# バイオメディカルエンジニアリング
# 1, 2 コマ目
# 画像処理概論(2)

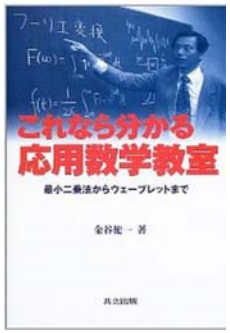
北大 & 理研　連携講座
2014/1/30
担当 井尻敬
理研 基礎科学特別研究員

## バイオメディカルエンジニアリング

- 前期
  - 生体シミュレーションの概要と現状-医療応用
  - 生体シミュレーションの概要と現状-スポーツ応用
  - 画像取得技術
  - 生体材料の力学情報取得技術
  - 画像処理概論(1)　– 画像領域分割の基礎
  - 形状モデリング　– 三次元モデルの表現法
- 後期
  - 画像処理概論(2) –フィルタリングの基礎 -
  - 画像処理演習
  - 構造力学演習 (横田先生)
  - 医療ロボットならびに人由来材料研究の倫理規定 (横田先生)
  - 血流解析演習（1）(姫野先生)
  - 血流解析演習（2）(姫野先生)

## 参考文献


CG-Art協会
ディジタル画像処理




金谷 健一
これなら分かる応用数学教室


新編画像解析ハンドブック

## 資料

USBで配布
+『BioMed2014_2』フォルダをコピーしてください
(演習で使います)

+ Fijiインストール

+ 資料は全て以下のURLより配布予定
www.riken.jp/brict/Ijiri/classes/index.html

## Contents

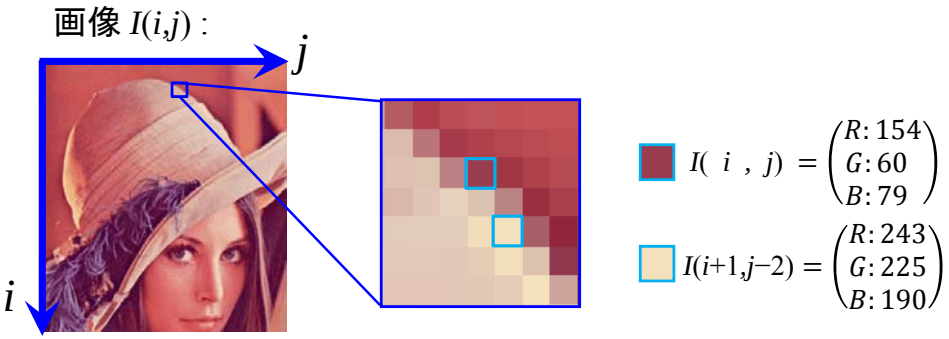
- 画像データ
- 画素ごとのフィルタ処理
- 空間フィルタ処理(線形)
- 空間フィルタ処理(非線形)
- Morphological operation

## 二次元画像

離散値を持つ画素が格子状に並んだデータ

画素 : pixel= picture + element

例) 24 bit bitmap : 各pixelが(R,G,B)毎に整数値[0,255]を持つ



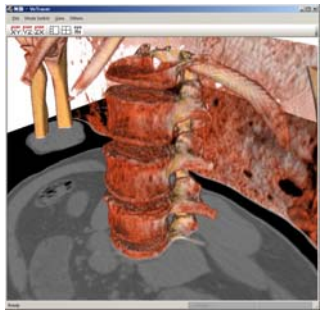

※原点位置は左下のことも多い

## 三次元画像

離散値を持つ画素が格子状に並んだデータ

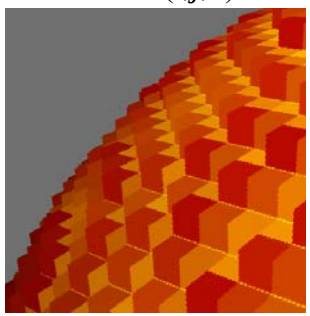
画素 : voxel = volumetric +picture + element

Volume や Volumetric Imageと呼ばれる



CTのVolume Rendering

画像 $I(i,j,k)$



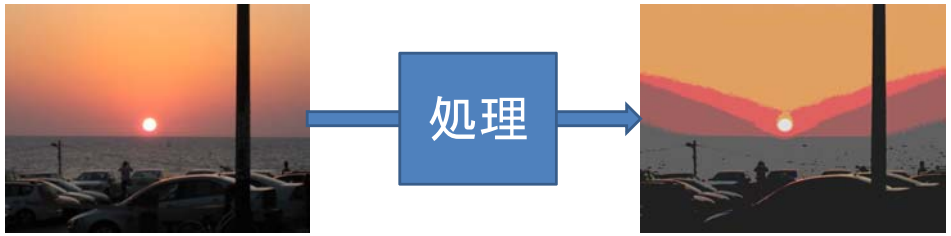
3D画像のVoxel

CT画像は理研生体力学チームより

## Contents

- デジタル画像とは
- 画素ごとのフィルタ処理
- 空間フィルタ処理(線形)
- 空間フィルタ処理(非線形)
- Morphological operation

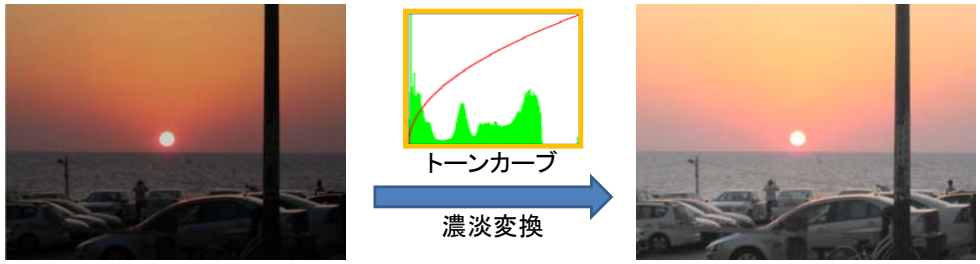
## 画像のフィルタ処理




- 入力画像に対し何らかの計算処理を施し
  - 特定の周波数を持つ信号を強調する・捨てる (ノイズ除去)
  - アーティスティックな効果を得る
  - 画像処理(ステレオ視・領域分割・識別器)に必要な特徴ベクトルを得る

## 画素ごとのフィルタリング

- 各画素のみに注目し、その画素値を何らかの規則に従い変換する処理
- 画像の見栄え(明るさ・コントラスト)を修正するのが主な目的

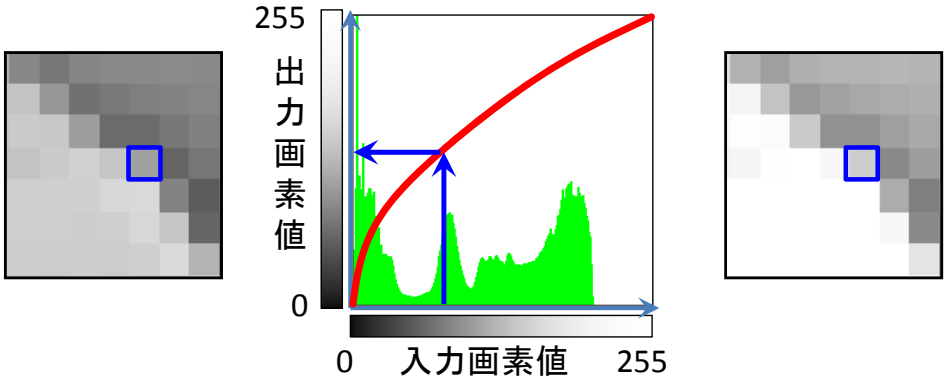



Key words: トーンカーブ, コントラスト補正, ガンマ補正, ネガポジ反転, 二値化, ポスタリゼーション

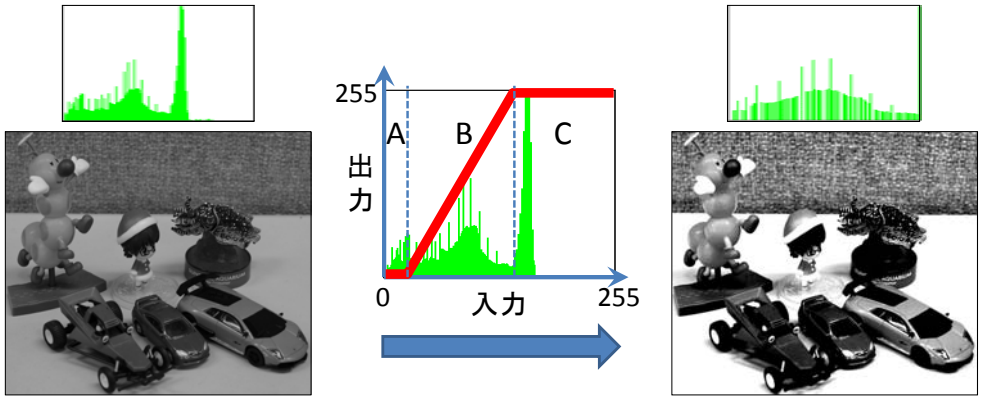
## トーンカーブ

ToneCurve.exe

- 入力画素値 と 出力画素値 の対応を表す関数(階調変換関数)を曲線で表したもの
- 写真編集ツールの基本ツール
- 明暗・コントラスト編集など様々な処理が行える

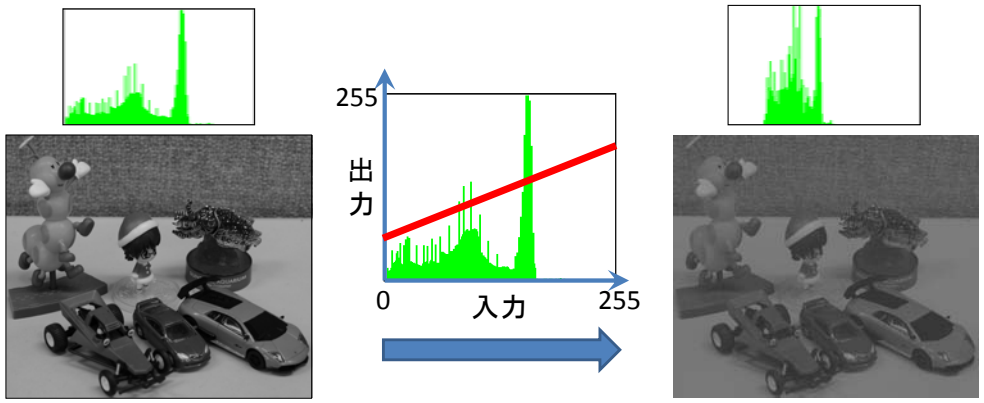



## トーンカーブ : コントラストを上げる




- 領域A : 出力画素値0となり黒つぶれ
- 領域C : 出力画素値255となり白飛び
- 領域B : 傾きが1より大きいため、画素値の取り得る範囲が広がりコントラストが上がる
  画素値は離散値であるため出力ヒストグラムは飛び飛びに

## トーンカーブ：コントラストを下げる



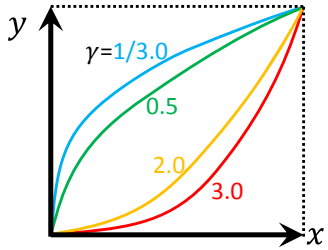

- 傾きが1より小さいため、出力画素値の取り得る範囲が縮まり、コントラストが下がる

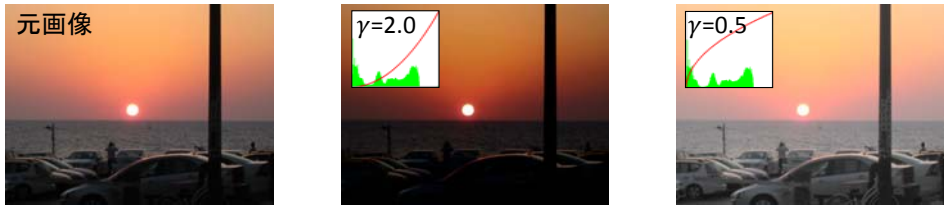
## トーンカーブ：ガンマ補正

次のトーンカーブを利用した濃淡変換をガンマ変換と呼ぶ

$$y = 255\left(\frac{x}{255}\right)^{\gamma}$$

$x$：入力値 [0,255]
$y$：出力値 [0,255]
$\gamma$：パラメータ ( >0)



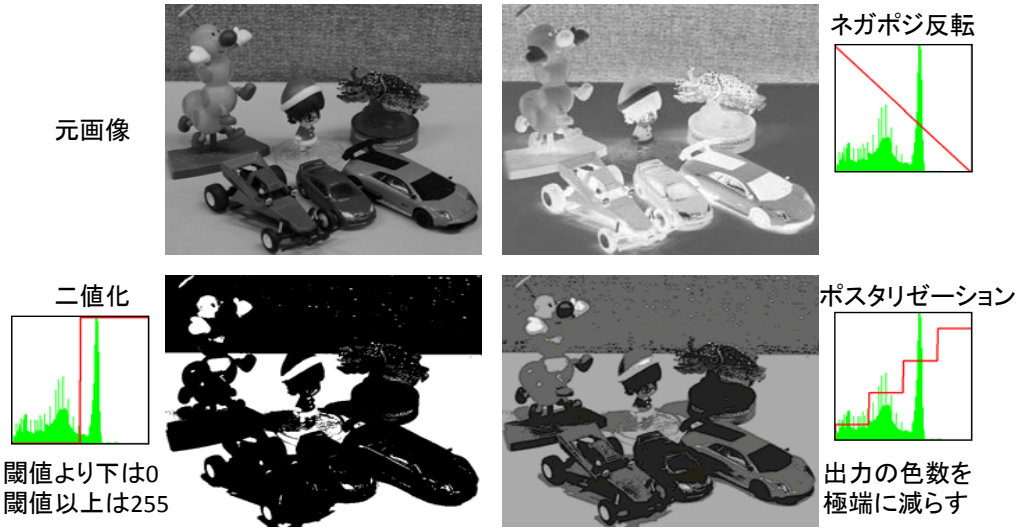




※ RGBの各チャンネルにガンマ補正を適用

※ 画像出力デバイスには『出力値 = (入力値)$^{\gamma}$』と言う関係があり、この特性を補正する目的で上記の関数が用いられていた。これを画像の補正に利用したのがガンマ変換

## トーンカーブ：反転・2値化・ポスタリゼーション

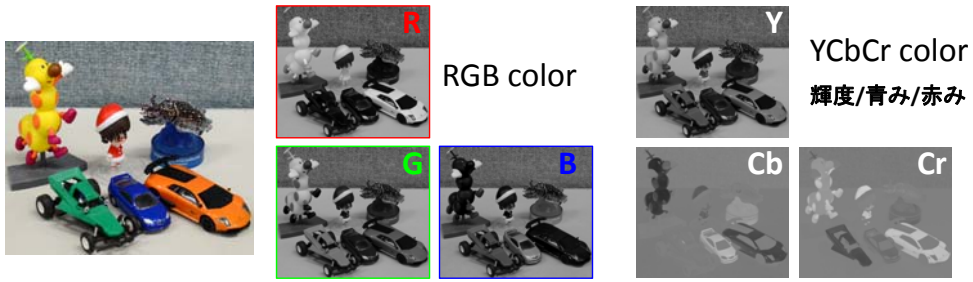
トーンカーブの調節により特殊な効果を持つ編集が可能




## トーンカーブ：Color 画像への適用

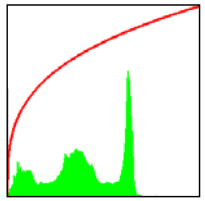
カラー画像をトーンカーブで編集するとき …

- RGBの各チャンネルにトーンカーブの画素値変換を適用
- YCbCr Colorに変換し輝度値成分(Y)のみに変換を適用
- その他



## トーンカーブ : Color 画像への適用 例

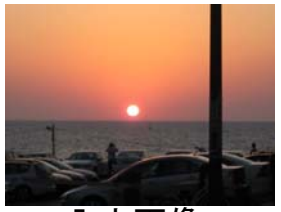

入力画像

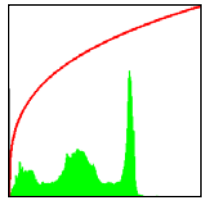

γ=0.3 のガンマ変換


RGB各チャンネル


YCbCrの輝度Yのみ

## トーンカーブ : Color 画像への適用 例


入力画像

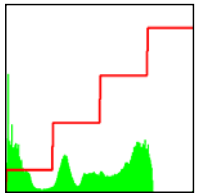

γ=0.3 のガンマ変換


RGB各チャンネル


YCbCrの輝度Yのみ

## トーンカーブ : Color 画像への適用 例


入力画像

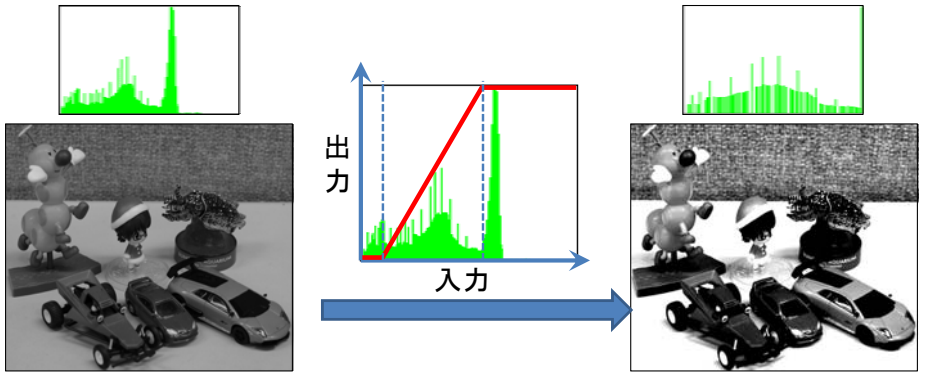

ポスタリゼーション


RGB各チャンネル


YCbCrの輝度Yのみ
（Cb・Crの階調数は減らない）

## まとめ : 画素ごとのフィルタ処理

- 各画素のみに注目し、その画素値を変換するフィルタ
- 変換規則の編集にトーンカーブが利用される
- 画像の見栄え（明るさ/コントラスト/特殊効果）の編集に利用される



## Contents

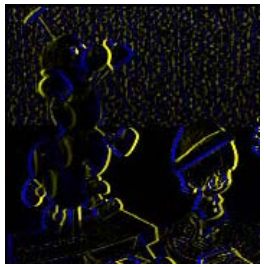
- デジタル画像とは
- 画素ごとのフィルタ処理
- 空間フィルタ処理(線形)
- 空間フィルタ処理(非線形)
- Morphological operation

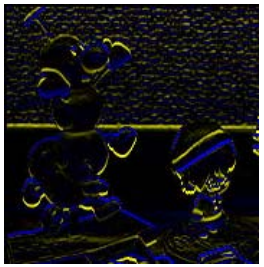
## 線形フィルタの例

ぼかす

先鋭化

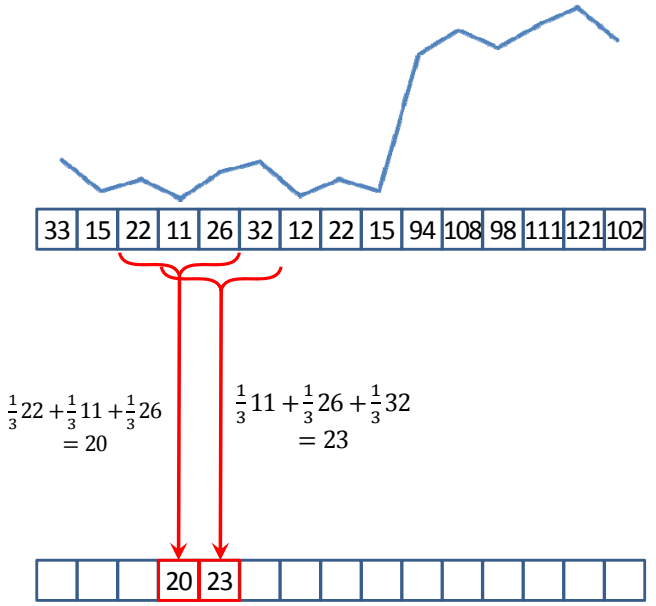
## 線形フィルタの例





エッジ抽出　　横方向　　縦方向

## 線形フィルタの例 1D

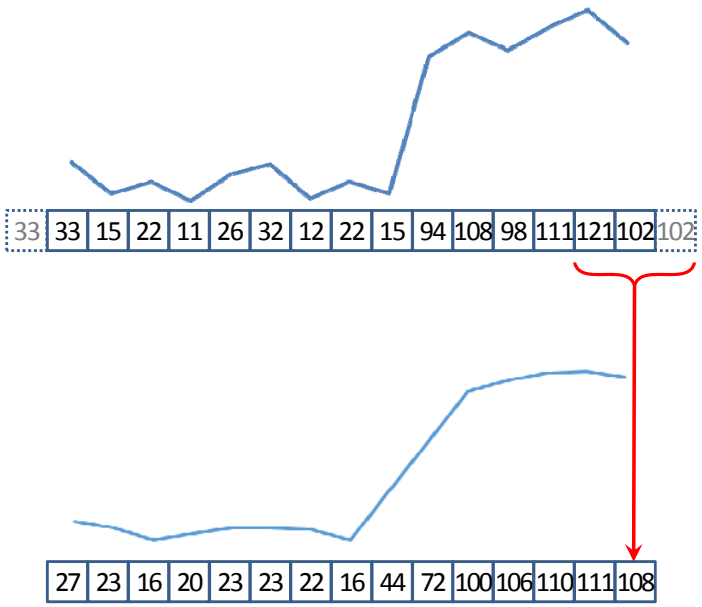



平滑化したい!

| 1/3 | 1/3 | 1/3 |
|---|---|---|

周囲3ピクセルの平均を取る

## 線形フィルタの例 1D



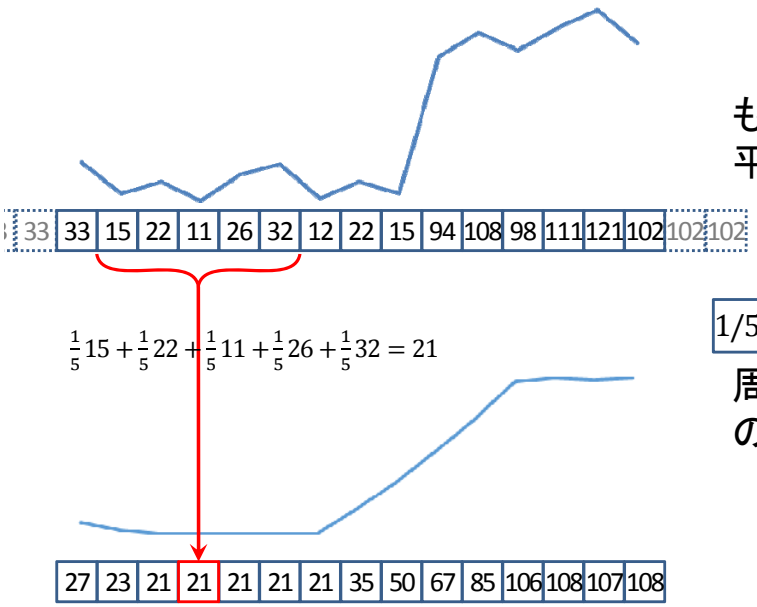

平滑化したい!

| 1/3 | 1/3 | 1/3 |
|---|---|---|

周囲3ピクセル
の平均を取る

※端ははみ出すので値をコピー（ほかの方法もある）

## 線形フィルタの例 1D

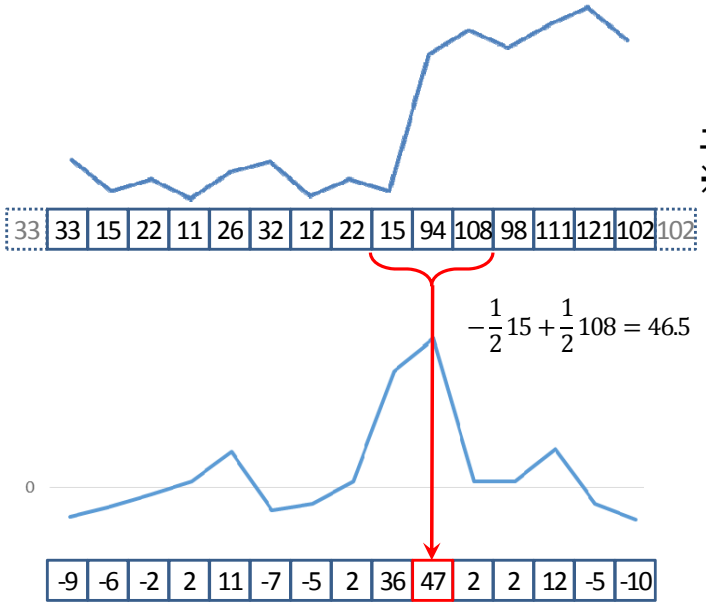



もっと
平滑化したい!

| 1/5 | 1/5 | 1/5 | 1/5 | 1/5 |
|---|---|---|---|---|

周囲5ピクセル
の平均を取る

## 線形フィルタの例 1D



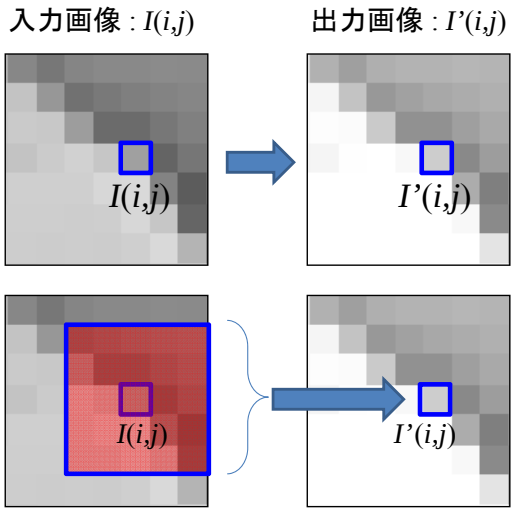

エッジを検出したい
※変化の大きい部分

| -0.5 | 0 | 0.5 |
|---|---|---|

右と左のピクセル
の差をとる

※端ははみ出すので値をコピー（ほかの方法もある）

## 画素ごとのフィルタ ⊂ 空間フィルタ




画素ごとのフィルタ :
出力画素 $I'(i,j)$ を求めるのに
入力画素 $I(i,j)$のみを利用

空間フィルタ:
出力画素 $I'(i,j)$ を求めるのに
入力画素 $I(i,j)$の周囲画素も利用

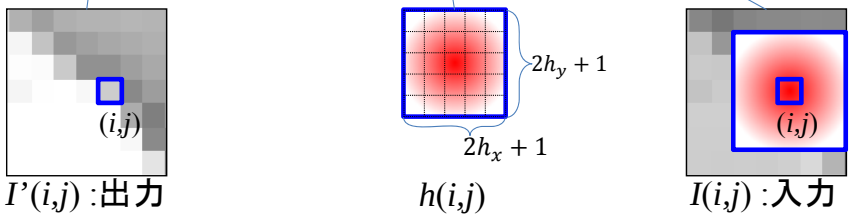
※この領域を窓（Window）と呼ぶ
※窓のサイズは問題/手法に依存して選択する
※一般的に、窓が大きいほうが計算量がかかる

## 線形フィルタ

LinearFilter.exe

出力画素値を周囲画素の重み付和で計算するフィルタ

$$I'(i,j) = \sum_{m=-h_y}^{h_y} \sum_{n=-h_x}^{h_x} h(m,n)\ I(i+m, j+n)$$




$I'(i,j)$ :出力　　$h(i,j)$　　$I(i,j)$ :入力

$h(i,j)$は『カーネル』『フィルタ』
『係数窓』と呼ばれる

## 重み付和と畳み込み

SKIP

$$I'(i,j) = \sum_{m=-h_y}^{h_y} \sum_{n=-h_x}^{h_x} h(m,n) I(i+m, j+n)$$

本講義では、CG-arts協会の教科書にならってこちらを利用

$$(h * I)(i,j) = \sum_{m=-h_y}^{h_y} \sum_{n=-h_x}^{h_x} h(m,n) I(i-m, j-n)$$

線形フィルタをこちらで定義する事も多い
カーネル画像を上下左右反転し重み付和をことと同義

『$*$』は convolution記号

交換　：　$f * g = g * f$

結合　：　$(f * g) * h = f * (g * h)$

分配　：　$f * (g + h) = f * g + f * h$

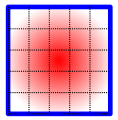
定数倍：　$a(f * g) = af * g$

フーリエ変換：$\mathcal{F}(f * g) = \mathcal{F}(f)\mathcal{F}(g)$

$f * g = \mathcal{F}^{-1}\left(\mathcal{F}(f)\mathcal{F}(g)\right)$

## 重みの正規化

$$I'(i,j) = \frac{\sum_{m=-h_y}^{h_y} \sum_{n=-h_x}^{h_x} h(m,n) I(i+m, j+n)}{\sum_{m=-h_y}^{h_y} \sum_{n=-h_x}^{h_x} h(m,n)}$$



重み係数の総和が1ならば
画像の輝度値の
総和が保存される


添付ソフト

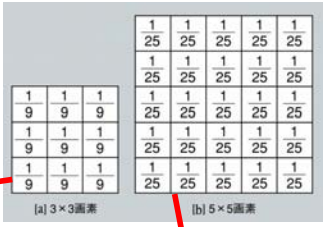
LinearFilter.exe

ここをクリックすると
自動で正規化

## 線形フィルタ：平滑化

LinearFilter.exe


窓内の平均値を新たな
画素値とするフィルタ → ボケる

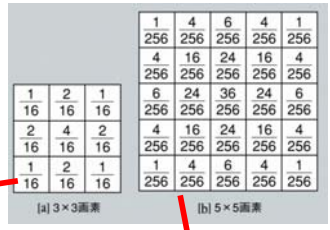




## 線形フィルタ : ガウシアンフィルタ(1/2)

LinearFilter.exe

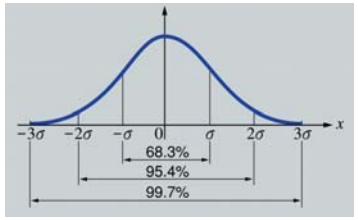

窓内の係数をガウス分布に近づけたもの






## 線形フィルタ : ガウシアンフィルタ(2/2)

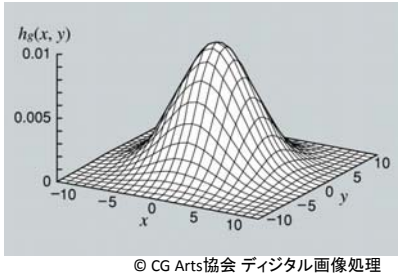
$$g_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{\sigma^2}\right)$$

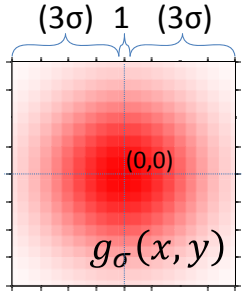



標準偏差σの大きなガウス関数の畳み込みを計算するとき『3×3』や『5×5』の窓では精度が悪い

→ 精度を出すには窓の半径を 3σ程度にすべき
(計算時間はかかる)

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

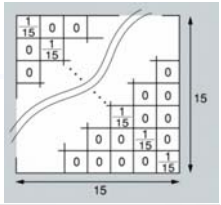


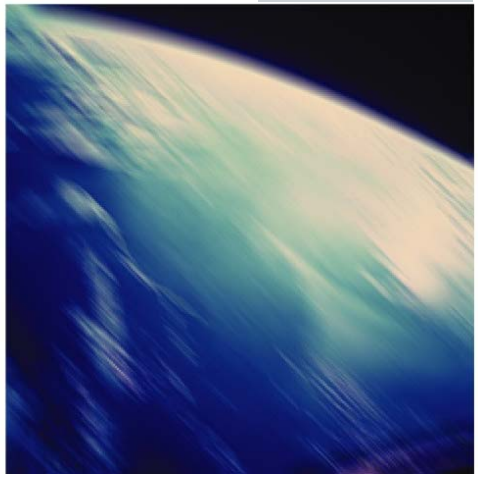




例) σ = 5 pixelの
ガウシアンフィルタ
↓
Window size は
31×31が適当



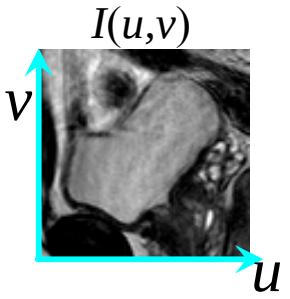
## 線形フィルタ : 特定方向の平滑化

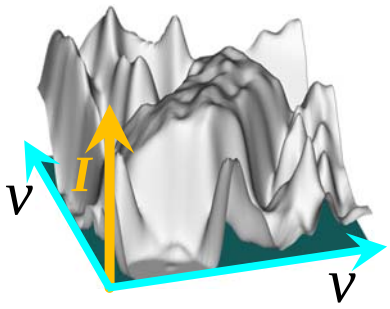




## 線形フィルタ : 微分フィルタ (1/5)




グレースケール画像 $I(u,v)$ は，高さ関数 $z = I(u,v)$ と見なせる
関数$z = I(u,v)$ の微分は、画像の変化の大きさを表す

画像は[Ijiri et al 2013, EuroGraphics]より

## 線形フィルタ : 微分フィルタ (2/5)

LinearFilter.exe

2次元関数 $z= f(x,y)$ の$x$方向偏微分

$$f_x = \frac{\partial f(x,y)}{\partial x} = \lim_{h\to 0} \frac{f(x+h,y) - f(x,y)}{h}$$

画像 $z = I(i,j)$ の横方向偏微分 (近似)

$$I_j(i,j) \approx f(i,j+1) - f(i,j) \quad ...(a)$$

$$\approx f(i,j) - f(i,j-1) \quad ...(b)$$

$$\approx \frac{f(i,j+1) - f(i,j-1)}{2} \quad ...(c)$$

※$h = pitch$ (画素サイズ) $= 1$ とした

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | -1 | 1 |
| 0 | 0 | 0 |

(a)

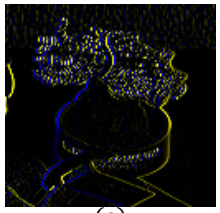
| | | |
|---|---|---|
| 0 | 0 | 0 |
| -1 | 1 | 0 |
| 0 | 0 | 0 |

(b)

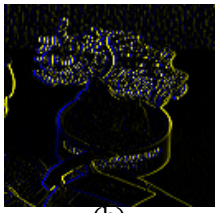
| | | |
|---|---|---|
| 0 | 0 | 0 |
| -1/2 | 0 | 1/2 |
| 0 | 0 | 0 |

(c)

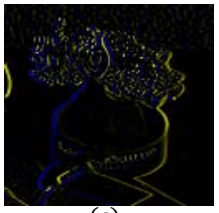
※細かい話だが、
フィルタをConvolutionで定義すると左右が逆転



(a)



(b)

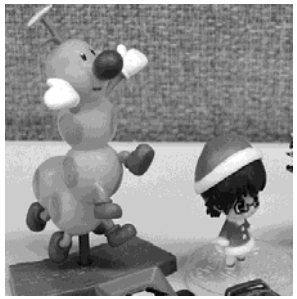


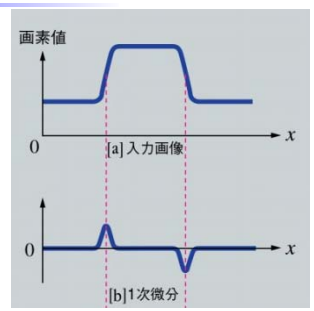
(c)

※ 正値:黄色 , 負値:青 で可視化

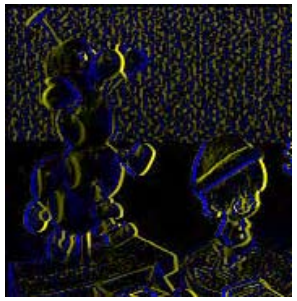
## 線形フィルタ : 微分フィルタ (3/5)

$I(i,j)$
入力画像






微分フィルタには画像のエッジを
検出する効果がある

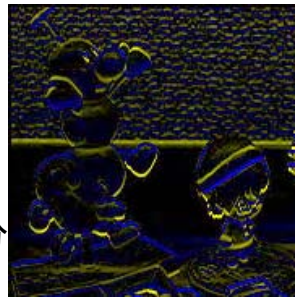


| | | |
|---|---|---|
| 0 | 0 | 0 |
| -1/2 | 0 | 1/2 |
| 0 | 0 | 0 |

$I_j(i,j)$
横方向微分

| | | |
|---|---|---|
| 0 | -1/2 | 0 |
| 0 | 0 | 0 |
| 0 | 1/2 | 0 |

$I_i(i,j)$
縦方向微分



## 線形フィルタ : 微分フィルタ (4/5)

- 前述の単純なフィルタはノイズにも鋭敏に反応する
- ノイズを押さえつつエッジを検出するフィルタが必要

横方向微分 ： 横方向微分 し 縦方向平滑化 する
縦方向微分 ： 縦方向微分 し 横方向平滑化 する

Prewitt filter

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

Sobel filter

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

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

## 線形フィルタ : 微分フィルタ (5/5)

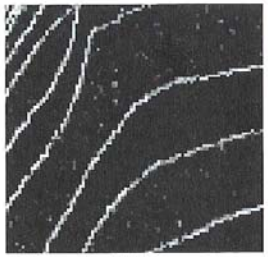

[a] 入力画像


[b] 微分フィルタ


[c] ソーベルフィルタ

画像は勾配強度

$$|\nabla \mathrm{I}| = \left(I_i{}^2 + I_j{}^2\right)^{\frac{1}{2}}$$


[d] 微分フィルタ（拡大）


[e] ソーベルフィルタ（拡大）

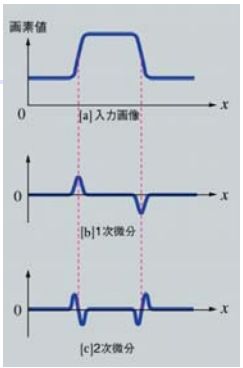
## 線形フィルタ：2階微分フィルタ

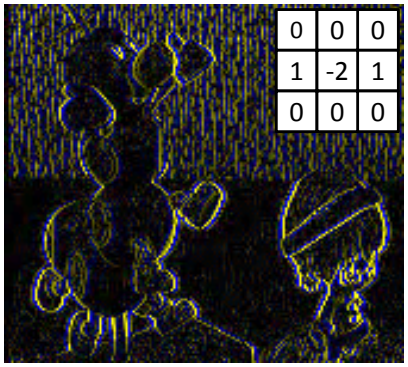
2次元連続関数 $f(x,y)$ の2階偏微分は…

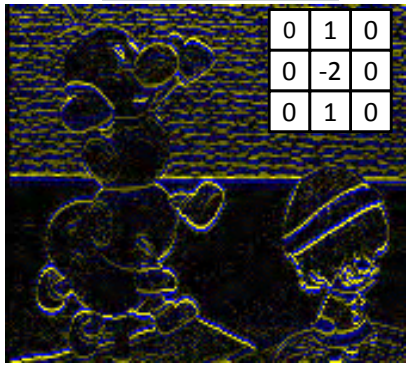
$$f_{xx} = \frac{\partial^2 f(x,y)}{\partial x^2} = \lim_{h\to 0} \frac{f(x+h,y) - 2f(x,y) + f(x-h,y)}{h^2}$$

2次元画像 $I(i,j)$ の2階偏微分の近似は…

$$I_{jj} = f(i,j+1) - 2f(i,j) + f(i,j-1)$$




## 線形フィルタ：ラプラシアンフィルタ

2次元連続関数 $f(x,y)$ のラプラシアンは…

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

2次元画像 $I(i,j)$ のラプラシアンは…

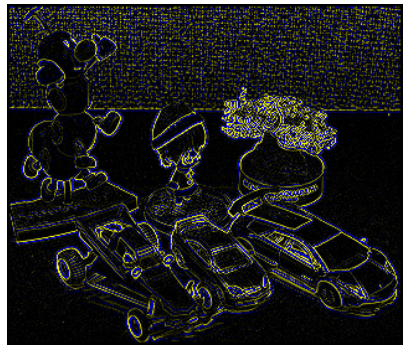
$$\Delta I(i,j) = I_{ii} + I_{jj}$$

$$\Delta I(i,j) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} * \text{[image]} + \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} * \text{[image]}$$

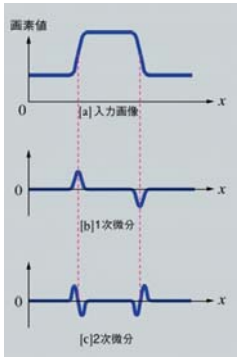
『＊』は convolution

$$= \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} * \text{[image]}$$

ラプラシアンフィルタ



## 線形フィルタ：先鋭化フィルタ





2回微分に関するラプラシアンフィルタを改良すると
画像のエッジを強調する先鋭化フィルタが設計できる

| | | |
|---|---|---|
| | ? | |
| | | |

## まとめ：空間フィルタ(線形)

出力画素値を周囲画素の重み付和で計算するフィルタ

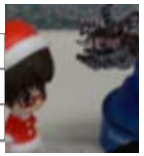
$$I'(i,j) = \sum_{m=-h_y}^{h_y} \sum_{n=-h_x}^{h_x} h(m,n)\ I(i+m, j+n)$$

平滑化フィルタ

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

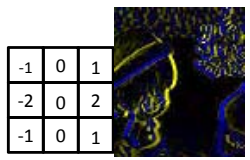
ガウシアンフィルタ

$$\begin{bmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{bmatrix}$$



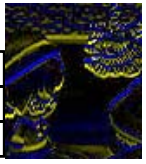
先鋭化フィルタ

| | | |
|---|---|---|
| | | |
| | ? | |
| | | |



Sobelフィルタ(横)

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



Sobelフィルタ(縦)

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

ラプラシアンフィルタ

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

## Contents

- デジタル画像とは
- 画素ごとのフィルタ処理
- 空間フィルタ処理(線形)
- 空間フィルタ処理(非線形) ⬅
- Morphological operation

## 中央値フィルタ(Median filter)

- 中央値 (median)とは…

数字の集合の代表値
数字の小さい順に並べ、ちょうど中央に位置する値

入力 : 6, 2, 1, 5, 3, 12, 1000

平均 : 1/7 x (6+2+1+5+3+12+1000) = 147

中央値 : 1, 2, 3, 5, 6, 12, 1000 → 5

平均値の友達、用途によって使い分ける
→ 平均年収など、外れ値の影響が大きい対象には中央値を

## 中央値フィルタ(Median filter)



[a] 入力画像　[b] メディアンフィルタの結果　[c] 平均化フィルタの結果

+ 画素$(i,j)$を中心とする 幅$h$の窓内の中央値を新しい画素値とする
+ 外れ値（スパイクノイズ）を除去出来る
+ 特徴(エッジ)をある程度保存する

## Bilateral Filter

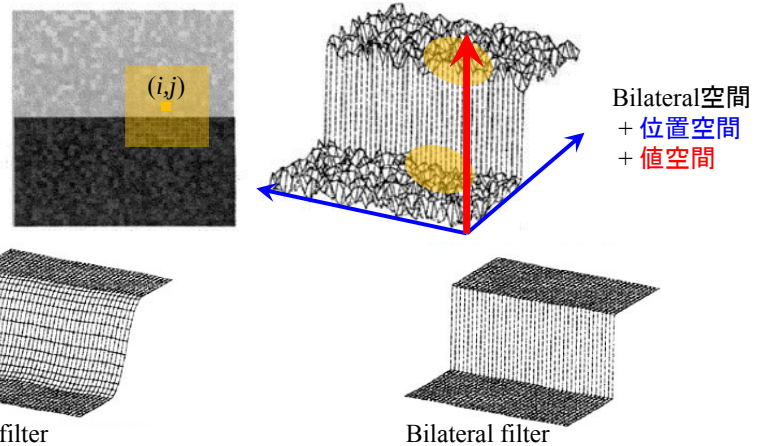
画像中の領域の境界(強いエッジ)をまたがずに平滑化

**単純な平滑化**　**元画像**　**特徴保存平滑化**

(Gaussian filter)　(bilateral filter)

# Bilateral Filter

最も有名な特徴保存フィルタの1つ
空間的距離だけでなく、画素値の差を利用して重み計算




Gaussian filter
位置空間の距離のみ考慮
(遠いほど重みを小さく)

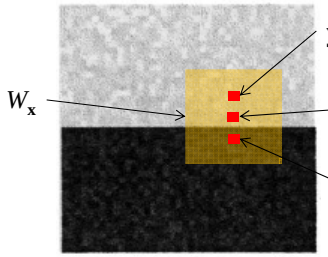
Bilateral filter
位置空間と値空間の距離を考慮
(遠いほど重みを小さく)

画像は, © CG-Arts協会 ディジタル画像処理 より

# Bilateral Filter

$$I_{new}(\mathbf{x}) = \frac{\sum_{\mathbf{y}\in W_\mathbf{x}} h(\mathbf{x},\mathbf{y})\, I(\mathbf{y})}{\sum_{\mathbf{y}\in W_\mathbf{x}} h(\mathbf{x},\mathbf{y})}$$

$\mathbf{x}$ : 注目画素位置 $(i,j)$
$\mathbf{y}$ : 局所窓内の画素 $(i+m, j+n)$
$W_\mathbf{x}$: $\mathbf{x}$が中心の窓(画素集合)




※『カーネル$h$』は窓内の画素値に依存するので線形フィルタではない

Gaussian filter: $h(\mathbf{x},\mathbf{y}) = G_s(|\mathbf{x}-\mathbf{y}|)$

Bilateral filter : $h(\mathbf{x},\mathbf{y}) = G_s(|\mathbf{x}-\mathbf{y}|) \cdot G_h(|I(\mathbf{x})-I(\mathbf{y})|)$

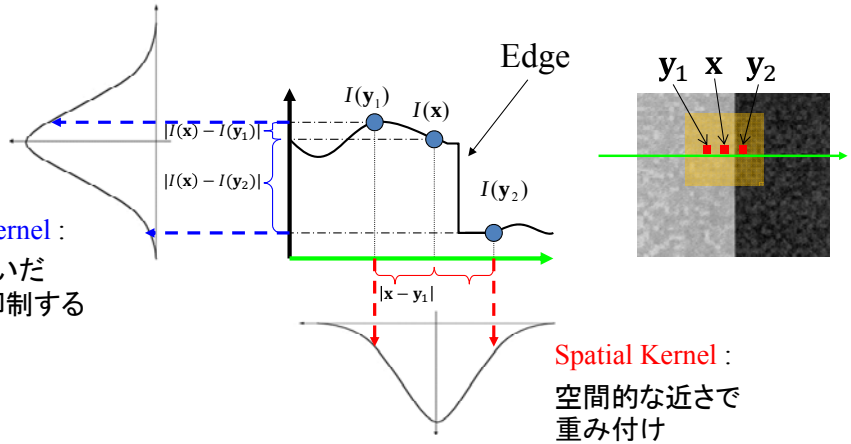
Spatial Kernel　Intensity Kernel

# Bilateral Filter

SKIP

$\mathbf{x} = (i,j)$　注目画素位置
$\mathbf{y} = (i+m, j+n)$　窓内の画素位置

$$h(\mathbf{x},\mathbf{y}) = G_s(|\mathbf{x}-\mathbf{y}|) \cdot G_h(|I(\mathbf{x})-I(\mathbf{y})|)$$




Intensity Kernel :
エッジを跨いだ
平滑化を抑制する

Spatial Kernel :
空間的な近さで
重み付け

# Bilateral Filter

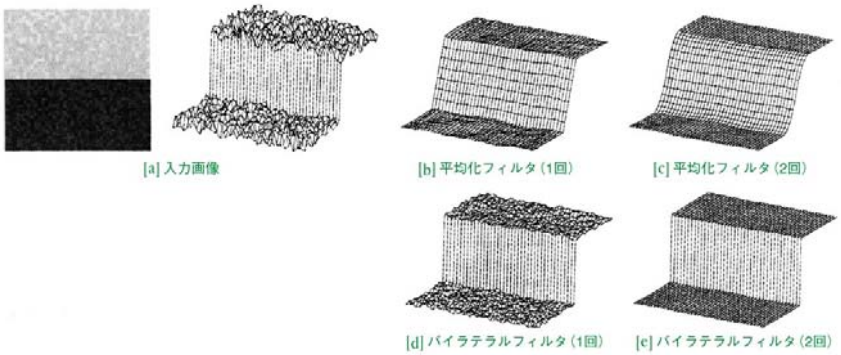
## Bilateral Filter

## Bilateral Filter

$$h(m,n) = G_s(|\mathbf{x}-\mathbf{y}|)\cdot G_h(|I(\mathbf{x})-I(\mathbf{y})|)$$

- パラメータ$h$ : 平滑化したい領域の輝度値の標準偏差の 0.5-2.0倍程度をよく用いる
- 複数回適用すると良い結果が出やすい
- カラー画像はチャンネル毎に処理するのでなく、以下を用いて同じ重みを利用するとよい

$$|\mathrm{I}(\mathbf{x})-\mathrm{I}(\mathbf{y})| = \left|\begin{pmatrix} R(\mathbf{x})-R(\mathbf{y}) \\ G(\mathbf{x})-G(\mathbf{y}) \\ B(\mathbf{x})-B(\mathbf{y}) \end{pmatrix}\right|$$




## まとめ :空間フィルタ処理(非線形)

- 特徴保存フィルタ : 重要な特徴を保存しつつ平滑化
  - メディアンフィルタ
  - Bilateral Filter
  - Non-local means filter, [Buades, CVPR 2005] (省略)
- 近傍画素を距離だけでなく特徴も考慮して混ぜ合わせる


Bilateral Filter

画像はShin Yoshizawaより提供

## Contents

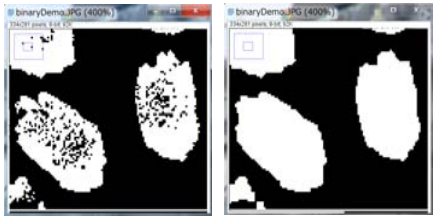
- デジタル画像とは
- 画素ごとのフィルタ処理
- 空間フィルタ処理(線形)
- 空間フィルタ処理(非線形)
- Morphological operation

# Morphological operation

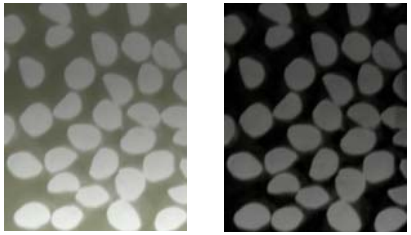
集合論の概念を利用した画像変換法
空隙/ノイズ除去・背景グラデーション除去などに利用可能

Opening: 細かなごみを除去

Dilation-ErosionでEdge抽出

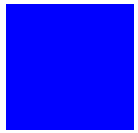


Closing: 抽出した領域にできた穴を除去



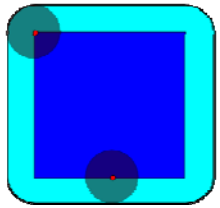
Top-hat: 入力画像のグラデーションを除去

# Morphological operator - 形態作用素-



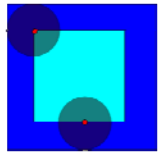
集合**A** (入力2値画像)

集合**B** (Structure Element)



Dilation (膨張)

$\mathbf{A} \oplus \mathbf{B} = \{\mathbf{c} | \mathbf{c} = \mathbf{a} + \mathbf{b}, \mathbf{b} \in \mathbf{B}, \mathbf{a} \in \mathbf{A}\}$
**B**の原点を**A**内で動かしたとき
**B**が描く図形



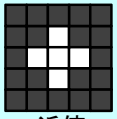
Erosion (収縮)

$\mathbf{A} \ominus \mathbf{B} = \{\mathbf{c} | \mathbf{c} + \mathbf{b} \in \mathbf{A}, \forall \mathbf{b} \in \mathbf{B}\}$
**B**全体が**A**に含まれるよう
**B**を動かしたときBの原点が描く図形

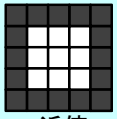
図はwikipediaより

# 2値画像のMorphological operation (1/3)

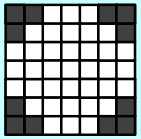
**Structure Element**
+ 2値の線形フィルタのようなもの
+ 円形のものが良く用いられる



4近傍

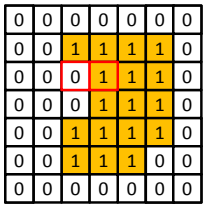


8近傍

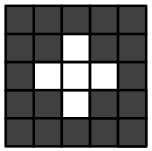


半径3pixelの円

**Basic operations**

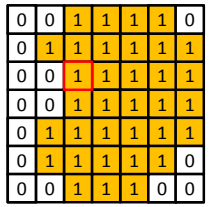


入力画像 $I(\mathbf{x})$



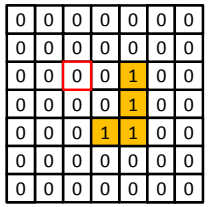
Str. Elem: $B$

Dilation



$(I \oplus B)(x) = \max_{t \in B}(I(x-t))$

Erosion



$(I \ominus B)(x) = \min_{t \in B}(I(x-t))$

※細かいことだが、max($I$(x-$t$))の『マイナス -』は、(ほとんどないけど)左右/上下非対称なStructure elementを利用するとき大切.
計算時は注目画素の周囲の領域の max / minを見るため Structure Elementをひっくり返す必要が有る。

# 2値画像のMorphological operation (2/3)

入力画像 $I(\mathbf{x})$

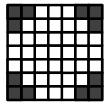
Dilate($I$, 10)

Dilate($I$, 15)

Dilate($I$, 20)

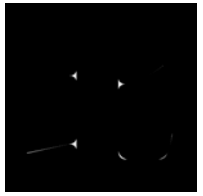
Structure Element
Radius : r-pixel

r=1,



r = 3

Erode($I$, 10)



Erode($I$, 15)

Erode($I$, 20)

※ Dilate(画像, 半径), Erode(画像, 半径),
※ Dilateでは、Structure elementが円なので角が取れて膨張する
※ Erodeでは、Structure element半径より細い構造はすべて消える

# 2値画像のMorphological operation (3/3)

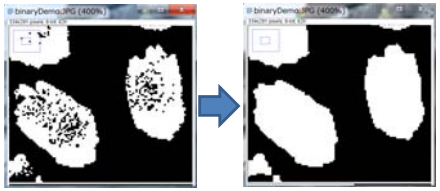
Opening（穴あけ）- 収縮させて →膨張させる
Open(*I*,*r*) = Dilate(Erode (*I*,*r*), *r*)

*r* =10

前景の小さな構造（線・点）を除去

Closing（穴うめ）- 膨張させて →収縮する
Close(*I*,*r*) = Erode(Dilate (*I*,*r*), *r*)

*r* =10

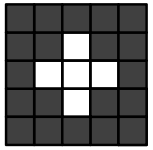


背景の小さな構造（穴）を除去する効果

# グレースケール画像のMorphological operation

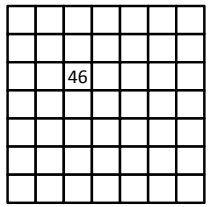
| | | | | | | |
|---|---|---|---|---|---|---|
| 52 | 2 | 3 | 90 | 34 | 53 | 42 |
| 42 | 20 | 46 | 24 | 12 | 3 | 32 |
| 23 | 15 | 10 | 1 | 23 | 4 | 5 |
| 24 | 3 | 4 | 2 | 42 | 2 | 66 |
| 45 | 14 | 58 | 4 | 2 | 23 | 13 |
| 2 | 12 | 6 | 5 | 2 | 2 | 12 |
| 11 | 23 | 56 | 75 | 67 | 7 | 12 |

入力画像 $I(\mathbf{x})$



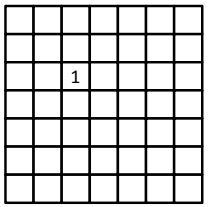
Str. Elem: $B$

Dilation




$(I \oplus B)(x)$
$= \max_{t \in B}(I(x-t))$

Erosion

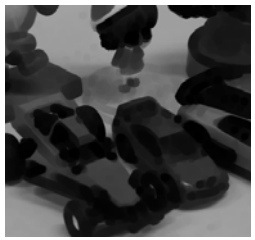



$(I \ominus B)(x)$
$= \min_{t \in B}(I(x-t))$

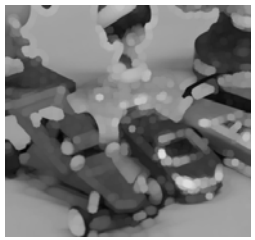
注目画素にStructure Elementを重ね、
周囲の最大値/最小値を新たな画素値とする

# グレースケール画像のMorphological operation

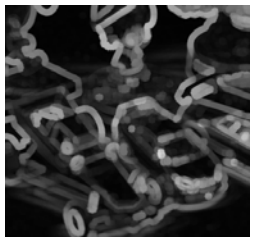
元画像



*Erode*: 明るい領域が収縮する

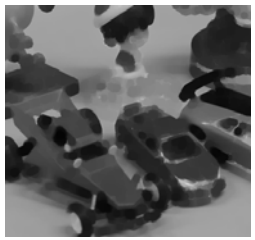


*Dilate*: 明るい領域が膨張する



Dilate – Erode: edgeの
ようなものが抽出出来る

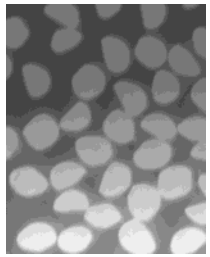
Openinig: 細かい明領域が
閉じる(無くなる)



Closing: 細かい暗領域が
閉じる(無くなる)

Structure elementは、すべて*r =10の円を利*

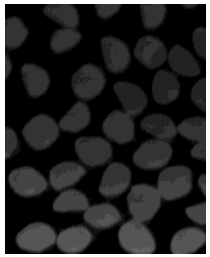
# Top-hat transform による背景除去



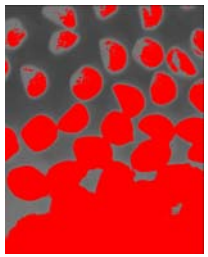
元画像*I*

*Erode*(*I*)

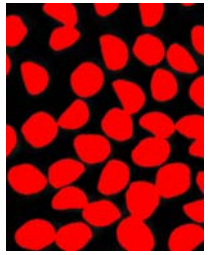
*Open*(*I*)=Dilate(Erode(*I*))



*I - Open*(*I*)



*I*を二値化



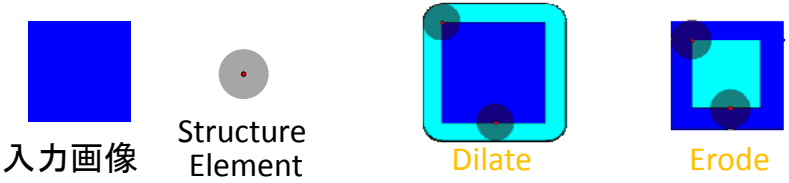
*TopHat*(*I*)を二値化

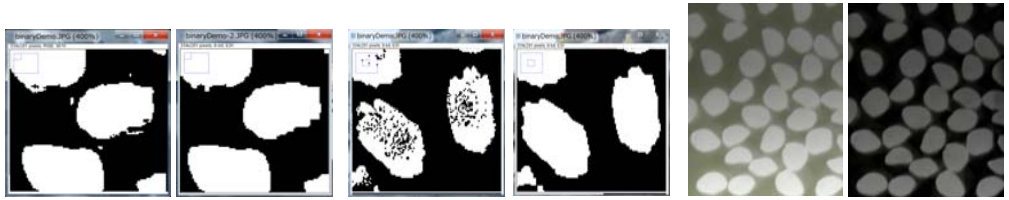
*TopHat*(*I*) = *I* - *Open*(*I*)
*Open*で消えた部分を強調
背景のShadeを消す効果がある
(暗い背景に有向)

*BottomHat*(*I*) = *Close*(*I*) - I
*Open*で消えた部分を強調
背景のShadeを消す効果がある
(明るい背景に有向)

## まとめ : Morphological operations

集合理論に基づく画像処理法



Opening
Dilate(Erode(・))

Closing
Erode(Dilate(・))

TopHat
*I* - Dilate(Erode(I))

## まとめ

- 画素ごとのフィルタ処理
  - トーンカーブ/コントラスト/ガンマ補正/二値可/ポスタリゼーション/ネガポジ反転
- 空間フィルタ処理(線形)
  - 線形フィルタ/Window(窓)/コンボリューション/平滑化フィルタ/ガウシアンフィルタ/微分フィルタ/Prewitt filter/Sobel filter/二階微分フィルタ/ラプラシアンフィルタ/先鋭化フィルタ
- 空間フィルタ処理(非線形)
  - メディアンフィルタ/Bilateral filter/Non-local mean filter/HDRI tone mapping
- Morphological operation
  - Structure element/Erosion/Dilation/Opening/Closing/Top-hat 変換Bottom hat変換

## バイオメディカルエンジニアリング

- 前期
  - 生体シミュレーションの概要と現状-医療応用　(姫野先生)
  - 生体シミュレーションの概要と現状-スポーツ応用 (姫野先生)
  - 画像取得技術　(横田先生)
  - 生体材料の力学情報取得技術(横田先生)
  - 画像処理概論(1)　– 画像領域分割の基礎 (井尻)
  - 形状モデリング　– 三次元モデルの表現法 (井尻)
- 後期
  - 画像処理概論(2) –フィルタリングの基礎 -
  - 画像処理演習
  - 構造力学演習 (横田先生)
  - 医療ロボットならびに人由来材料研究の倫理規定 (横田先生)
  - 血流解析演習（1）(姫野先生)
  - 血流解析演習（2）(姫野先生)

## Image-J

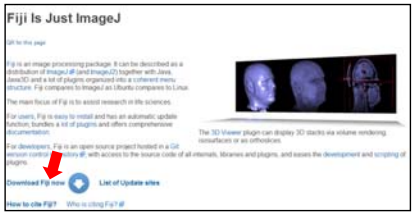
- NIH(アメリカ国立衛生研究所)が開発した画像解析ソフトウエア
  - Java
  - Windows/Mac/Linux
  - Open source
  - http://rsbweb.nih.gov/ij/
- 医用・生物画像の解析に優れ多くの研究者が利用
- 拡張性が高くプラグイン開発可能
- 本講義では『Image-J』を利用して画像処理に触れてみる

## FIJI ( Fiji Is Just ImageJ)

- Web-page http://fiji.sc/Fiji
- Image-Jに基づいた画像処理ソフト (Image-Jの実装の1つ)
- 自然科学者が手軽に利用できるように…
  - インストールが容易
  - 自然科学研究用の画像処理に適したプラグインが充実
  - 各処理に関するドキュメントが充実

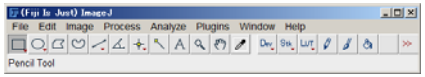
## Fijiを起動する

1. http://fiji.sc/Fiji にアクセス
2. 『Download Fiji now』をクリック



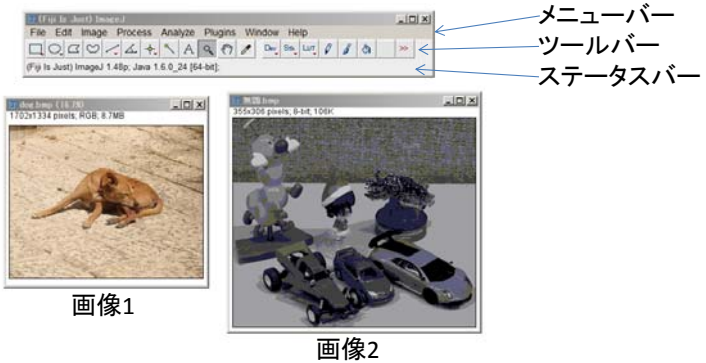

3. OSにあったzipをダウンロード

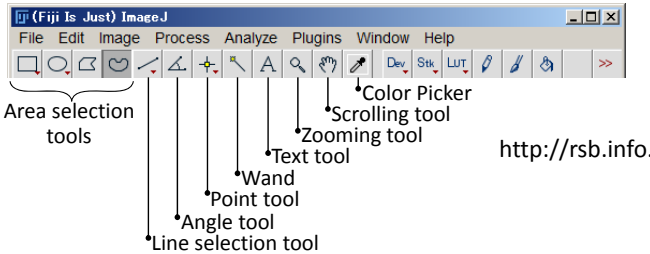

4. zipを展開し『imageJ-win*.exe』をダブルクリック
5. 起動を確認する




※必要なファイルはFijiappフォルダ内にあるので、アンインストールするときはFijiappフォルダを捨てればOK

## Image-J の 基本画面

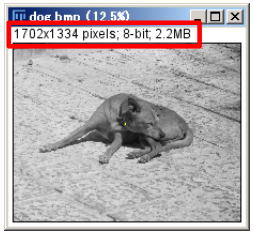


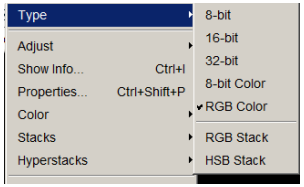




http://rsb.info.nih.gov/ij/docs/tools.html

## 画像の読み込み と Format

1. 画像の読み込み
   『File > Open』をクリックし画像を選択
   画像をImageJ上へドロップしてもOK
2. 画像のFormatを確認
   読み込んだ画像の上部にFormatが表示される
3. Format変換
   『Image > type > *』より変換先を選択
   (Morphologyはグレースケールのみなど、
   処理によって対応していないFormat有り)

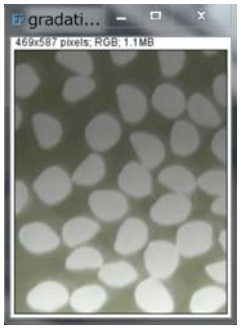



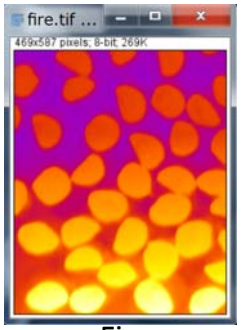



## LUT (Look Up table)

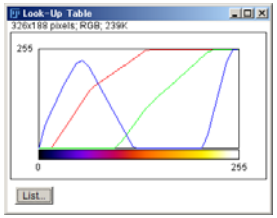
Image-Jにはグレースケール画像に疑似カラーをつけるLUT機能がある
0. グレースケール画像の読み込み
1.『menu > image > lookup tables > *(疑似カラーセット名) 』
2.『menu > image > color > show LUT』でLUTの中身を表示

※ LUTは疑似カラーで表示されるだけで、画像自体がカラーになるわけでない
※『 menu > image > type > 8 bit (※元の画像タイプ)』とするとLUTの効果が消える



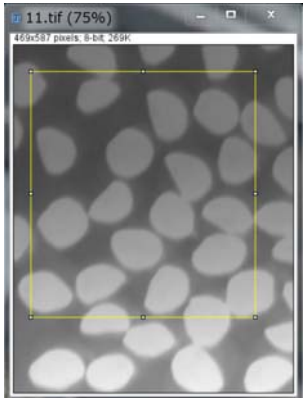



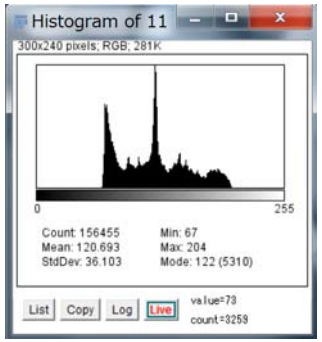

Fire




## 簡単な解析 - 選択領域内のヒストグラム

1. 選択ツールで画像の一部を選択
2.『menu > analysis > Histogram 』もしくは『h』キー
3. Histogram dialogの『live』をonにする
   → 選択領域を変更しながらProfileを確認できる

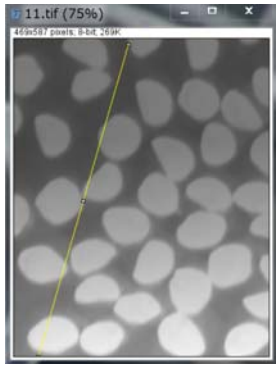


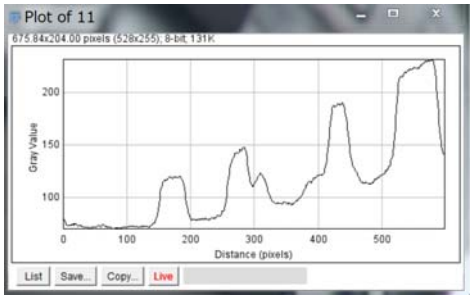




図は、朝顔の種の画像を反転しグラデーションを付加したもの

## 簡単な解析 - 直線に沿った輝度値プロファイル

1. Line tool を選択し読み込んだ画像上にLine配置
2.『menu > Analysis > Plot Profile 』
3. Profile dialogの『live』をonにする
   → lineを変更しながらProfileを確認できる

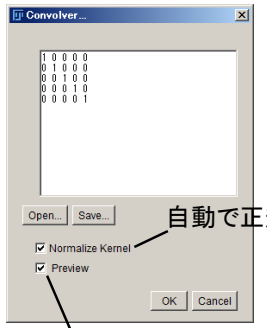



## 線形フィルタ

1.『menu > Process > Filters > Convolve』
2. Dialogで係数を編集する






自動で正規化

チェックすると
結果が即時表示される

## その他のフィルタ

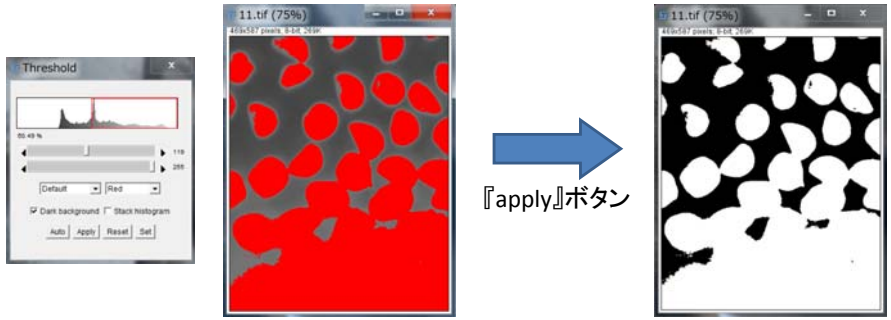
+ Gaussian filer『menu > Process > Filters > Gaussian Blur』
  Dialogから標準偏差を指定する
+ Median filter『menu > Process > Filters > Median』
  DialogからWindow sizeを指定する
+ Bilateral filter『 menu > Plug in > Process > Bilateral Filter』
  Dialogから位置空間と値空間のGaussianの標準偏差を指定
  (8bit gray scaleのみ)

入力　Gaussian　Median　Bilateral

## 二値化 – gray scale

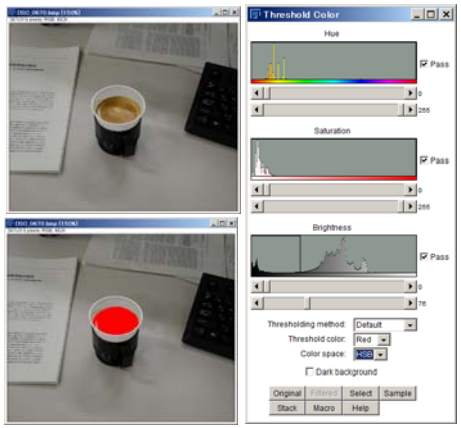
1.『menu > image > adjust > threshold』
2. (手動) ダイアログ内のスライダーで閾値(最大最小)をセット
2. (自動) ダイアログ内のタブから手法を選択し『auto』ボタンをクリック
  ※この時点で前景領域に赤色がつく
  ※この時点では画像は変化せず前景領域が登録される(LUTのようなもの)

3.『apply』ボタンをクリックすると前景→白、背景→黒と二値化される
  (設定によって, LTU-invertが適用され, 前景→黒・背景→白となることも)

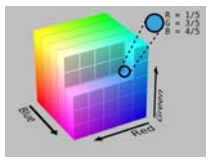



## 二値化 - color

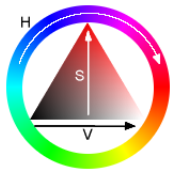
1.『menu > image > adjust > color threshold』
2.『Color space(RGB/HSB/YUV/Lab)』と『Threshold color(マスクの色)』を選択
3. 各channelのthresholdを指定
4.『filtered』ボタンをクリックすると現在のマスクの色が適用される




『Pass』の意味は…
チェックすると、閾値内が前景に
チェックを外すと閾値内が背景に



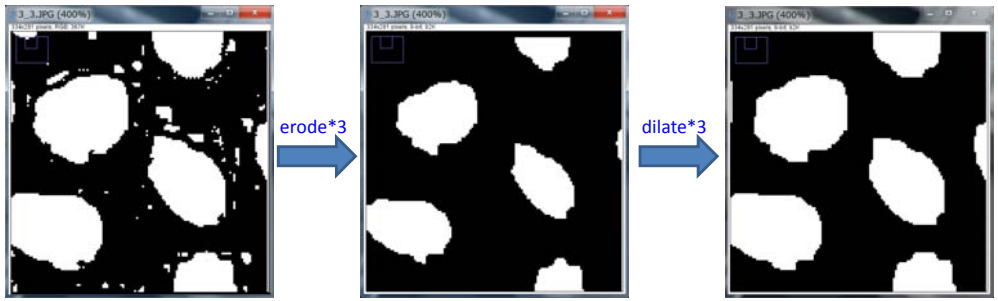
RGB space




HSB space
(色相/彩度/明度)

## Morphological operation (二値画像)

0. 画像を二値化する( menu > image > adjust > thresholdを利用)
1. erosion :『menu > Process > Binary > erode』
1. dilation :『menu > Process > Binary > dilate』
1. closing :『menu > Process > Binary > close 』
1. opening:『menu > Process > Binary > open』

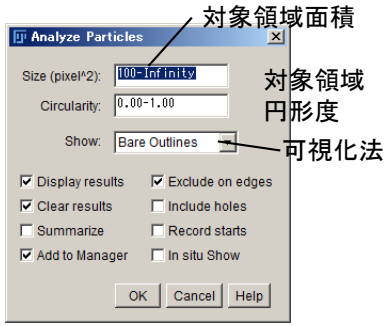


# 非連結領域解析

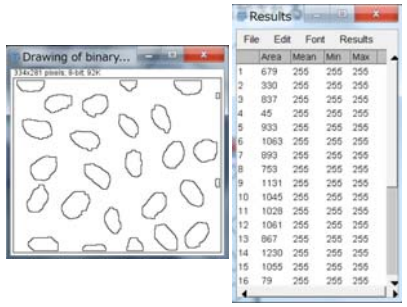
0.『menu > Image > Adjust > threshold』で前景抽出 or 二値化した画像を用意
1.『 menu > Analyze > Analyze particle』を選択
2. Dialogにおいて、対象領域サイズ/円形度/その他を指定
   Display results / Clear resultsはチェックする
   Exclude on edgeをチェックすると 画像の端のparticleは無視される
   Add to managerをチェックすると対象領域が選択状態になり『ROI manager』に追加される
3. 対象領域数と各領域の面積・輝度値が『result dialog』表示される


入力


二値化



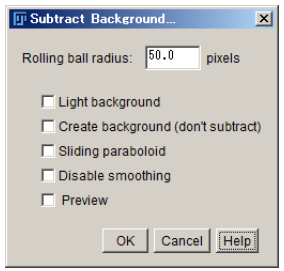

※円形度 = 4π(面積/円周^2)

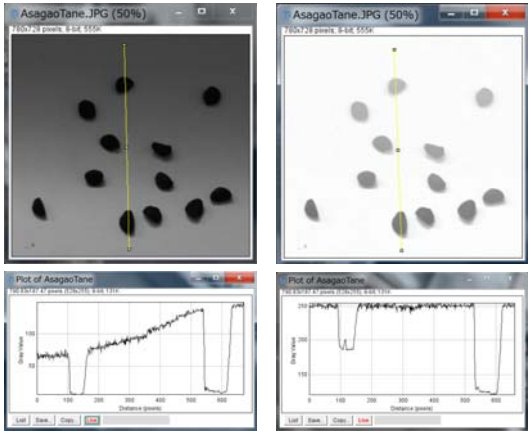



http://www.hm6.aitai.ne.jp/~maxcat/imageJ/menus/analyze.html#AP

# 背景グラデーションの除去方法

Image-Jには TopHat変換以外の背景除去手法が実装されている
http://imagej.nih.gov/ij/docs/menus/process.html#background
1.『menu > Process > Subtract Background』を選択
2. DialogからBall radius (前景領域の半径) を指定し『ok』

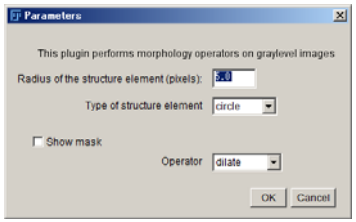


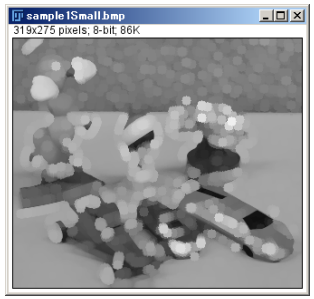

Radius = 100の結果

# Morphological operation – gray scale

0. グレースケール画像を読み込む(※二値画像でもOK)
1.『menu > Process > Morphology > grayscale Morphology』
2. Dialog から Structure element（円）の半径 と operationを指定
3. Dialogの『ok』ボタンをクリック

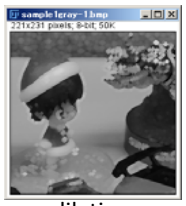




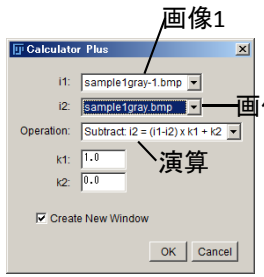


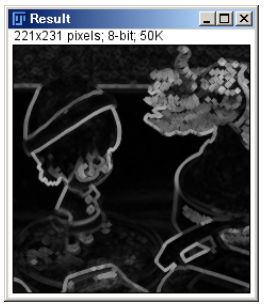

# 画像の加算・減算

0. 2枚の同じサイズのグレースケール画像を読み込む
   - 例えば…
   - 画像を読み込み『menu > Image > Duplicate』で複製
   - 片方に『erosion』,もう一方に『dilation』をかける
1.『menu > Process > Calculator Plus』を起動し Operationを指定し『ok』

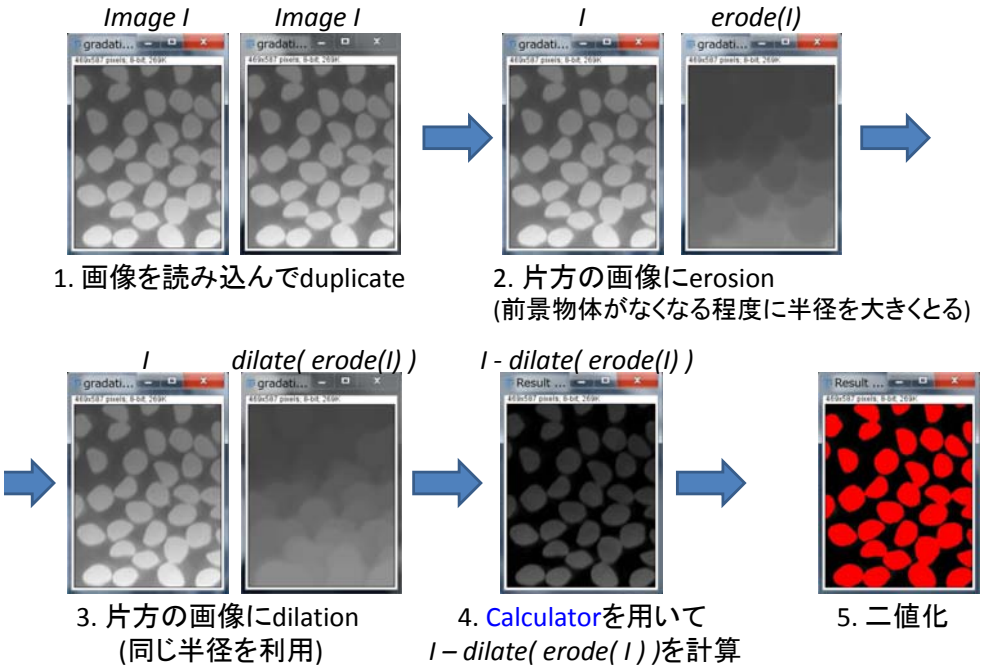

dilation


erosion



# 手動TopHat変換による背景グラデーション除去




1. 画像を読み込んでduplicate

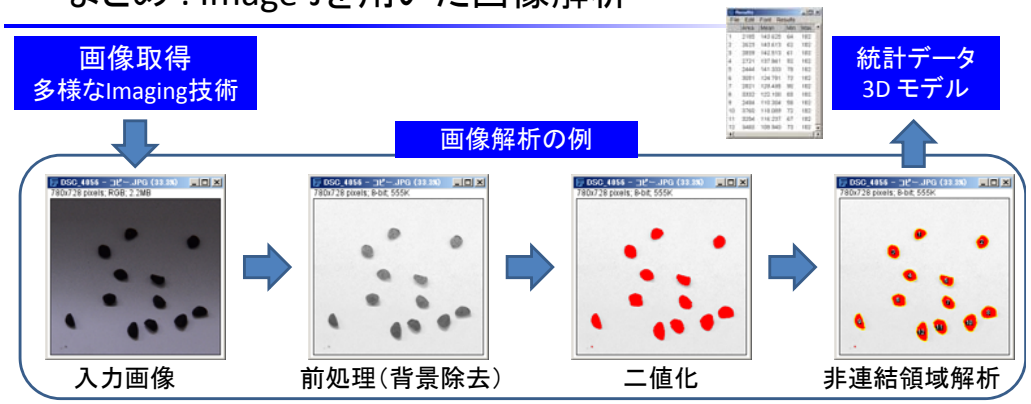
2. 片方の画像にerosion
(前景物体がなくなる程度に半径を大きくとる)

3. 片方の画像にdilation
(同じ半径を利用)

4. Calculatorを用いて
*I – dilate( erode( I ) )*を計算

5. 二値化

# まとめ : Image-Jを用いた画像解析




Image–Jを用いた簡単な画像解析の一例を紹介した
画像解析は、撮影した画像から統計データや3Dモデルを取得するのに不可欠
画像解析は、細かな処理を組み合わせて成り立つことが多い
→ 新しい処理法や既存法の高精度化/高速化が日々研究される
画像処理ソフトはImage-Jだけでない
→ ソフトウエアが違っても個々の処理法や一連の流れは似たようなものなので、
全体の大まかな流れを押さえておけば他のソフトでも困らない

# レポート

提出方法 : 解答用紙 : BioMed2014/レポート課題/report.txt
takashi.ijiri@riken.jp へmail
提出期限 : 近日中ならOKだがなるべく本講義時間内

配点　　: 提出(＋出席)→65, 課題1-4 → 10 (計105点)

# 課題1: 以下の線形フィルタを設計せよ

次の機能を持つ線形フィルタを設計しそれが動く理由を簡潔(1行程度)に述べよ
+ カーネル係数と理由を回答
+ 例題を参考に

入力画像
1orig.jpg

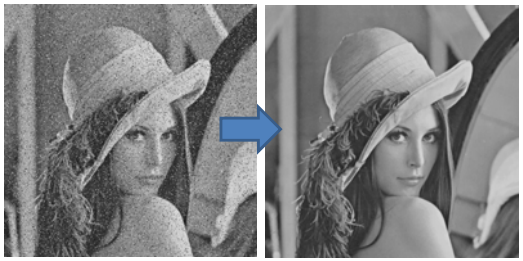
0 0 0
-5 0 5
0 0 0

例)横方向エッジ検出

1_1) 斜めエッジ検出

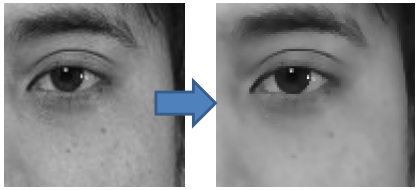
1_2)先鋭化

## 課題2: ノイズ除去をせよ



2_1noise.bmp

2_2trgt.jpg



**課題2-1.** ノイズ画像に何らかのFilterをかけ、元の画像に近づけよ.

**課題2-2.** 人物画像 に何らかのFilterを掛け、あらを消せ.（自分の顔画像でもやってみてください）

※レポートには利用したフィルタとパラメータを明記する

## 課題3: 種の数を数えよ

- 以下の三枚の画像にある種の数をImage-Jを利用して数え その数と利用した処理の流れを回答せよ

3_1.jpg

3_2.jpg

3_3.jpg

## 課題4: 講義へのコメント

- 講義のコメントを以下の点について、述べてください
  - 難易度・トピックは適切か

    例)簡単・難しい・既知の話題が多い・知らない話題が多い

  - 画像処理に関して、話してほしいトピック
  - その他自由に