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，ハーフトーニング

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

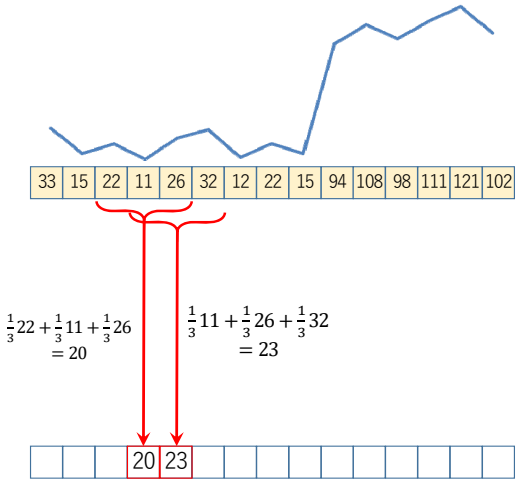
Contents

- 線形フィルタの復習
- 非線形フィルタ
- ハーフトーニング

線形フィルタの復習

復習

線形フィルタの例 1D



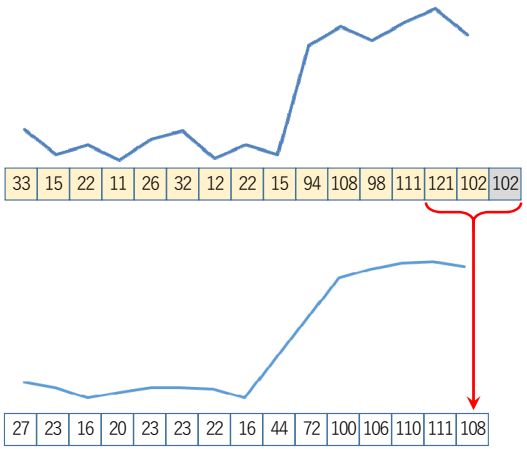
平滑化したい!

1/3	1/3	1/3

周囲 3 ピクセルの平均を取る

復習

線形フィルタの例 1D



※端ははみ出すので値をコピー（ほかの方法もある）

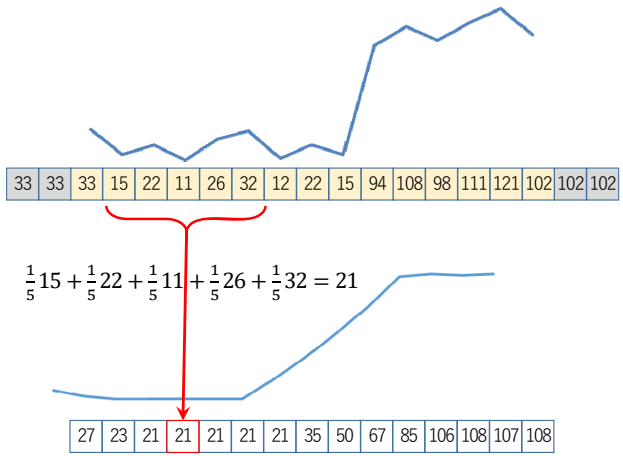
平滑化したい!

1/3	1/3	1/3

周囲３ピクセル
の平均を取る

復習

線形フィルタの例 1D



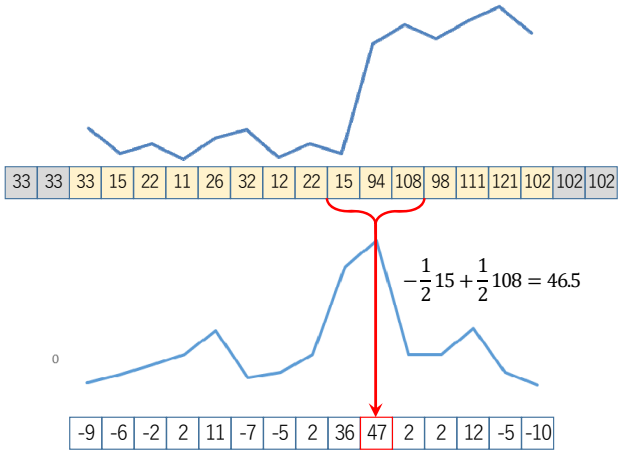
もっと
平滑化したい!

1/5	1/5	1/5	1/5	1/5

周囲5ピクセル
の平均を取る

復習

線形フィルタの例 1D



※端ははみ出すので値をコピー（ほかの方法もある）

エッジ
(変化の大きい部分)
を検出したい

-0.5	0	0.5

右と左のピクセルの
差をとる

予習・復習

Convolution(畳み込み)とは

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

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

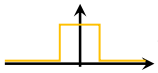
離散関数 $$(f*g)(n)=\sum_{k=-\infty}^{\infty}f(k)g(n-k)$$

$f(t)$ を固定し，$g(t)$ を平行移動しながら$f(t)$に掛けあわせ，得られた関数を積分するとみてもよいかも

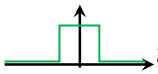
予習・復習

例題)
2つの関数f gの畳み込みを求めよ

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



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

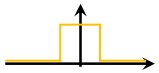


$$g(x)=\begin{cases}1 & -\frac{1}{2}\le x\le\frac{1}{2}\\ 0 & otherwise\end{cases}$$

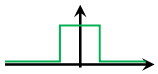
予習・復習

例題)
2つの関数f gの畳み込みを求めよ

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

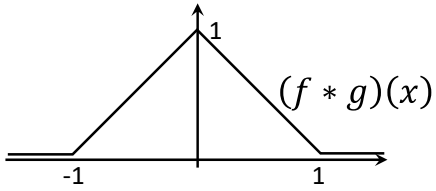


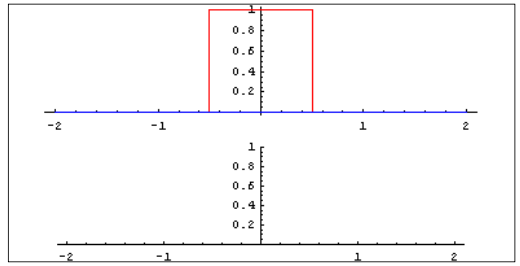
$$f(t)=\begin{cases}1 & -\frac{1}{2}\le t\le\frac{1}{2}\\ 0 & otherwise\end{cases}$$



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

$$(f*g)(x)=\begin{cases}x+1 & -1\le x\le 0\\ 1-x & 0\le x\le 1\\ 0 & otherwise\end{cases}$$





By Lautaro Carmona [CC-BY-SA] from wikipedia

線形フィルタ：平滑化

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

$$\frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

線形フィルタ :エッジ検出（**微分**）

- 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

$$\begin{bmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{bmatrix}\quad\begin{bmatrix}-1&-1&-1\\0&0&0\\1&1&1\end{bmatrix}$$

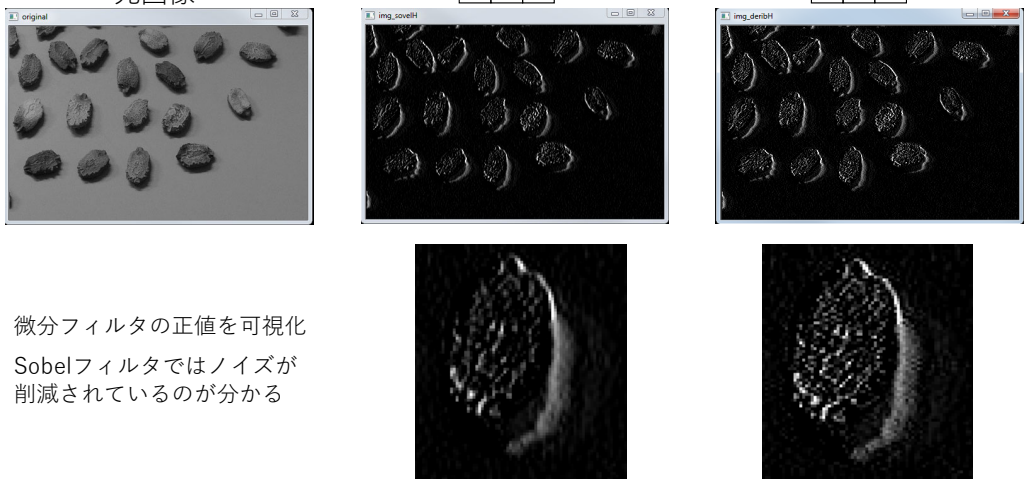
Sobel filter

$$\begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix}\quad\begin{bmatrix}-1&-2&-1\\0&0&0\\1&2&1\end{bmatrix}$$

元画像

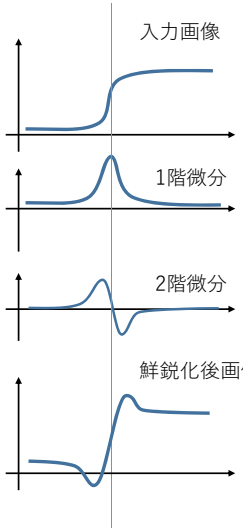
-1	0	1
-2	0	2
-1	0	1

0	0	0
-4	0	4
0	0	0



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

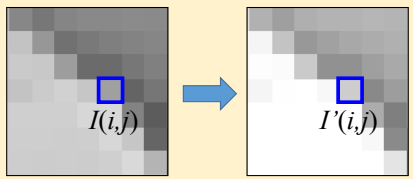
	?	

復習

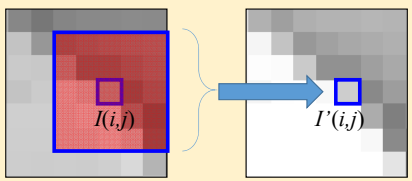
空間フィルタとは

- 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:
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

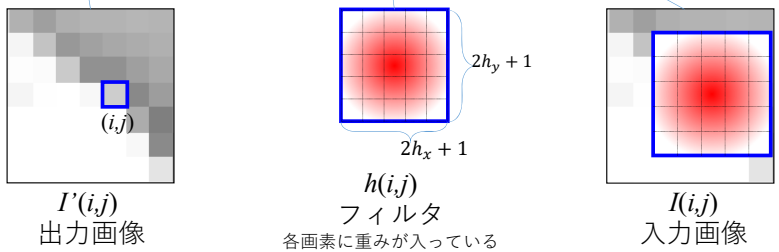


復習

線形フィルタとは

出力画素値を，入力画像の周囲画素の重み付和で計算する

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



$I'(i,j)$
出力画像

$h(i,j)$
フィルタ
各画素に重みが入っている

$I(i,j)$
入力画像

非線形フィルタ

準備：平均と分散

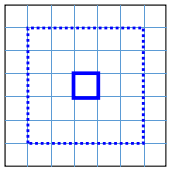
実数値の集合 $\{x_i | i = 1, \ldots, N\}$ が与えられたとき、

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

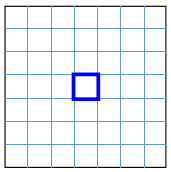
1. 以下の集合の平均と分散を求めよ
   {3,0,3,5,4,3,5,1}

2. 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

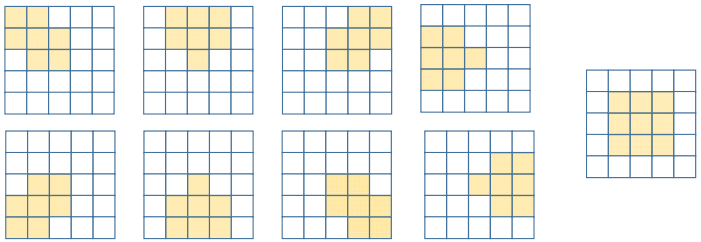


出力画像

平滑化フィルタでは，
画素(i, j)を計算するため周囲の
画素の平均を計算した

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

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



入力
画像

平滑化フィルタ

エッジ保存平滑化フィルタ

中央値フィルタ(Median filter)

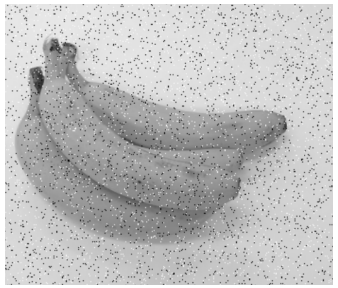
• 中央値 (median)とは…
数値の集合の代表値
数値の小さい順に並べ、ちょうど中央に位置する値

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

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 filter


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)

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

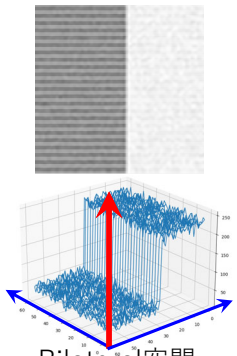

Bi-Lateral Filer
Spatial radi:5
Range radi:80

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

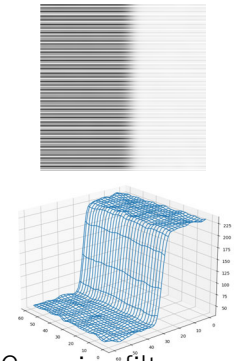
画像は [CG-Arts協会 ディジタル画像処理 図5.37]を
参考に井尻が再作成したもの

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

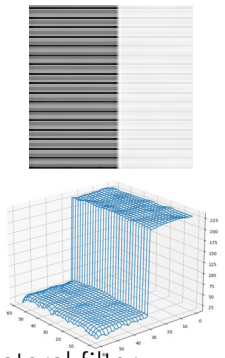


入力画像

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)

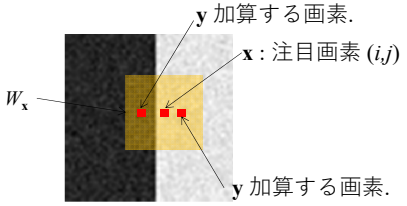


Bilateral filter
Bilateral空間の距離で重み付け
(遠いほど重みを小さく)

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



※『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 :

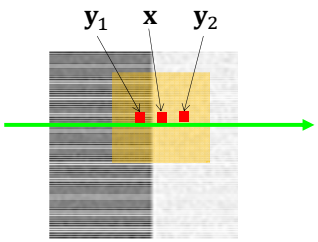
$$h(\mathbf{x},\mathbf{y}) = \underbrace{G_s(|\mathbf{x}-\mathbf{y}|)}_{\text{Spatial Kernel}} \cdot \underbrace{G_h(|I(\mathbf{x})-I(\mathbf{y})|)}_{\text{Intensity Kernel}}$$

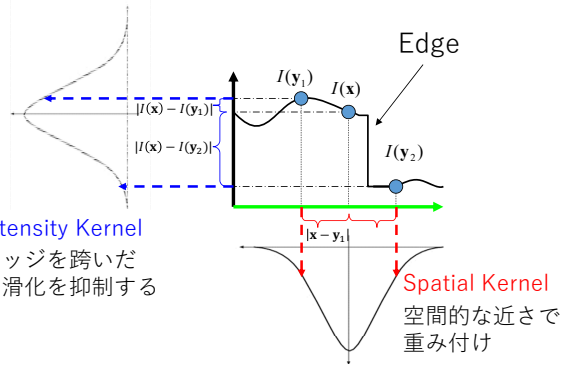
G_σは標準偏差σのガウス関数

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　$\mathbf{y} = (i+m, j+n)$

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





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

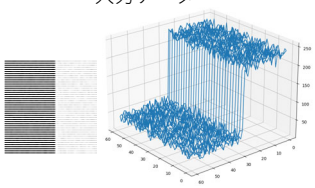
バイラテラルフィルタ（パラメタ）

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

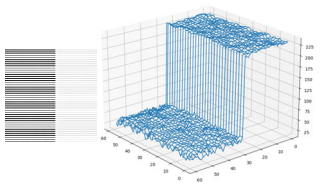
パラメータh: 平滑化したい領域の輝度値の標準偏差の 0.5-2.0倍程度をよく用いる
複数回適用すると良い結果が出やすい
カラー画像はチャンネル毎でなく、以下を用いて同じ重みを利用するとよい

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

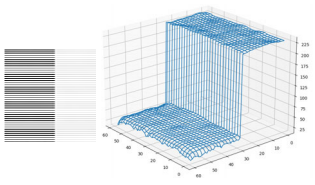
入力データ



Bilateral filter 1回



Bilateral filter 2回



まとめ：空間フィルタ（非線形）

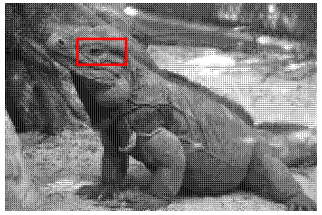
- エッジ保存効果のあるフィルタを紹介した
  - エッジ保存平滑化
  - メディアンフィルタ
  - バイラテラルフィルタ
- 線形フィルタと比べ計算量は大きいが、特殊な効果が得られる

写真は Shin Yoshizawa氏により提供されたもの

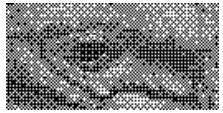
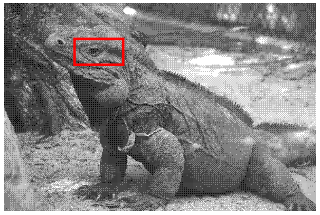
ハーフトーン処理

ハーフトーン処理

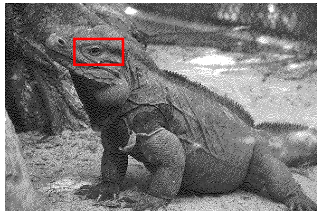

- グレースケール画像を白黒2値画像に変換する
  - 白黒画素の密度により濃淡を表現する
  - 画素が十分細かければ人の目に濃淡として認識される

濃度パターン法

ディザ法

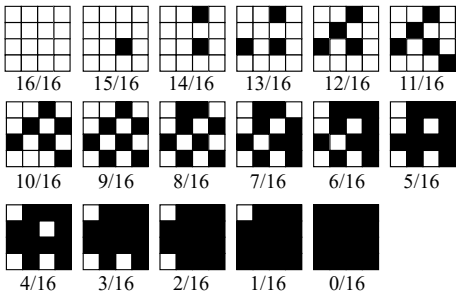



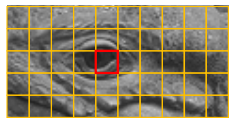
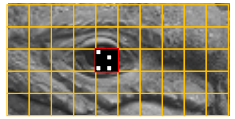
誤差拡散

濃度パターン法

1. 元画像を4 x 4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

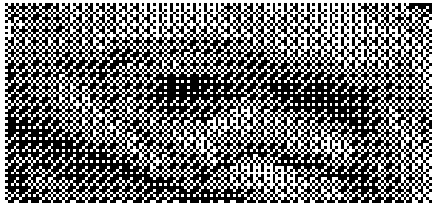
※ブロックのサイズは変更可（今回は4x4）




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

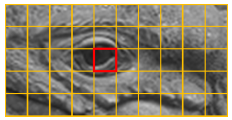


欠点 : 繰り返しパターンが目立つ

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

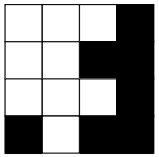


0	8	2	10
12	4	14	6
3	11	1	9
15	7	13	5

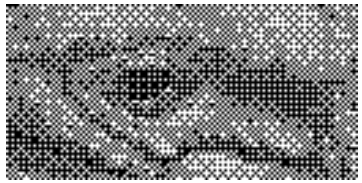
ディザパターン

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

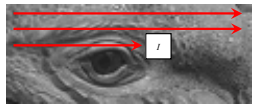


出力パターン



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 → 注目画素を白に

I ≦ 127 → 注目画素を黒に

2. 誤差拡散

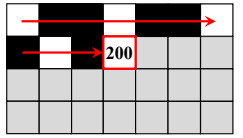
上の二値化で以下の誤差eが発生した

I > 127 → $e = I - 255$

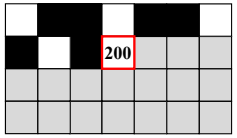
I ≦ 127 → $e = I - 0$

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)

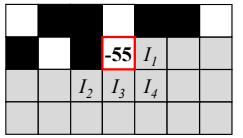
$$I_1 \leftarrow I_1 + \frac{5}{16}e,\ I_2 \leftarrow I_2 + \frac{3}{16}e,$$
$$I_3 \leftarrow I_3 + \frac{5}{16}e,\ I_4 \leftarrow I_4 + \frac{3}{16}e$$



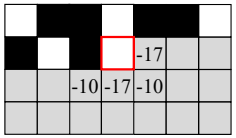
注目画素



二値化

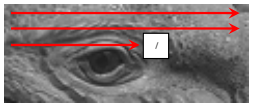


誤差拡散する隣接画素



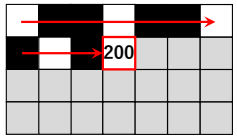
誤差拡散し画素値を変更

誤差拡散法

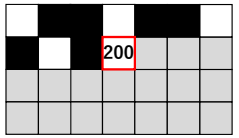


- 実装時の問題：右端や下端では誤差を拡散させる画素がない
- 解決策：右端や下端の計算時，誤差を拡散させる画素がない場合には誤差拡散を行わない

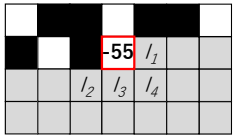
※ 小テストにて、誤差拡散法の欠点を問う課題を出しています。この右端・左端にておきる問題は、回避可能なのでこれ以外の欠点を回答してください。



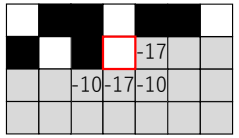
注目画素



二値化



誤差拡散する隣接画素



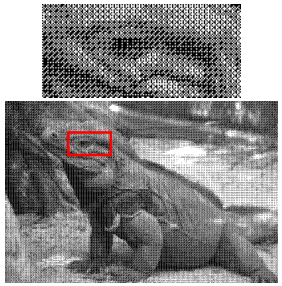
誤差拡散し画素値を変更

まとめ：ハーフトーン処理

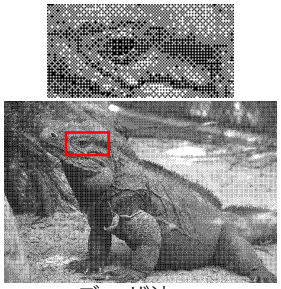
- グレースケール画像を白黒2値画像で表現する手法

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

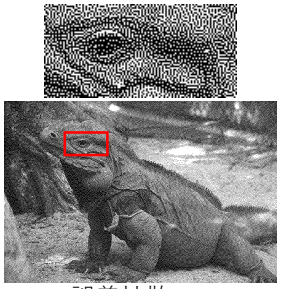
- プログラミング演習で実装します



濃度パターン法



ディザ法



誤差拡散

時間が余ったら、フーリエ級数の話をする