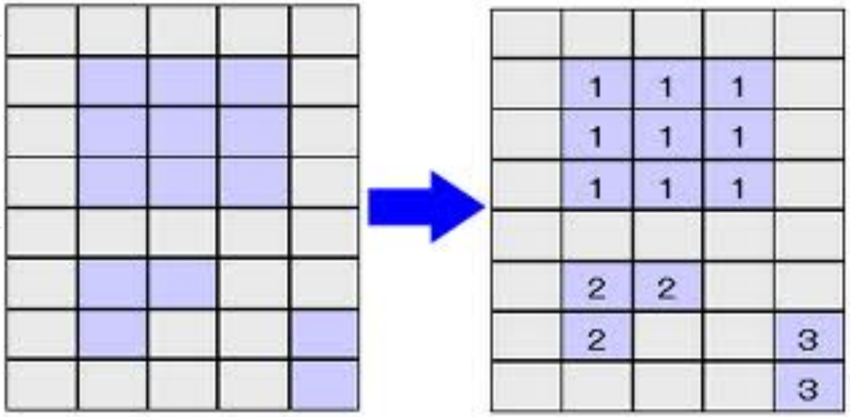
参考　http://imagingsolution.blog107.fc2.com/blog-entry-113.html

ラベリング

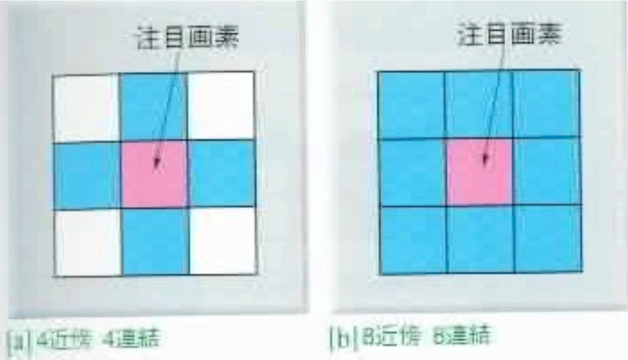
つながった領域に「同じ値」を割り当てる処理

- 連続領域の数を調べる
- それぞれの連続領域の面積や重心を求める

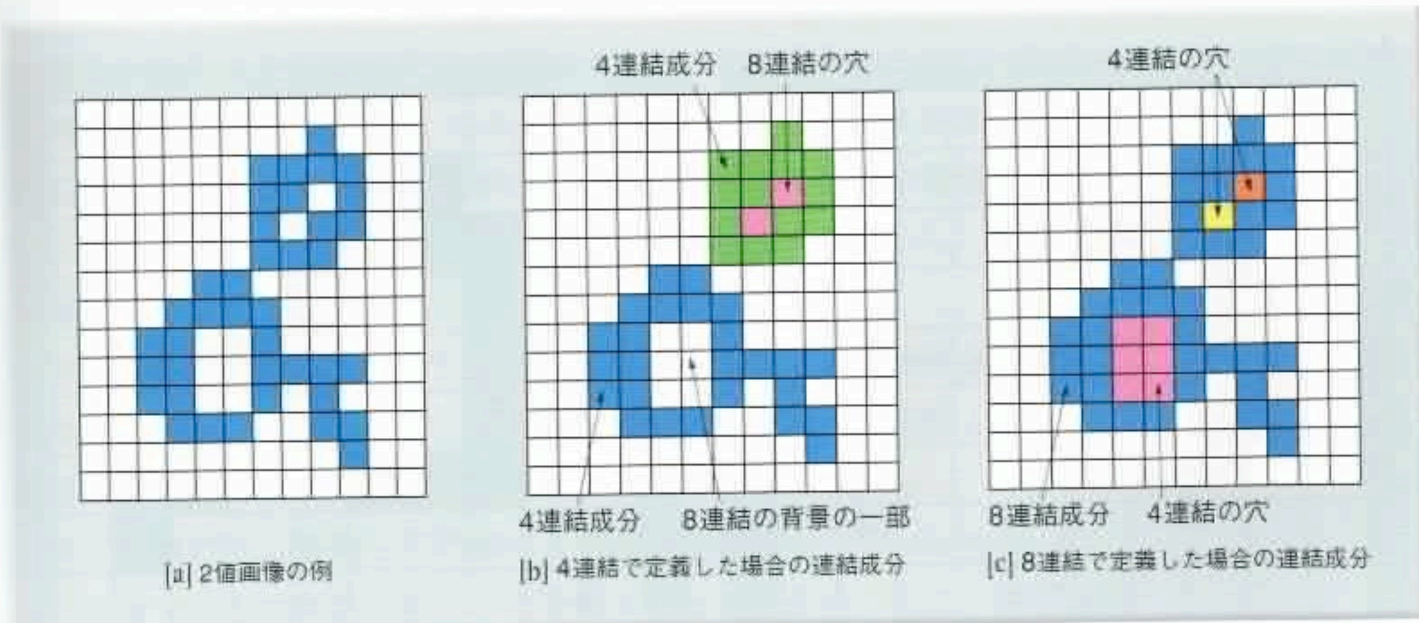
などの処理が可能になる



4近傍と8近傍



■図10.5——近傍・連結の種類



■図10.6——2値画像の連結成分

ラベリング結果の違い

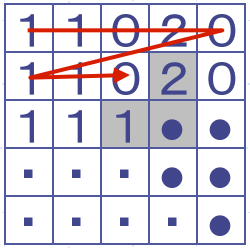
【4連結】

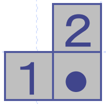
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	2	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0	2	2	0	0	0
0	1	1	1	1	1	1	0	0	0	2	2	2	0	0	0
0	0	1	1	1	1	1	0	0	0	2	2	2	0	0	0
0	0	0	1	1	1	1	0	0	2	2	2	0	0	0	0
0	0	0	1	0	0	0	2	2	2	2	0	0	0	0	0
0	0	0	0	0	3	0	0	2	2	0	0	0	0	0	0
0	0	0	0	3	3	3	0	0	0	0	4	0	0	0	0
0	0	0	3	3	3	3	3	0	4	4	4	4	0	0	0
0	0	0	0	3	3	3	0	0	4	4	4	4	0	0	0
0	0	0	0	0	3	0	0	0	4	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

【8連結】

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0	1	1	0	0	0
0	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0
0	0	1	1	1	1	1	0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	1	1	1	0	0	0	0
0	0	0	1	0	0	0	1	1	1	1	0	0	0	0	0
0	0	0	0	0	2	0	0	1	1	0	0	0	0	0	0
0	0	0	0	2	2	2	0	0	0	0	3	0	0	0	0
0	0	0	2	2	2	2	2	0	3	3	3	3	0	0	0
0	0	0	0	2	2	2	0	0	3	3	3	3	0	0	0
0	0	0	0	0	2	0	0	0	3	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ラベリング





重複リスト

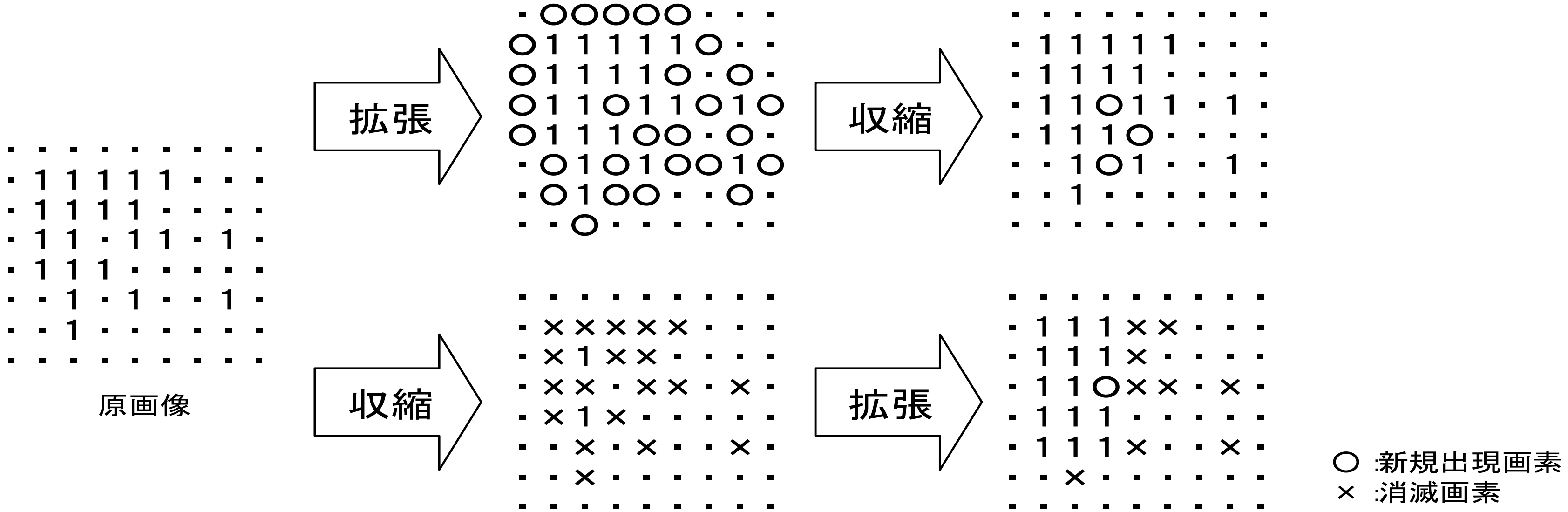
1	2
・	・
・	・
・	・

[1パス]
上または左の画素と同じラベルを付与
- 左と上の画素が異なるラベルを持つ場合
  - 重複リストに追加
- 上も左も，0画素である場合
  - 新しいラベル番号を付与

[2パス]
重複リストを元に，ラベルを更新

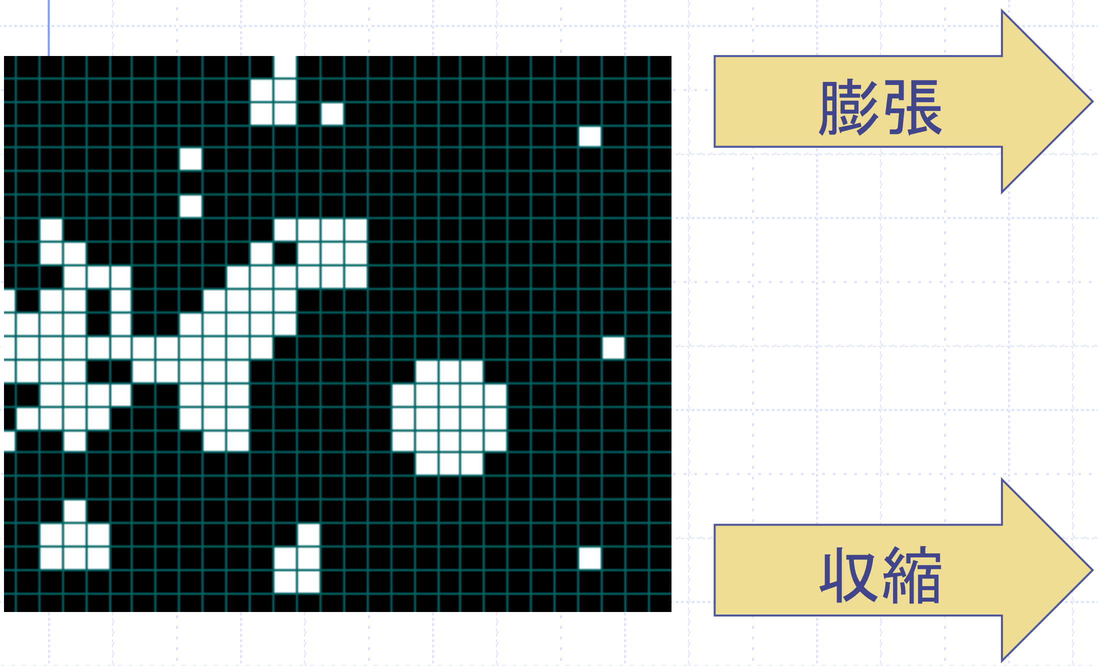
参考 http://imagingsolution.blog107.fc2.com/blog-entry-193.html

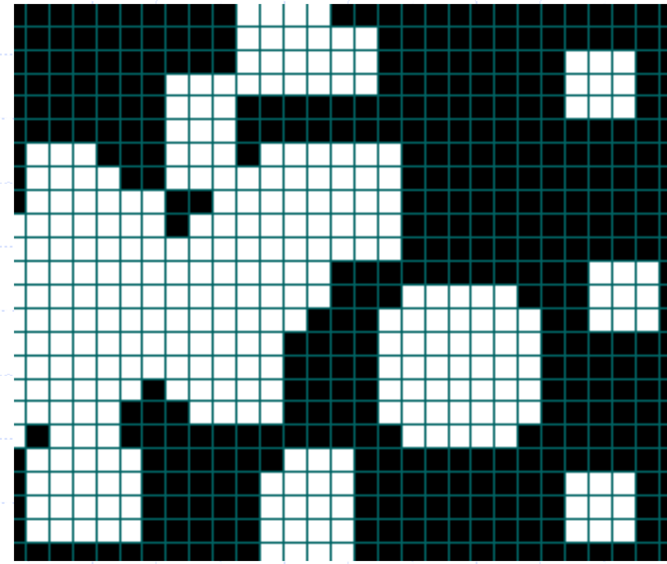
膨張・収縮

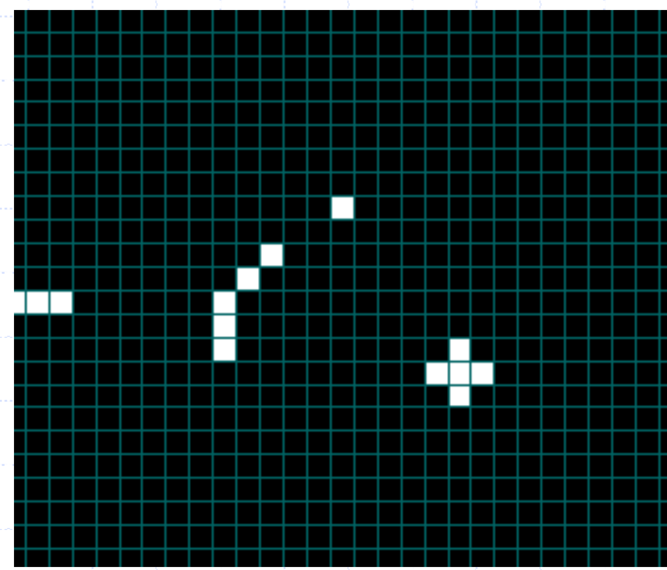


◆拡張・収縮 – 穴を埋める効果

◆収縮・拡張 – 孤立点を除去する効果



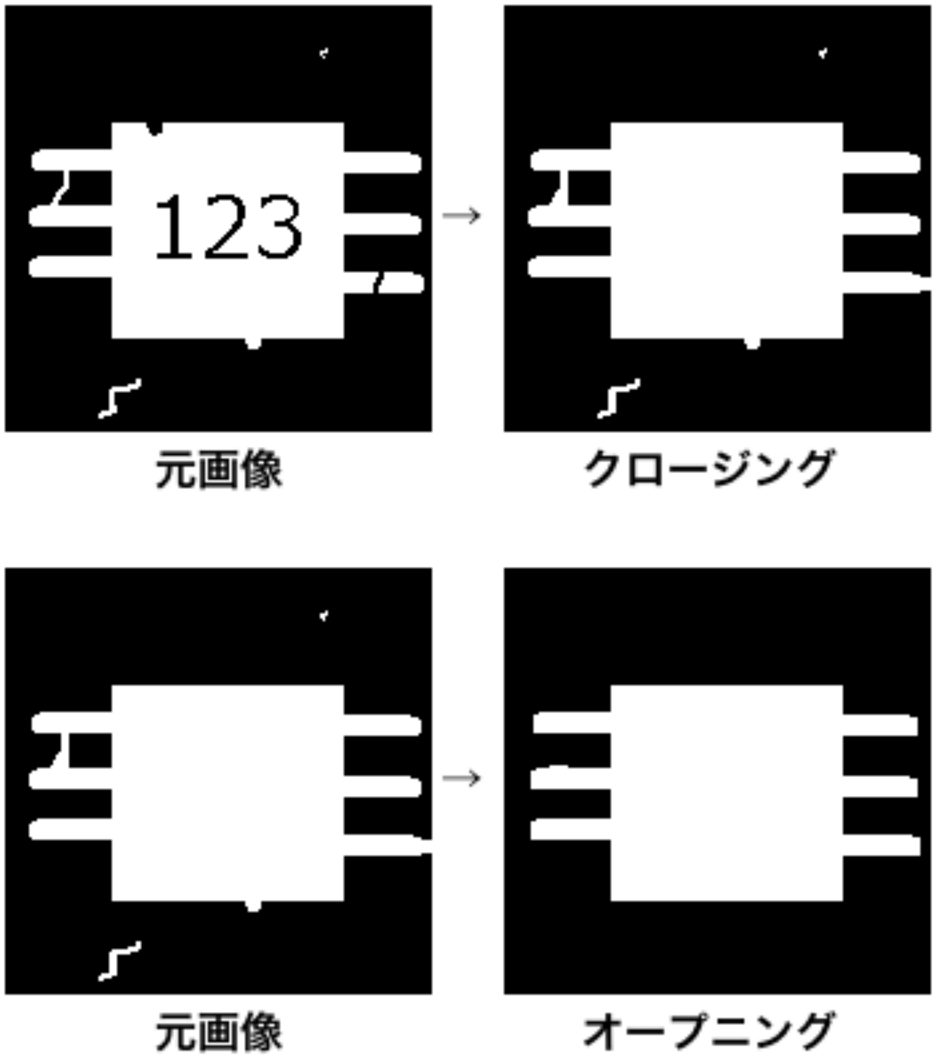




http://imagingsolution.blog107.fc2.com/blog-entry-101.html

膨張・収縮処理は単独で処理を行う事はまれで、膨張・収縮を繰り返し処理を行う場合が多くあります。

とくに、同じ回数分だけ膨張して収縮する処理を**クロージング（Closing）**、
同じ回数分だけ収縮して膨張する処理を**オープニング（Opening）**とよびます。



穴や溝を埋める効果

孤立点・細線を消す効果

距離変換・スケルトン

```
・・・・・・・・・
・11111・・・
・111111・・
・1111111・
・11111・・・
・・1・1・・・・
・・1・1・・・・
・・・・・・・・・
```

原画像

距離変換

```
・・・・・・・・・
・11111・・・
・122221・・
・1223211・
・12121・・・
・・1・1・・・・
・・1・1・・・・
・・・・・・・・・
```

距離変換

スケルトン化

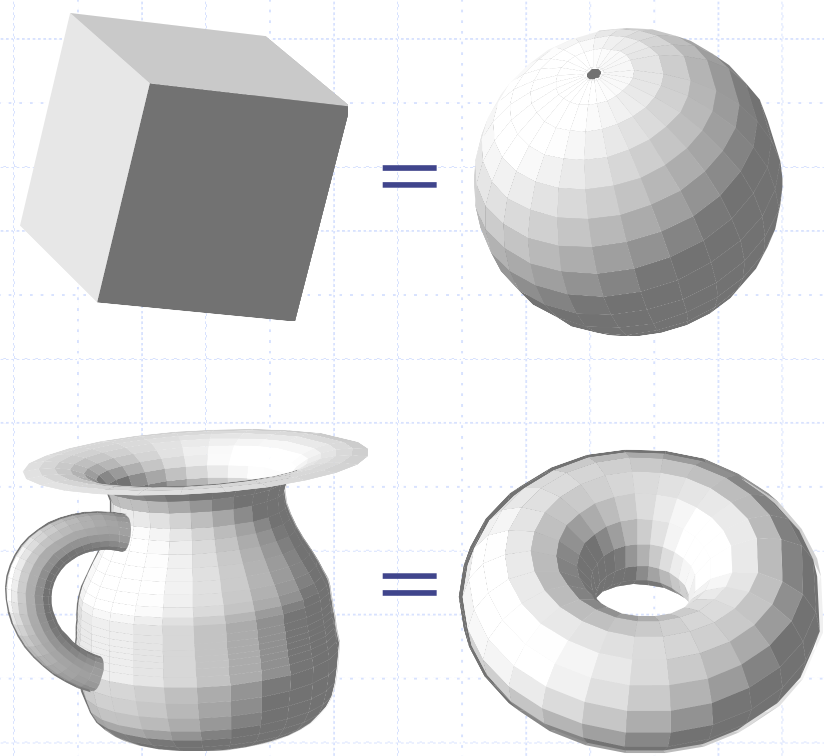
```
・・・・・・・・・
・・・・・・・・・
・・22・2・・・
・・2・3・・1・
・・2・・・・・・
・・・・・・・・・
・・1・1・・・・
・・・・・・・・・
```

スケルトン

- 距離変換 – 何度目の収縮処理で0画素になるか
- スケルトン – 距離変換画像の極大点（近傍画素値が中央画素の値以下）
  - 元の画像を復元可能

近傍演算による大局的情報:トポロジーの利用

- トポロジーとは
  - 変形しても変化しない図形の性質



二次元二値画像のトポロジー

- 連結成分の穴の数に対応

000000000000	000000000000
001110000000	001111100000
001111110000	001100110000
000011110000	000110010000
000110011000	000111111000
000000000000	000000000000
穴なし	穴1つ

オイラー数

- (連結成分の個数) − (穴の個数)

000000000000	000000000000	000000000000
001110000000	001111100000	001111100000
001111110000	001100110000	001101110000
000011110000	000110010000	000111010000
000110011000	000111111000	000111111000
000000000000	000000000000	000000000000
オイラー数：1	オイラー数：0	オイラー数：−1

オイラー数の計算方法

- 4近傍の場合

000000000000	000000000000	000000000000
001111100000	001111100000	001111100000
001100110000	001100110000	001100110000
000110010000	000110010000	000110010000
000111111000	000111111000	000111111000
000000000000	000000000000	000000000000
1の個数	縦・横2連接	2x2連接

("1"の数:18) - (縦・横2連接の数:8+12) + (2x2連接の数:2) = 0

オイラー数の原理(1)

◆位相が不変な操作

	000000 011100 001000 000000	000000 011100 001110 000000	000000 011100 001100 000000	000000 011100 001100 001100	000000 011110 001110 001100
画素数	＋1	＋1	＋1	＋1	＋1
縦・横連接	＋1	＋1	＋2	＋3	＋4
2x2連接	±0	±0	＋1	＋2	＋3
オイラー数	±0	±0	±0	±0	±0

000000
011110
001010
001100

注意：左の0は穴ではない
図(”1”)が4近傍の場合，地(“0”)は8近傍で考える

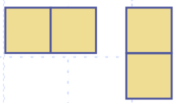
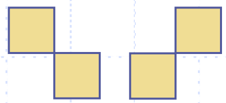
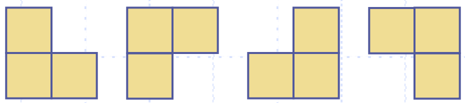
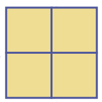
オイラー数の原理(2)

- 位相が変化する操作

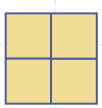
	000000 000000 001000 000000	000000 111010 000010 000010	000000 000110 001110 001100	000000 011110 001110 001110	000000 011110 001110 000100
画素数	＋1	＋1	＋1	＋1	＋1
縦・横連接	＋0	＋2	＋4	＋4	＋4
2x2連接	±0	±0	＋2	＋4	＋2
オイラー数	＋1	－1	－1	＋1	－1
操作	出現	接続	接続	穴埋め	接続

８近傍のオイラー数

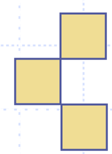
◆ちょっと複雑

				
画素数 V	縦・横 2連接 E	斜め接続 D	2x2領域中，3画素 T	2x2連接 F

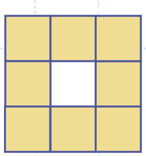
オイラー数＝V－E－D＋T－F



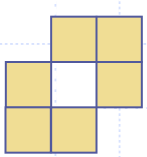
４－４－２＋４－１＝１



３－０－２＋０－０＝１



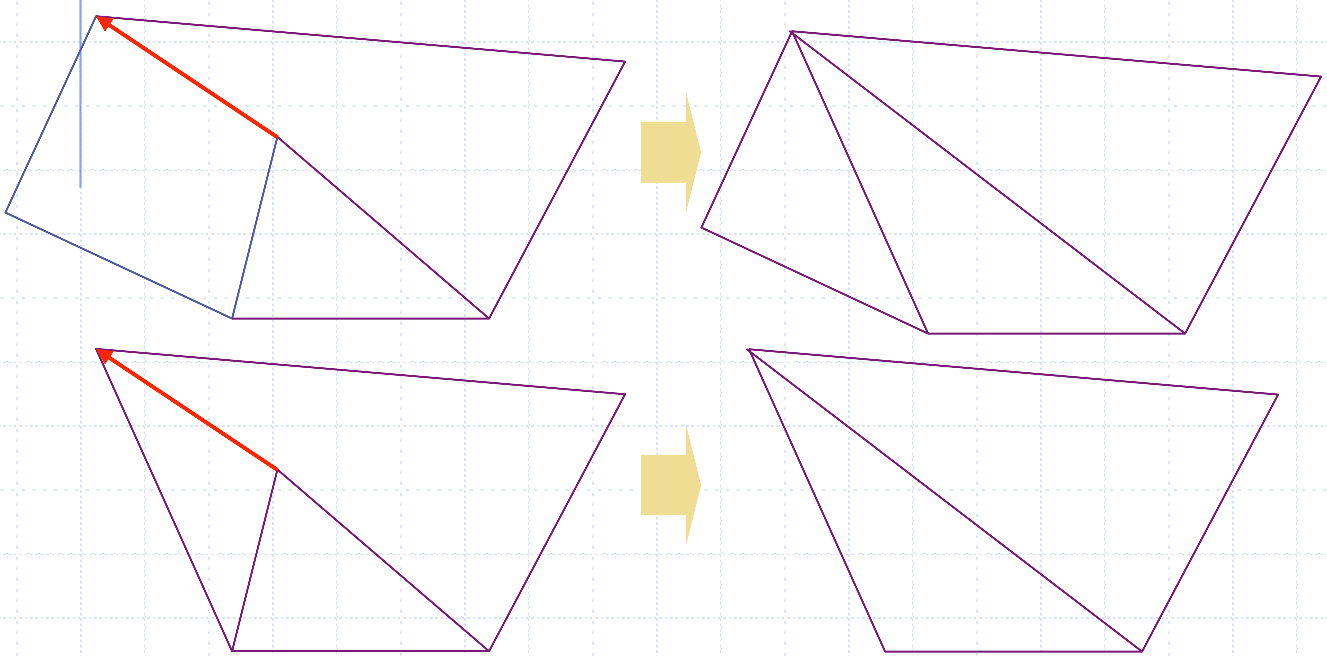
８－８－４＋４－０＝０



６－４－４＋２－０＝０

余談：オイラー数について

- 例：多面体の性質
  - （頂点数）－（辺数）＋(面数）＝２－２＊（穴数）
    - 立方体： 8 – 12 + 6 = 2　四角錐： 5 – 8 + 5 = 2



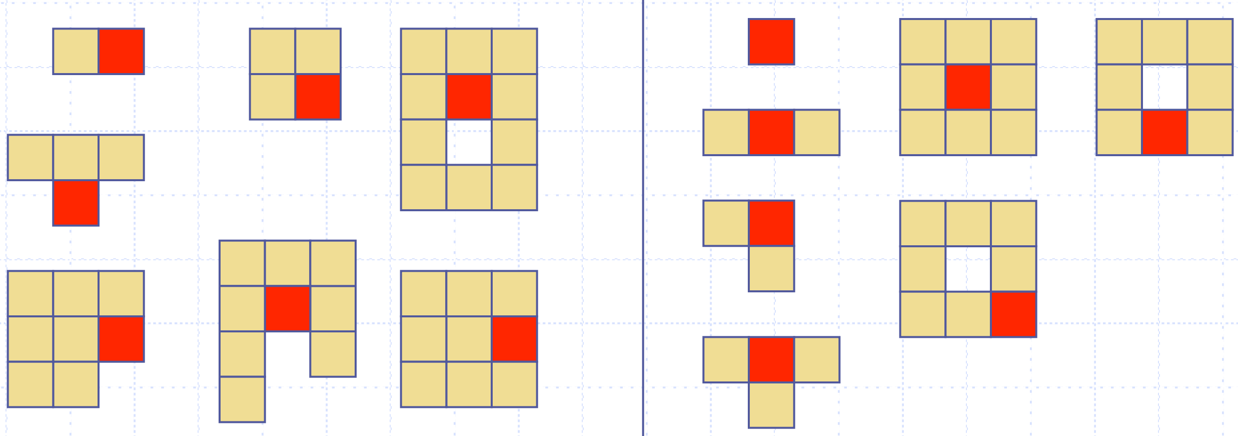
両側が四辺形である辺の消去：辺と頂点が1つづつ消える

一方が三角形である辺の消去：辺が2つ消え，面と頂点が1つづつ消える

→ 最終的に4面体（オイラー数：２）に帰着

消去可能性

- 消去可能画素とは
  - 画像全体の連結性が変化しない画素
  - 位相構造を変化させずに図形を変換



消去可能　　消去不可能

4近傍の場合

消去可能性の計算

X_4	X_3	X_2
X_5	X_0	X_1
X_6	X_7	X_8

- 以下の式が1のとき消去可能
- 4近傍の場合
  - $X_1 + X_3 + X_5 + X_7$
    $-X_1X_2X_3 - X_3X_4X_5 - X_5X_6X_7 - X_7X_8X_1$
- 8近傍の場合
  - $\overline{X_1} + \overline{X_3} + \overline{X_5} + \overline{X_7}$
    $-\overline{X_1}\,\overline{X_2}\,\overline{X_3} - \overline{X_3}\,\overline{X_4}\,\overline{X_5} - \overline{X_5}\,\overline{X_6}\,\overline{X_7} - \overline{X_7}\,\overline{X_8}\,\overline{X_1}$
- 8近傍：4近傍の図と地を入れ替えて計算

濃淡のある画像の処理

- 二値画像処理よりも高度
  - 画像の明るさや色を調整する
  - 画像をぼかしたり，鮮明にしたりする
  - 画像から輪郭線を抽出する

・・・などなど

フィルタリング

- 畳み込み演算フィルタ

畳み込み演算

$$* \begin{array}{|c|c|c|c|c|}\hline 1&1&1&1&1\\ \hline 1&1&1&1&1\\ \hline 1&1&1&1&1\\ \hline 1&1&1&1&1\\ \hline 1&1&1&1&1\\ \hline \end{array} \Big/ 25 =$$

$$g(x,y) = \iint k(u,v) \cdot f(x-u, y-v)\,du\,dv$$

$$= k * f$$

畳み込みフィルタの種類

1	1	1
1	1	1
1	1	1

平滑化

0	0	0
-1	1	0
0	0	0

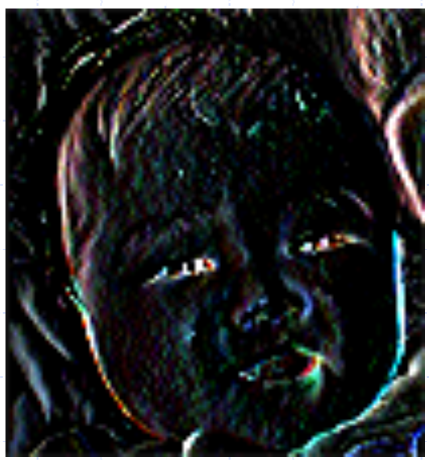
微分
（距離1）

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

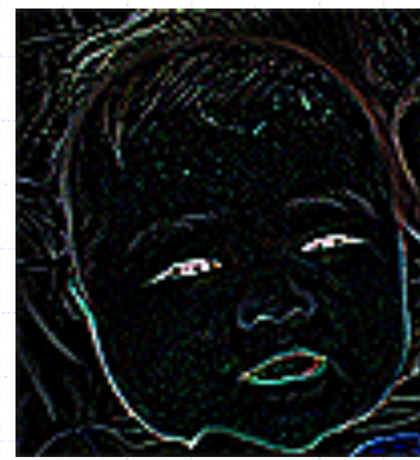
ソーベル

0	1	0
1	-4	1
0	1	0

ラプラシアン



（10倍に明るく）



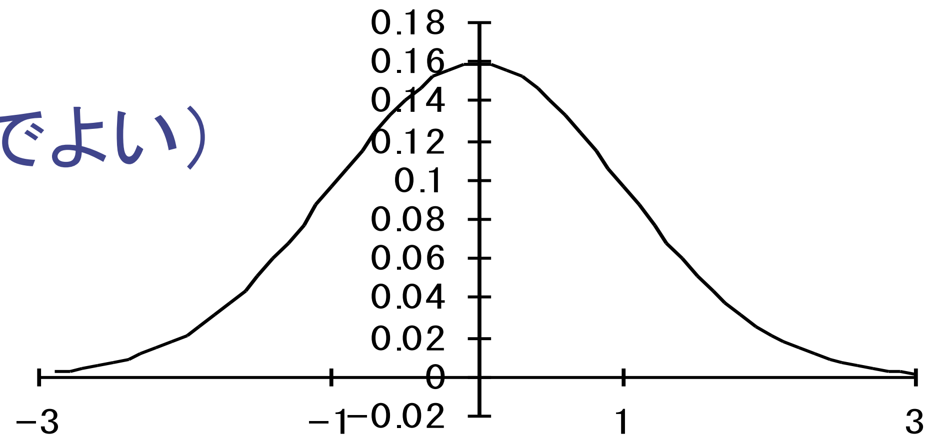
（3倍に明るく）

フィルタの数理的定義

- ガウシアンオペレータ
  - 平滑化オペレータ(数学的意味は後述)
  - 畳み込みカーネル関数

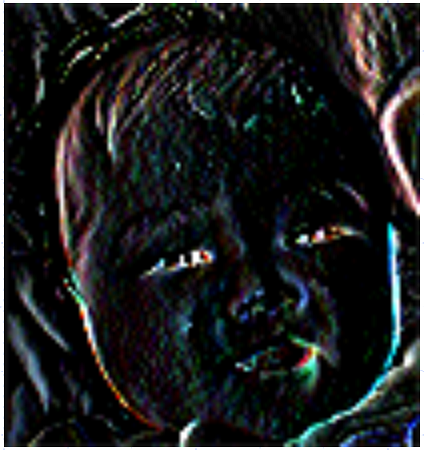
$$k(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

  - 無限に続く関数
(実際には3σぐらいでよい)
  - σは，オペレータの広がり(平滑化の度合い)

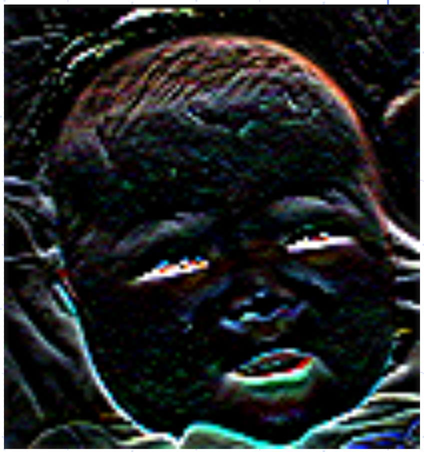


微分フィルタ

X微分



Y微分



- 2次元微分フィルタ

$$\nabla f(x,y) = \left[\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y}\right]$$

  - ベクトル値を持つフィルタ

- エッジ強度

$$Df(x,y) = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2}$$

- ラプラシアン

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

LoG フィルタ

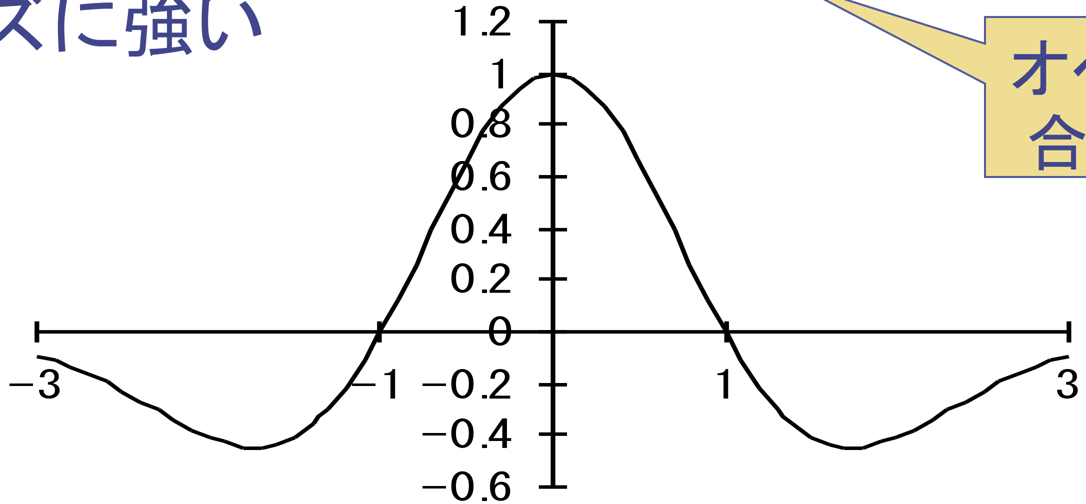
- 平滑化と微分フィルタを組み合わせたもの

ガウシアン

$$LoG(f) = \nabla^2 \cdot \left(G * f \right) = \left(\nabla^2 \cdot G \right) * f$$

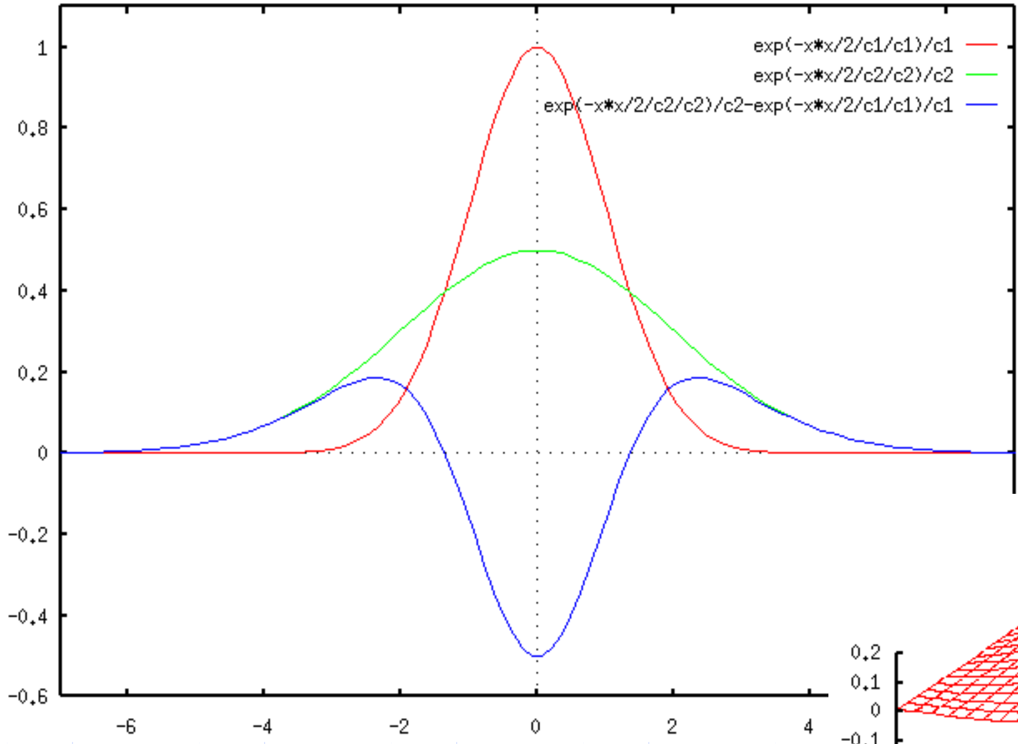
オペレータの合成が可能

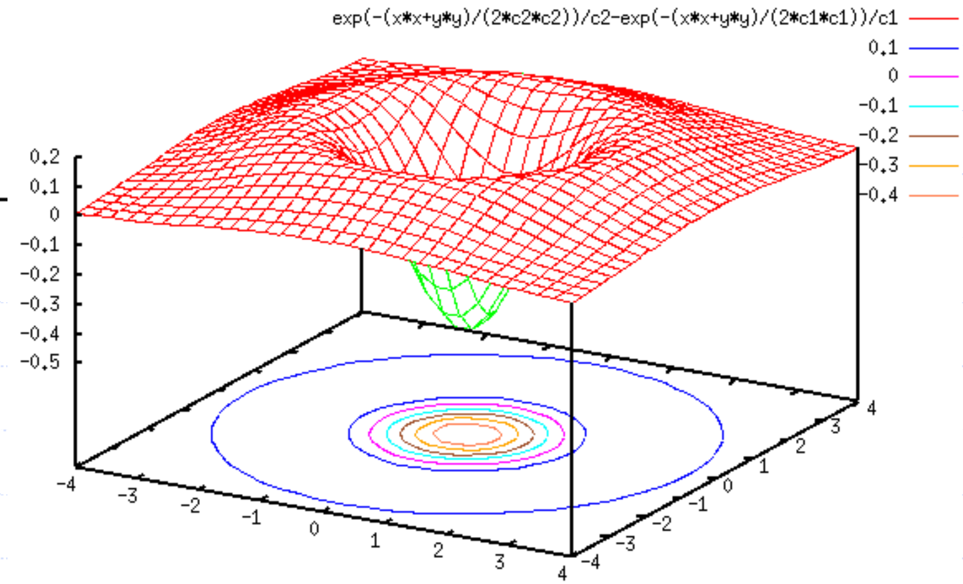
  - ノイズに強い



LoG オペレータ

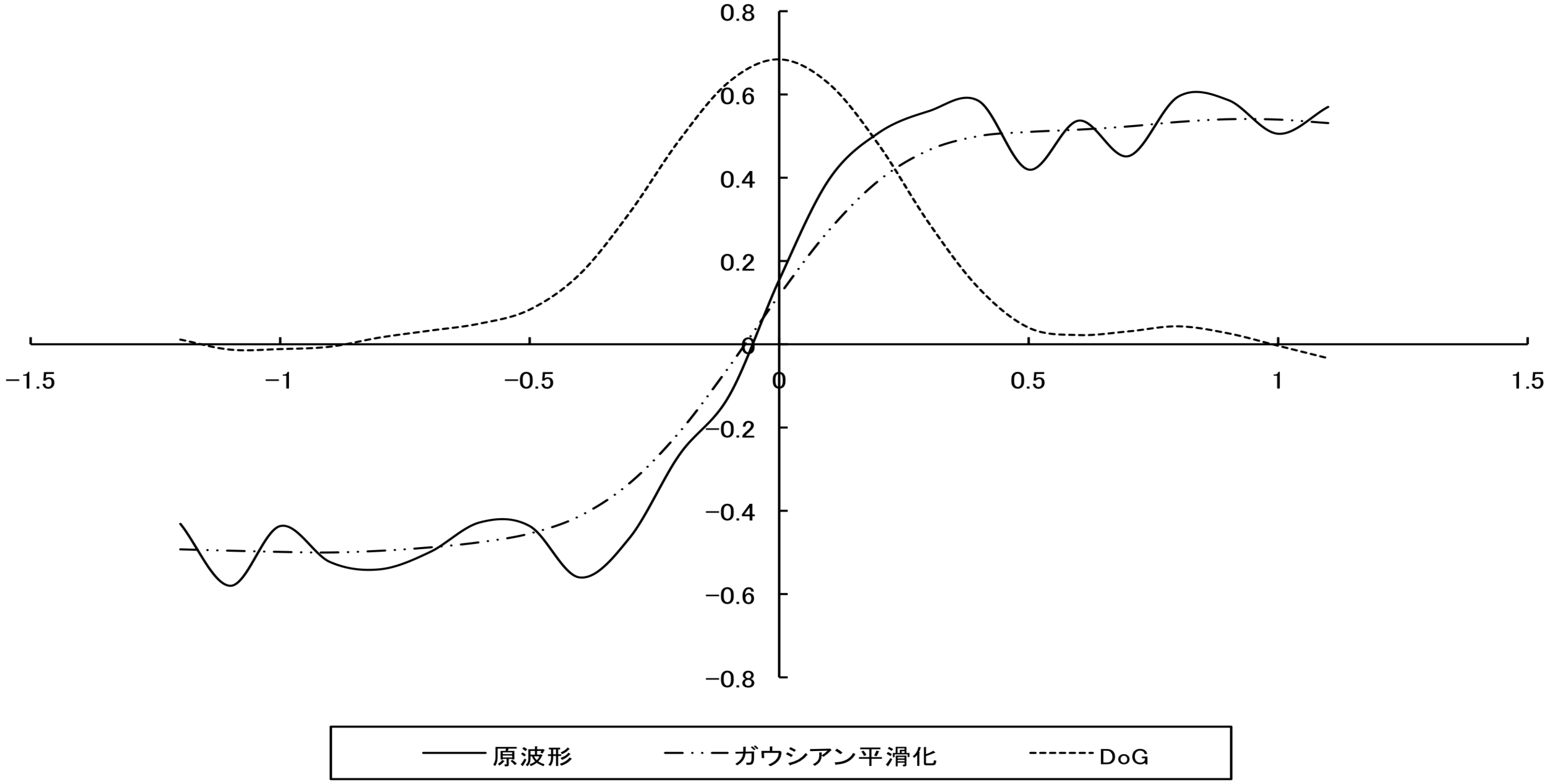
DoG (Difference of Gaussian)





http://fourier.eng.hmc.edu/e161/lectures/gradient/node11.html

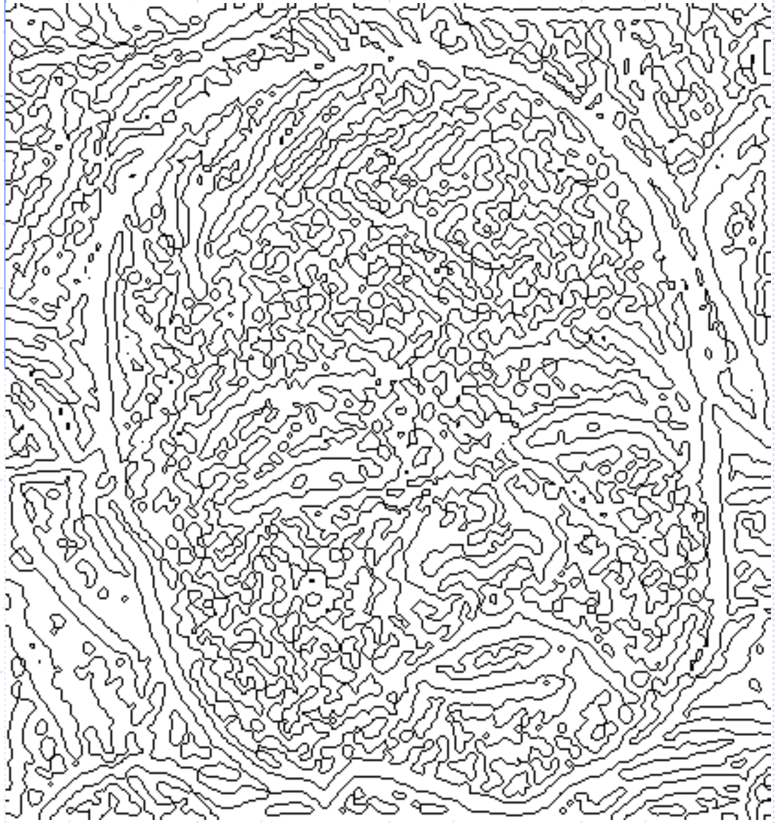
LoG フィルタの効果



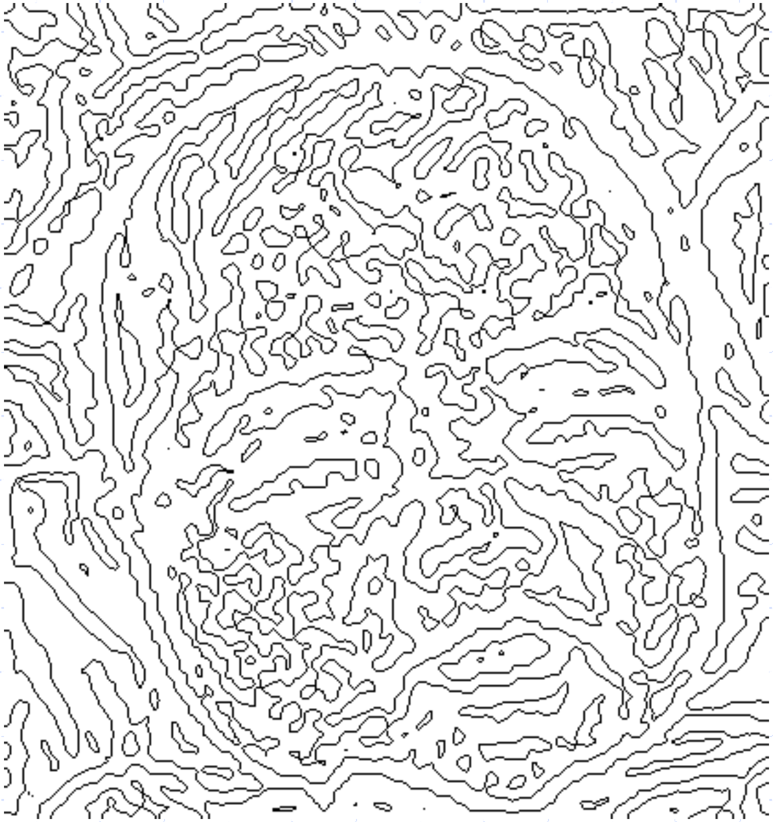
- ノイズに対して強い微分が可能（感応する空間周波数帯を選択可能）

LoG フィルタの例

LoG のゼロクロス抽出



ぼかし: 小



ぼかし: 大

エッジの位置が，若干移動することあり

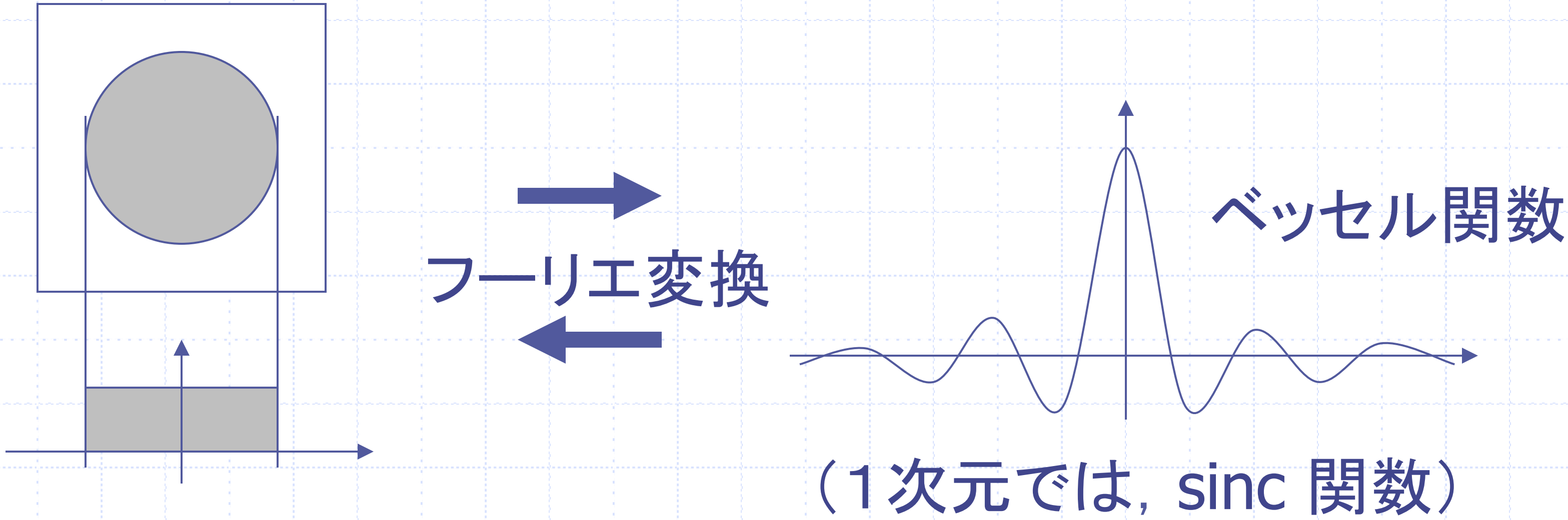
二次元フーリエ変換

- フーリエ変換の定義
  - 連続 $$F(u,v)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\cdot e^{-2j\pi(ux+vy)}dxdy$$
  - 離散 $$S(u,v)=\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}s(m,n)\cdot e^{-2j\pi\left(\frac{um}{M}+\frac{vn}{N}\right)}$$
- 性質

$$f*k=F^{-1}\{F\{f*k\}\}=F^{-1}\{F\{f\}\cdot F\{k\}\}$$

  - 畳み込みは，積に移される
  - 畳み込みオペレータは，空間周波数領域でのフィルタリングに置き換え可能（逆も真）

有限関数のフーリエ変換



- 有限関数のフーリエ変換は，有限ではない
  - ある周波数以上の信号をカットするための畳み込みオペレータの径は無限

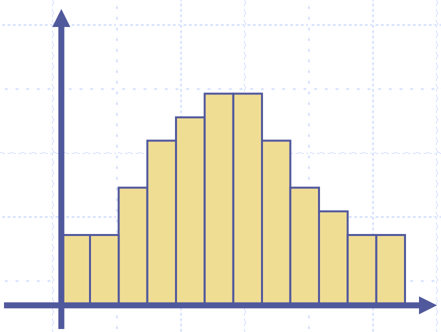
畳み込みOp．のフーリエ変換

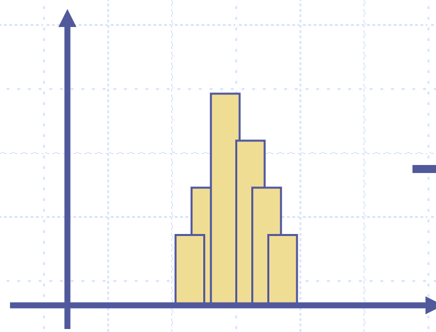
- ガウシアン
  - ガウス関数のフーリエ変換はガウス関数
    → ローパスフィルタ
- 微分･二次微分
  - ハイパスフィルタ
- DoG, LoG
  - バンドパスフィルタ
- 総称して，線形フィルタと呼ばれる

画像圧縮

DPCM

横微分

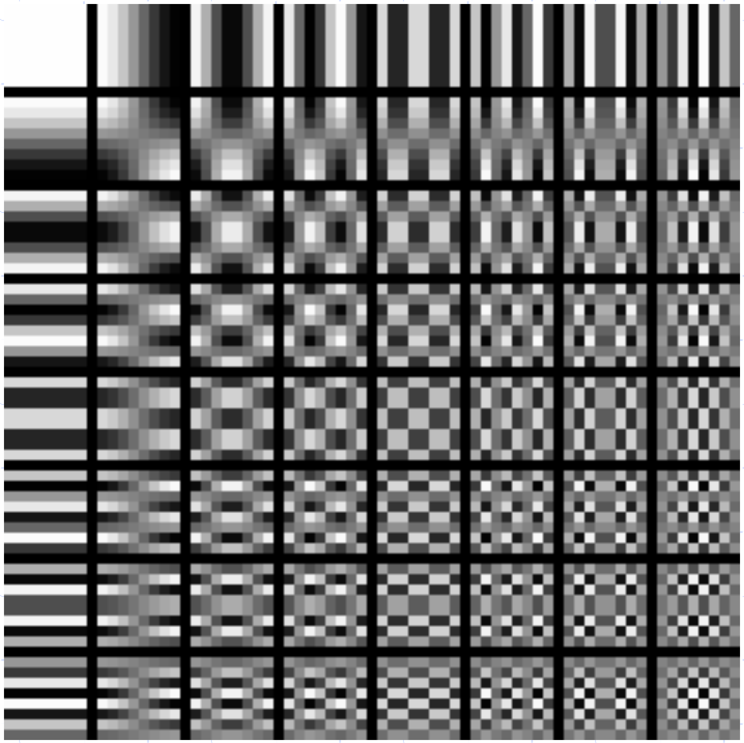




エントロピー符号化
（ハフマンコードなど）

- →ランレングスコード化
  - ランのハフマンコード化など

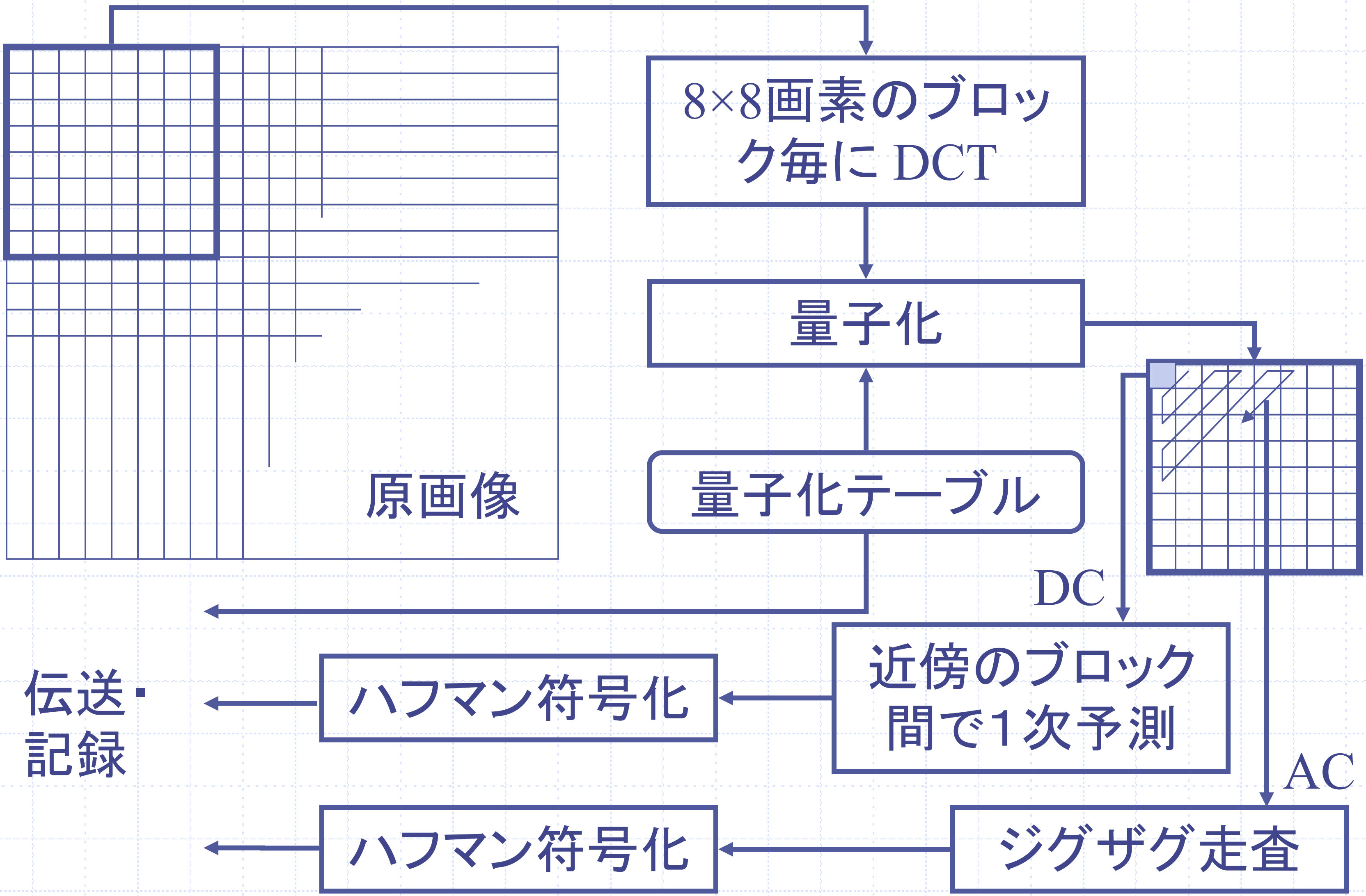
直行変換符号化



離散コサイン変換の基底

- K-L 変換
  - 理論上最高性能
  - 時間がかかる
  - 基底伝送の必要性
- アダマール変換
  - 乗算が不要
- 離散コサイン変換
  - K-L 変換に近い性能

JPEG符号化


原画像
8×8画素のブロック毎にDCT
量子化
量子化テーブル
DC
AC
近傍のブロック間で1次予測
ジグザグ走査
ハフマン符号化
ハフマン符号化
伝送・記録