ディジタルメディア処理

担当: 井尻 敬

フィルタ処理2：非線形フィルタ，ハーフトーン処理

達成目標
- 非線形フィルタ処理（エッジ保存平滑化フィルタ）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
- Convolution (畳み込み)
- 非線形フィルタ
- ハーフトーン処理

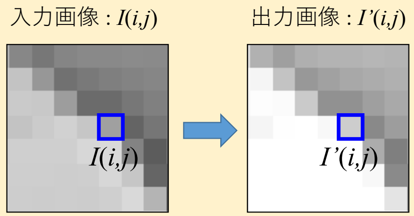
線形フィルタの復習

復習

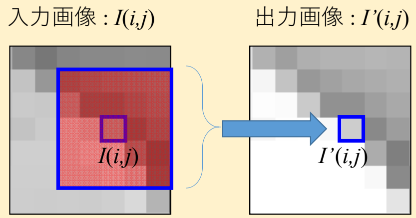
空間フィルタとは

- 空間フィルタとは周囲の情報を利用して画素値を決めるフィルタ
- 空間フィルタは、線形フィルタと非線形フィルタに分けられる

トーンカーブ：
ある画素(i,j)の出力 を求めるのに
その画素のみを利用する



空間フィルタ:
ある画素(i,j)の出力 を求めるのに
その画素の周囲も利用する



復習

1D線形フィルタのイメージ

33	15	22	11	26	32	12	22	15	94	108	98	111	121	102

$\frac{1}{3}22+\frac{1}{3}11+\frac{1}{3}26$
$=20$

$\frac{1}{3}11+\frac{1}{3}26+\frac{1}{3}32$
$=23$

			20	23										

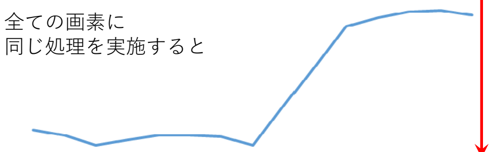
平滑化したい場合、、、
↓
周囲３ピクセルの平均
を取ると良さそう

復習

1D線形フィルタのイメージ

33	15	22	11	26	32	12	22	15	94	108	98	111	121	102	102

全ての画素に
同じ処理を実施すると



27	23	16	20	23	23	22	16	44	72	100	106	110	111	108

※端ははみ出すので値をコピー（ほかの方法もある）

平滑化したい場合、、、
↓
周囲３ピクセルの平均
を取ると良さそう

着目画素・左隣・右隣に1/3を
かけて総和を取る処理を下記の
線形フィルタで表現する

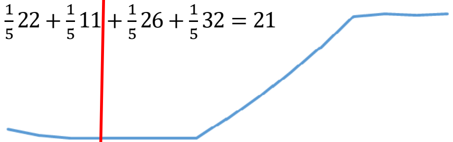
1/3	1/3	1/3

復習

1D線形フィルタのイメージ

33	33	33	15	22	11	26	32	12	22	15	94	108	98	111	121	102	102	102

$\frac{1}{5}15+\frac{1}{5}22+\frac{1}{5}11+\frac{1}{5}26+\frac{1}{5}32=21$



27	23	21	21	21	21	21	35	50	67	85	106	108	107	108

もっと平滑化したい
↓
周囲5画素の平均を取る
と良さそう

周囲5画素に1/5をかけて総和
を取る処理を下記の線形フィ
ルタで表現する

1/5	**1/5**	**1/5**	**1/5**	**1/5**

復習

1D線形フィルタのイメージ

33	33	15	22	11	26	32	12	22	15	94	108	98	111	121	102	102

$-\frac{1}{2}15+\frac{1}{2}108=46.5$

-9	-6	-2	2	11	-7	-5	2	36	47	2	2	12	-5	-10

※端ははみ出すので値をコピー（ほかの方法もある）

変化の大きい部分(エッジ)
を検出したい
↓
『右の画素 - 左の画素』を計
算すると良さそう
※微分から導出される

右に1/2をかけ、左に-1/2を
かけて和を取る処理は下記の
線形フィルタで表現される

-0.5	**0**	**0.5**

復習

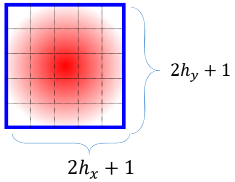
画像の線形フィルタ

周囲画素の重み付和で出力画素値を計算するフィルタ

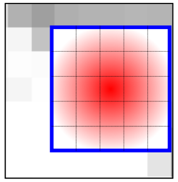
$$I'(i,j) = \sum_{m=-h_y}^{h_y} \sum_{n=-h_x}^{h_x} h(m,n)\ I(i+m, j+n)$$

重みの定義されたフィルタを用意

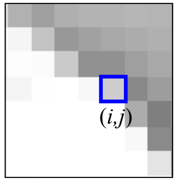
- サイズ 3x3 5x5 などがよく利用される
- 重みによりいろいろな出力が可能



計算時，各画素を中心にフィルタを重ね合わせ重み付き和を計算



$I(i,j)$ 入力画像



$I'(i,j)$ 出力画像

復習

線形フィルタ：平滑化フィルタ

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

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

サイズ 3x3 : 9近傍の平均

サイズ 5x5 : 25近傍の平均

復習

線形フィルタ：ガウシアンフィルタ

係数をガウス分布に近づけ
中央ほど強い重みに

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

$\frac{1}{256}$

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

サイズ 3x3

サイズ 5x5

復習

線形フィルタ :エッジ検出のための**微分フィルタ**

- 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

横方向の変化を検出 ： 横方向微分 し 縦方向平滑化 する
縦方向の変化を検出 ： 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

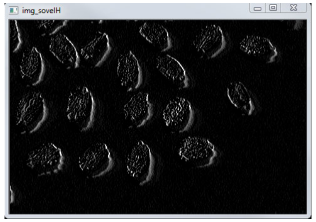
-1	0	1
-2	0	2
-1	0	1

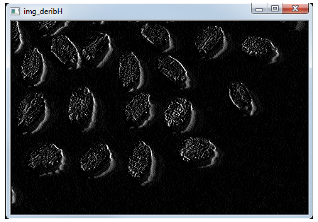
-1	-2	-1
0	0	0
1	2	1

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

0	0	0
-4	0	4
0	0	0

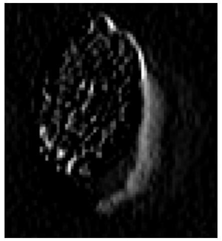
元画像

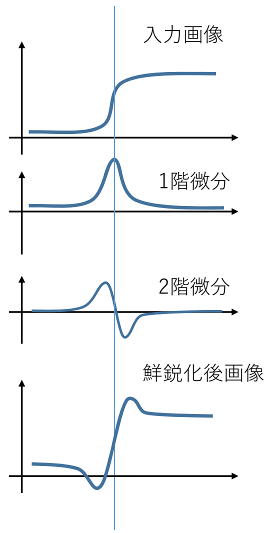




微分フィルタの正値を可視化

Sobelフィルタではノイズが
削減されているのが分かる





線形フィルタ：鮮鋭化フィルタ

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鮮鋭化フィルタが設計できる

	?	

Convolution(畳み込み)

Convolution(畳み込み)とは

二つの関数 $f(x)$ $g(x)$ を重ね合わせる演算で以下の通り定義される

連続関数

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

離散関数

$$(f * g)(n) = \sum_{k=-\infty}^{\infty} f(k)g(n-k)$$

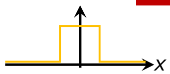
※ 関数$f(x)$ $g(x)$を入力すると、新たな関数 $f * g(x)$ が出力される

予習・復習

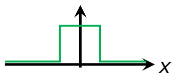
練習問題 ：
2つの関数 f g の畳み込みを求めよ

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

$$f(x) = \begin{cases} 1 & -\frac{1}{2} \le x \le \frac{1}{2} \\ 0 & otherwise \end{cases}$$



$$\mathrm{g}(x) = \begin{cases} 1 & -\frac{1}{2} \le x \le \frac{1}{2} \\ 0 & otherwise \end{cases}$$

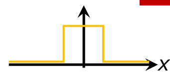


予習・復習

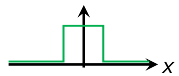
練習問題 ：
2つの関数 f g の畳み込みを求めよ

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

$$f(x) = \begin{cases} 1 & -\frac{1}{2} \le x \le \frac{1}{2} \\ 0 & otherwise \end{cases}$$



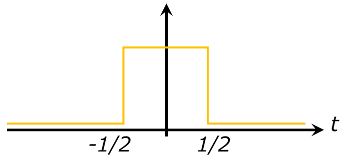
$$\mathrm{g}(x) = \begin{cases} 1 & -\frac{1}{2} \le x \le \frac{1}{2} \\ 0 & otherwise \end{cases}$$



2つの関数 f(x)とg(x)を畳の込みの式に入れるため変形すると

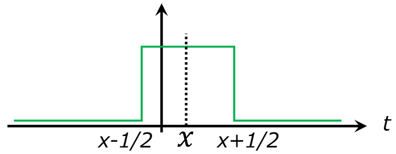
$$f(t) = \begin{cases} 1 & -\frac{1}{2} \le t \le \frac{1}{2} \\ 0 & otherwise \end{cases}$$

※ t の関数



$$\mathrm{g}(x-t) = \begin{cases} 1 & x-\frac{1}{2} \le t \le x+\frac{1}{2} \\ 0 & otherwise \end{cases}$$

※ t の関数、xは定数として扱う



非線形フィルタ

準備：平均と分散

N個の実数値の集合 $\{x_i | i = 1, \dots, N\}$ 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

練習1）以下の集合の平均と分散を求めよ

{3,0,3,5,4,3,5,1}

練習2）以下の集合AとBのうち分散が大きい方を求めよ

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（1/4）

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

平滑化フィルタでは，着目画素の周囲の画素の平均を出力した

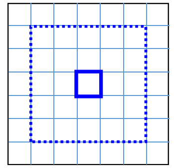
※これだと境界（エッジ）もボケてしまう

※境界（エッジ）保存しつつエッジをぼかしたい

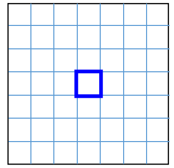
エッジ保存平滑化フィルタ（2/4）

エッジ（境界）を保持したまま、画像を平滑化するフィルタ

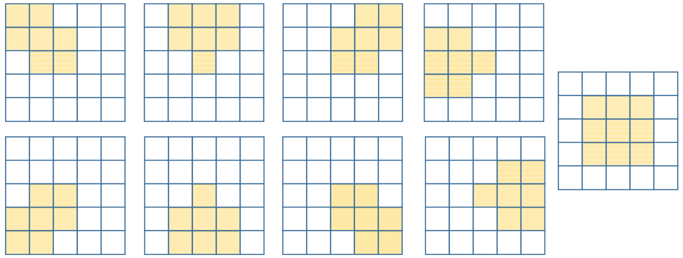
- **注目画素を中心とし以下9種の領域の分散を計算**
- **分散の最も小さな領域の平均をその画素値とする**



入力画像



出力画像



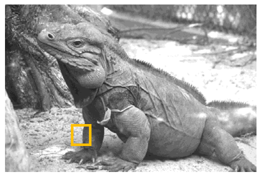
エッジ保存平滑化フィルタ（3/4）

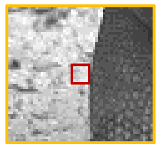
入力画像

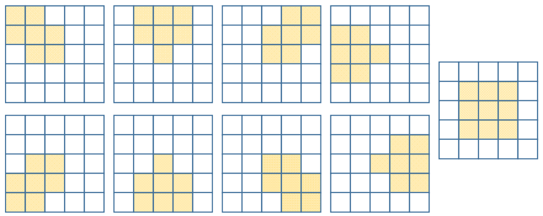
平滑化フィルタ

エッジ保存平滑化フィルタ

エッジ保存平滑化フィルタ（4/4）







エッジが保存される理由

- 左図の画素にフィルタを適用する場合を考える

  ※対象画素は白い領域に属す
- 9領域それぞれの分散を計算する
  - 白領域と黒領域にまたがる領域の分散は大きくなる
  - 白領域のみに含まれる領域の分散は小さくなる

→ 境界をまたがない領域形状が選ばれその平均が出力される

中央値 (Median)フィルタ (1/2)

中央値 (median)とは…

- 数値の集合の代表値の一つ
- 数値の集合を小さい順に並べ、ちょうど中央に位置する値

※ 要素数が偶数のときは、真ん中２つの平均が中央値

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

中央値と平均値は，用途によって使い分ける

『ある世代の年収』のような指標では、平均は外れ値の影響を大きく受けやすいため、中央値を示す

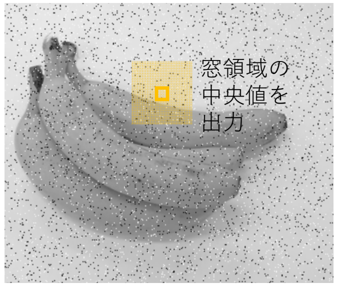
中央値 (Median)フィルタ (2/2)

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

- 中央値を利用したフィルタ
- 注目画素を中心とする 幅hの窓領域を考え、窓内の中央値を出力する

利点- 外れ値（スパイクノイズ）を除去出来る

利点- 境界（エッジ）をある程度保存する



Salt &pepper noise image


Gaussian filter


Median filter

バイラテラルフィルタ

最も有名なエッジ保存平滑化フィルタの一種

画像中の領域境界(強いエッジ)をまたがずに平滑化

(Gaussian filter) **元画像** **(bilateral filter)**

写真は Shin Yoshizawa氏により提供されたもの

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

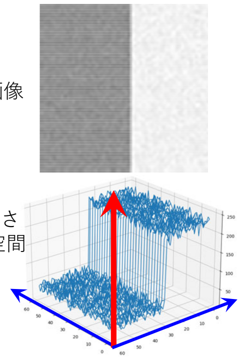
領域内部へのブラー効果により顔の”あら”が消える

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の考え方

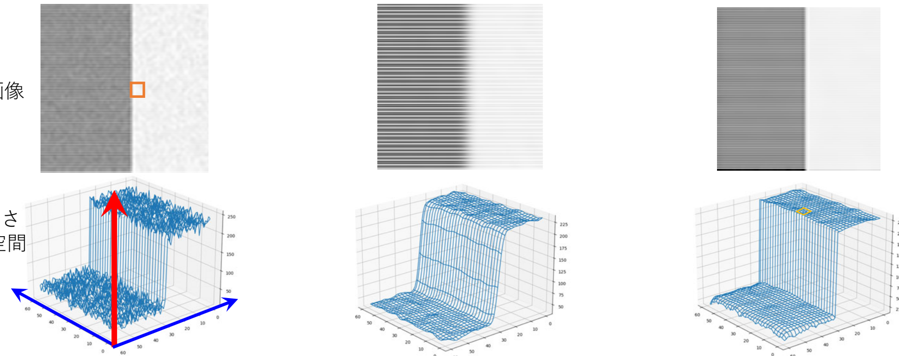
空間的距離だけでなく**画素値の距離も利用**して重み計算



Bilateral空間
+ 位置空間(spatial domain)
+ 値空間(range domain)

バイラテラルフィルタの考え方

空間的距離だけでなく**画素値の距離も利用**して重み計算



Bilateral空間
+ 位置空間(spatial domain)
+ 値空間(range domain)

Gaussian filter
位置空間の距離で重み付け
（遠いほど重みを小さく）
→境界がボケる

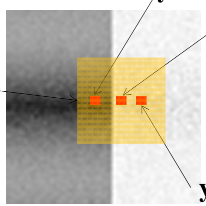
Bilateral filter
Bilateral空間の距離で重み付け
(遠いほど重みを小さく)
→境界は明瞭なまま

バイラテラルフィルタの計算法(1/3)

※ 画素位置を$\mathbf{x} \in \boldsymbol{R}^2$ とベクトルで表現する
下式の通り注目画素 **x** に対する全近傍画素 **y**の重み付き和を計算する

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

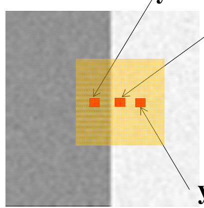
x ：注目画素位置
$W_\mathbf{x}$: **x**を中心とする近傍領域
y ：近傍領域内の画素位置
$h(\mathbf{x},\mathbf{y})$: 注目画素**x**に対する**y**の重み関数



バイラテラルフィルタの計算法(2/3)

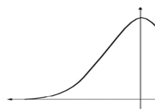
$$I_{new}(\mathbf{x}) = \frac{\sum_{\mathbf{y} \in W_\mathbf{x}} h(\mathbf{x},\mathbf{y})\, I(\mathbf{y})}{\sum_{\mathbf{y} \in W_\mathbf{x}} h(\mathbf{x},\mathbf{y})}$$

x ：注目画素位置
$W_\mathbf{x}$: **x**を中心とする局所窓
y ：局所窓内の画素位置
$h(\mathbf{x},\mathbf{y})$: 注目画素xに対するyの重み関数



重みを下記の通り定義してみる

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



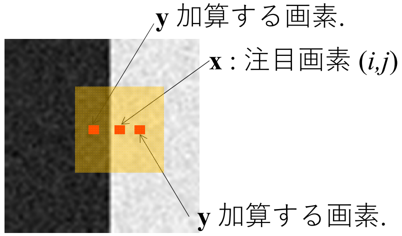
注目画素**x**と**y**が近いほど重みが大きくなる
この重みはGaussian Filterとなる

バイラテラルフィルタの計算法(3/3)

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

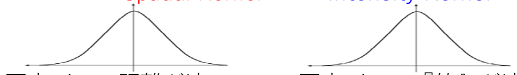
$\mathbf{x}$ ：注目画素位置
$W_\mathbf{x}$: $\mathbf{x}$を中心とする局所窓
$\mathbf{y}$ ：局所窓内の画素位置
$h(\mathbf{x},\mathbf{y})$: 注目画素xに対するyの重み関数

G_s/G_hは標準偏差s/hのガウス関数



Bilateral filterでは以下の重みを利用する

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



画素xとyの距離が遠いほど重みが小さくなる

画素xとyの『値』が遠いほど重みが小さくなる

→ 距離が近く、値も近い画素に大きい重みを付与

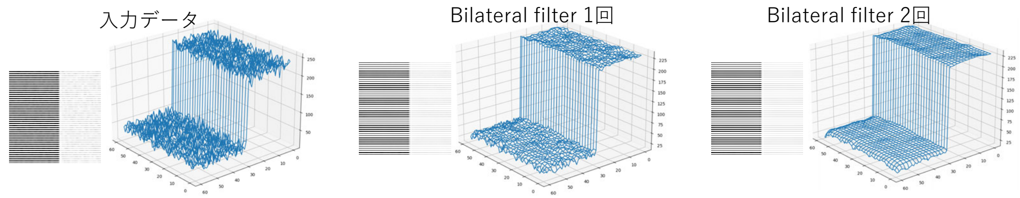
→ 画素ごとに重み係数が変化するので線形フィルタでない

バイラテラルフィルタのパラメタ

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

- 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、以下を値の距離を用いてIntensity Kernelを計算すると良い

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



まとめ：非線形空間フィルタ

- 境界（エッジ）を保存する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が、特殊な効果が得られる

写真は Shin Yoshizawa氏により提供されたもの

練習問題

1. 中央値フィルタがスパイクノイズを除去する効果を持つ理由を簡潔に説明せよ(20文字以内)

2. 講義中に解説したエッジ保存平滑化フィルタにおいて、エッジが保存される理由を簡潔に説明せよ（30文字以内）

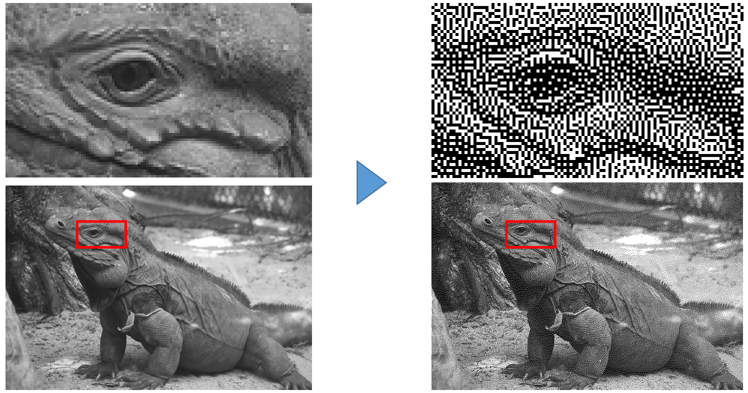
3. バイラテラルフィルタに関する説明として正しいものを選択せよ
   - 画像全体を明るくする効果を持つ
   - 画像全体を暗くする効果を持つ
   - 画像のエッジを保存する効果を持つ
   - 画像のエッジをまたがずに平滑化する効果を持つ
   - 対象画素の周囲情報を利用する空間フィルタの一種である
   - 対象画素の周囲の重み付き和を計算する線形フィルタの一種である
   - 対象画素毎に重み係数が変化する
   - 対象画素毎に重み係数は変化しない

ハーフトーン処理

ハーフトーン処理

白または黒の画素のみを用いて、中間色（グレー）を表現する手法
白い画素・黒い画素の密度により濃淡を表現する
→画素が十分細かければ人の目に濃淡が認識される

グレースケール画像

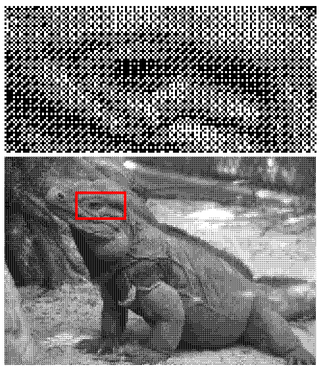


白い画素・黒い画素の密度で中間色を表現

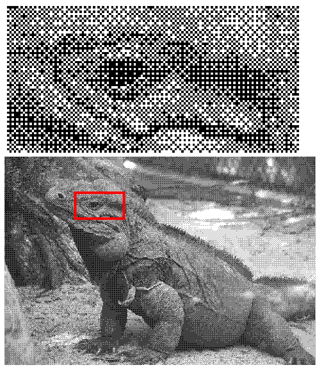
白黒 2値画像

ハーフトーン処理

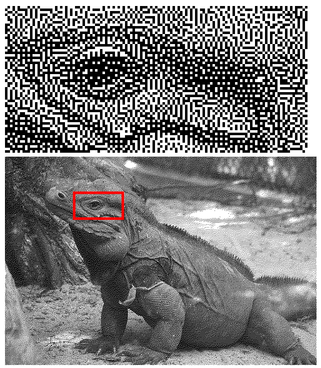
ここではグレースケール画像にハーフトーン処理を施す３種のアルゴリズムを紹介する



濃度パターン法



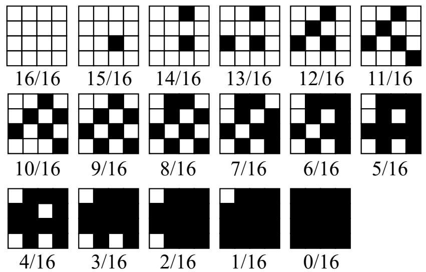
ディザ法



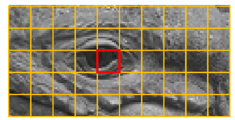
誤差拡散

濃度パターン法

1. 白画素の数が異なるサイズ4x4のパターンを17個用意

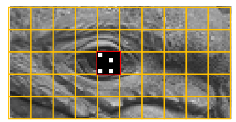


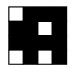
2. 入力画像を4 x 4のブロックに分割
3. 各ブロックの平均輝度値を計算
4. 各ブロックについて近い平均輝度値をもつパターンを選択し，置き換える



4x4のブロック

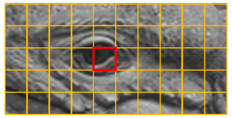
平均画素値 : 73.0
[0,1]に正規化: 0.286
4/17~5/17の範囲なのでパターン4を採用する





欠点 : 繰り返しパターンが目立つ

ディザ法



1. 4x4のディザパターンを用意
※各マスに0~15の整数値が書かれる

2. 入力画像を4x4のブロックに分割
※画素値を[0,255]から[0,16]に変更

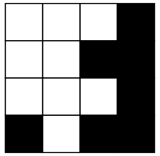
3. 各ブロックの各画素をディザパターンと比較
ディザパターンの値以上　　　→白
ディザパターンの値より小さい→黒

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

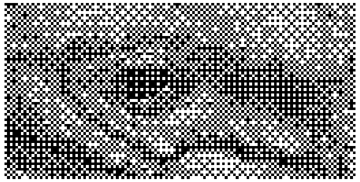
ディザパターン

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

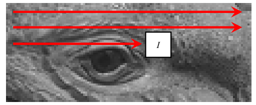


出力パターン



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画素を訪問し一画素ずつ2値化する

注目画素の画素値が I のとき…

STEP 1 : 二値化処理

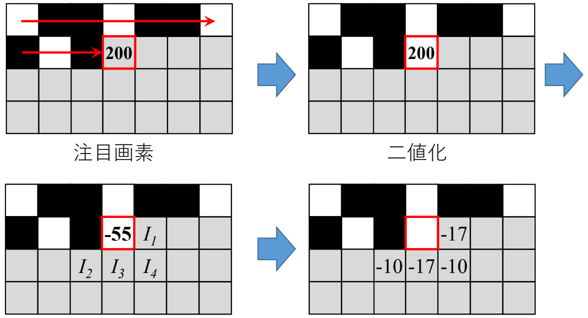
I > 127 → 注目画素を白に
I ≦ 127 → 注目画素を黒に

STEP2 : 誤差拡散

上の二値化で以下の誤差eが発生した
I > 127 → e = I - 255
I ≦ 127→ 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)

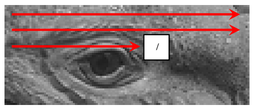
$$I_1 \leftarrow I_1 + \frac{5}{16}e,\ I_2 \leftarrow I_2 + \frac{3}{16}e,$$

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



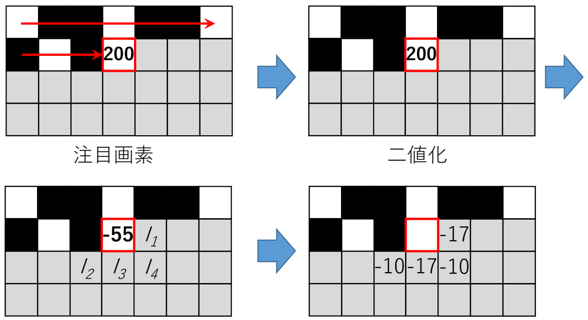
注目画素　二値化
誤差拡散する隣接画素　誤差拡散し画素値を変更

誤差拡散法



- 実装時の問題：右端や下端では誤差を拡散させる画素がない

→ 解決策：右端や下端の計算時，誤差を拡散させる画素がない場合には誤差拡散を行わない



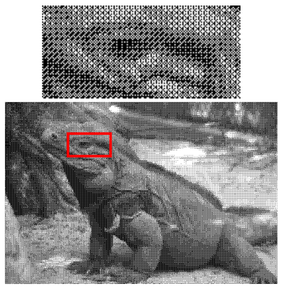
注目画素　二値化
誤差拡散する隣接画素　誤差拡散し画素値を変更

まとめ：ハーフトーン処理

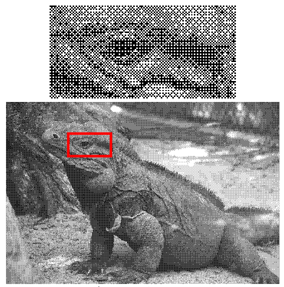
グレースケール画像を白黒2値画像で表現する手法
白黒画素の密度を利用して中間色を表現する

濃度パターン法：ブロックの輝度値を利用し濃度パターンで置き換える
ディザ法　　　：ディザパターンと画素値を比較し二値化
誤差拡散法　　：ラスタスキャン順に二値化し，発生した誤差を隣接画素に拡散する

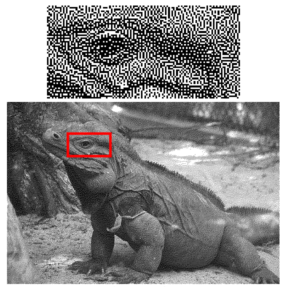
- プログラミング演習で実装します



濃度パターン法



ディザ法



誤差拡散