

デジタルメディア処理1

担当: 井尻 敬

スケジュール

- 10/01 イントロダクション1: デジタル画像とは, 量子化と標本化, Dynamic Range
- 10/08 イントロダクション2: デジタルカメラ, 人間の視覚, 表色系
- 10/15 フィルタ処理1: トーンカーブ, 線形フィルタ
- 10/29 フィルタ処理2: 非線形フィルタ, ハーフトーニング
- 11/05 フィルタ処理3: 離散フーリエ変換と周波数フィルタリング
- 11/12 画像処理演習1: python入門 (PC教室9,10)
- 11/19 画像処理演習2: フィルタ処理 (PC教室9,10) ※
- 11/26 画像処理演習3: フィルタ処理 (PC教室9,10, 前半部分の課題締め切り 11/29 23:59)
- 12/03 画像処理演習4: フィルタ処理 (PC教室9,10)
- 12/10 画像処理演習5: フィルタ処理 (PC教室9,10, 後半部分の課題締め切り 12/20 23:59)
- 12/17 画像の幾何変換: アファイン変換と画像補間
- 01/07 ConvolutionとDe-convolution (進度に合わせて変更する可能性有り)
- 01/14 画像圧縮 (進度に合わせて変更する可能性有り)
- 01/21 後半のまとめと期末試験

フィルタ処理2: 非線形フィルタ, ハーフトーニング

達成目標

- 非線形フィルタ処理の計算法と効果を説明できる.
- ハーフトーン処理の計算法と効果を説明できる

Contents

- 線形フィルタの復習
- 非線形フィルタ
- ハーフトーニング

復習: 空間フィルタ (線形)

空間フィルタ（非線形）

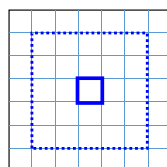
準備:平均と分散

実数値の集合 $\{x_i | i = 1, \dots, N\}$ が与えられたとき、

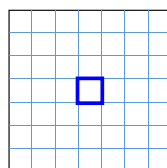
その平均は $\mu = \frac{1}{N} \sum_{i=1}^N x_i$, 分散は $\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2$ で与えられる

- 以下の集合の平均と分散を求めよ
 $\{3, 0, 3, 5, 4, 3, 5, 1\}$
- 以下の集合AとBどちらが分散が大きい
A: $\{3, 4, 3, 4, 3, 2, 2\}$, B: $\{3, 5, 3, 5, 3, 1, 1\}$

エッジ保存平滑化フィルタ



入力画像

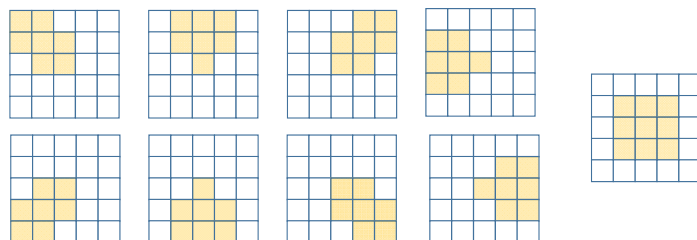


出力画像

- 線形平滑化フィルタでは、画素 (i, j) を計算するため周囲の画素の重み付和を計算した

1/25	1/25	1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25	1/25	1/25

- エッジ保存平滑化フィルタでは、以下9種の領域を考え、一番分散の小さな領域の平均値を、その画素の値とする



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 : $1/7 \times (6+2+1+5+3+12+1000) = 147$

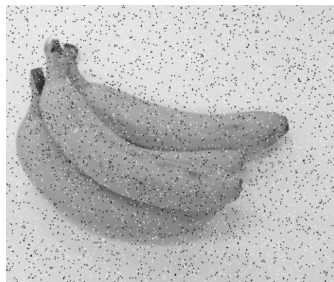
中央値 : 1, 2, 3, 5, 6, 12, 1000 → 5

中央値と平均値は、用途によって使い分ける

→ 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)

ImageJ
Process>Filters>Gaussian Blur
Process>Filters>median



Salt & pepper noise image



Gaussian Blur



Median filter

- + 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

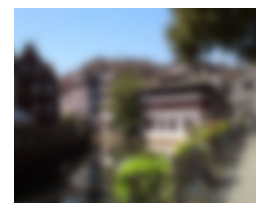
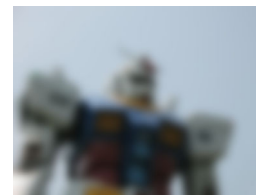
バイラテラルフィルタ

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)

画像の出典
[© Shin Yoshizawa]

バイラテラルフィルタ

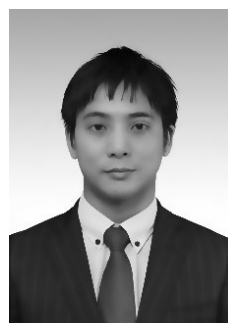
ImageJ
Plug in>Process > Bilateral Filters



Original image



Bi-Lateral Filer
Spatial radi:3
Range radi:50



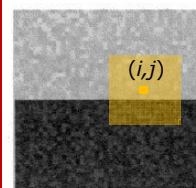
Bi-Lateral Filer
Spatial radi:5
Range radi:80

ブラー効果により顔の"あら"が消える
輪郭が保持されるのでフィルターをかけたことに気づきにくい
あまり強くかけすぎると不自然な画像になる

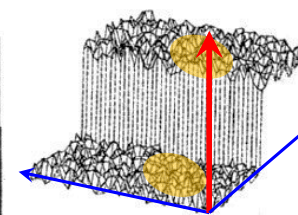
バイラテラルフィルタ

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



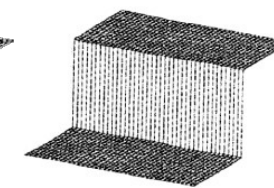
入力画像



Bilateral空間
+ 位置空間
+ 値空間



Gaussian filter
位置空間の距離で重み付け
(遠いほど重みを小さく)



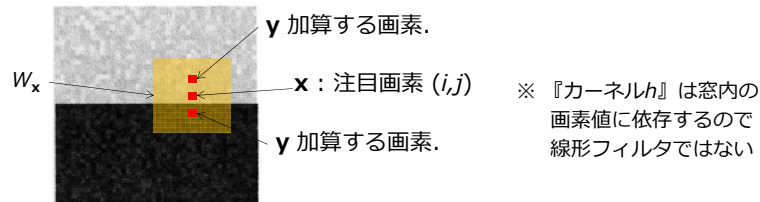
Bilateral filter
Bilateral空間の距離で重み付け
(遠いほど重みを小さく)

画像の出典 [CG-Arts協会 デジタル画像処理 図5.37]

バイラテラルフィルタ

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

\mathbf{x} : 注目画素位置
 \mathbf{y} : 局所窓内の画素位置
 $W_{\mathbf{x}}$: \mathbf{x} が中心の局所窓



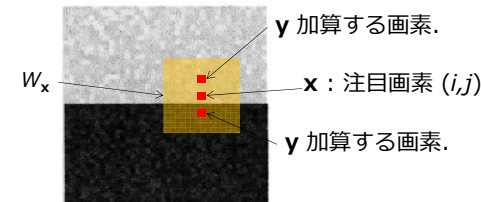
Gaussian filter : $h(\mathbf{x}, \mathbf{y}) = G_s(|\mathbf{x} - \mathbf{y}|)$

Bilateral filter : $h(\mathbf{x}, \mathbf{y}) = \underbrace{G_s(|\mathbf{x} - \mathbf{y}|)}_{\text{Spatial Kernel}} \cdot \underbrace{G_h(|I(\mathbf{x}) - I(\mathbf{y})|)}_{\text{Intensity Kernel}}$

バイラテラルフィルタ

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

\mathbf{x} : 注目画素位置
 \mathbf{y} : 局所窓内の画素位置
 $W_{\mathbf{x}}$: \mathbf{x} が中心の局所窓



Gaussian filter :

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

Bilateral filter :

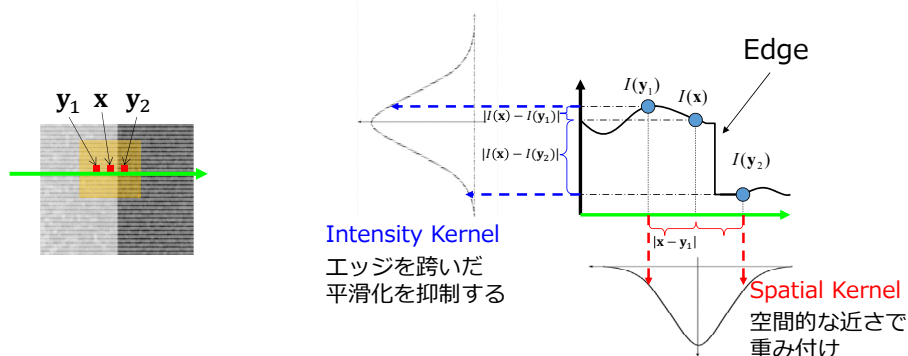
$$h(\mathbf{x}, \mathbf{y}) = \underbrace{G_s(|\mathbf{x} - \mathbf{y}|)}_{\text{Spatial Kernel}} \cdot \underbrace{G_h(|I(\mathbf{x}) - I(\mathbf{y})|)}_{\text{Intensity Kernel}}$$

G_σ は標準偏差 σ のガウス関数

バイラテラルフィルタ

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

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



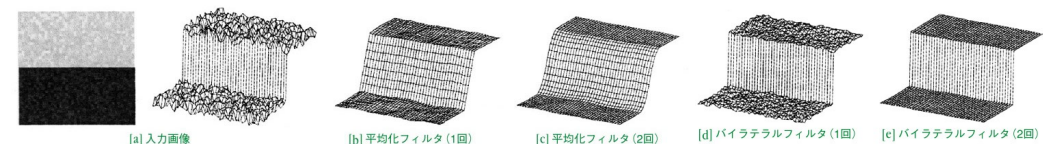
バイラテラルフィルタ (パラメタ)

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

パラメタ 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でなく、以下を用いて同じ重みを利用するとよい

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



まとめ：空間フィルタ（非線形）

- エッジ保存効果のあるフィルタを紹介した
 - エッジ保存平滑化
 - メディアンフィルタ
 - バイラテラルフィルタ
- 線形フィルタと比べ計算量は大きい、特殊な効果が得られる



画像の出典[©Shin Yoshizawa]

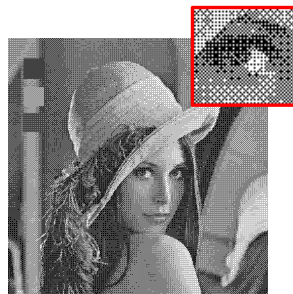
ハーフトーン処理

ハーフトーン処理

- グレースケール画像を白黒2値画像で表現する手法
- ドットパターンにより濃淡を表現する.
- 十分細かなドットパターンは、人の目に濃淡として認識される



濃度パターン法



ディザ法

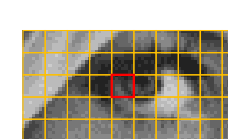
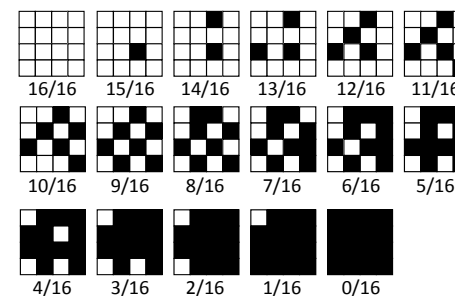


誤差拡散

濃度パターン法

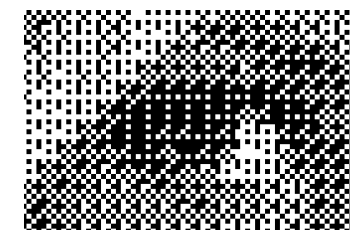
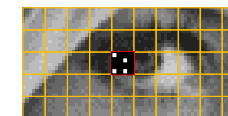
1. 元画像を4X4のブロックに分割
2. 各ブロックの平均輝度値を計算
3. 各ブロックについて似た平均輝度値をもつパターンを選択し、置き換える

※ブロックのサイズは変更可（今回は4x4）



4x4のブロック

平均画素値：73.0
[0,1]に正規化: 0.286
4/17~5/17の範囲なので
パターン4を採用する



欠点：繰り返しパターンが目立つ

ディザ法

1. 元画像を4X4のブロックに分割
2. 4X4のディザパターンを用意
3. 各ブロックの画素においてディザパターンと比較

ディザパターンの値以上 → 白
ディザパターンの値より小さい → 黒

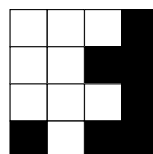
※比較する際、画像の画素値を[0,255]から[0,16]に変更しておく

0	8	2	10
12	4	14	6
3	11	1	9
15	7	13	5

ディザパターン

10	8	2	5
11	7	8	3
9	8	6	4
12	11	6	2

4x4のブロック
[0,16]に変換済み



出力パターン



欠点：繰り返しパターンが目立つ

誤差拡散法

- 左上からラスタスキャンし一画素ずつ以下の通り2値化する
- 注目画素の画素値がIのとき

1. 二値化処理

$I > 127 \rightarrow$ 注目画素を白に

$I \leq 127 \rightarrow$ 注目画素を黒に

2. 誤差拡散

上の二値化で以下の誤差eが発生した

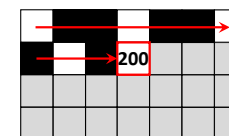
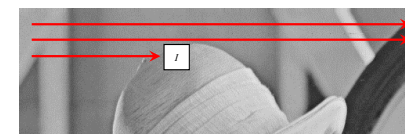
$I > 127 \rightarrow e = I - 255$

$I \leq 127 \rightarrow e = I - 0$

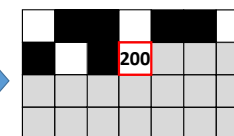
この誤差を隣接画素 I_1, I_2, I_3, I_4 分配
(画素値を変化させる)

$$I_1 \leftarrow I_1 + \frac{5}{16}e, I_2 \leftarrow I_2 + \frac{3}{16}e,$$

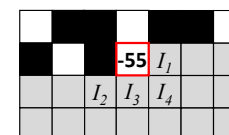
$$I_3 \leftarrow I_3 + \frac{5}{16}e, I_4 \leftarrow I_4 + \frac{3}{16}e$$



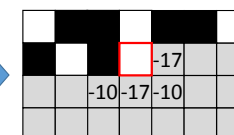
注目画素



二値化



誤差拡散する隣接画素



誤差拡散し画素値を変更

まとめ：ハーフトーン処理

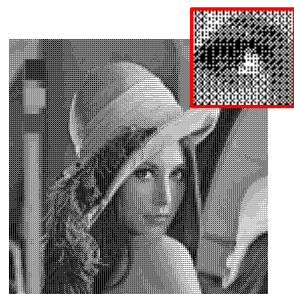
- グレースケール画像を白黒2値画像で表現する手法

濃度パターン法：ブロックの輝度値を利用しパターンで置き換える

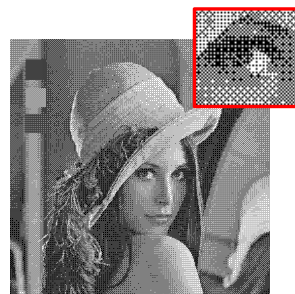
ディザ法：ブロック内でディザパターンと画素値を比較し二値化

誤差拡散法：ラスタスキャンし二値化する。発生した誤差を利用し隣接画素の画素値を変更する

- プログラミング演習で実装します



濃度パターン法



ディザ法



誤差拡散

時間が余ったら フーリエ変換の準備部分をやります