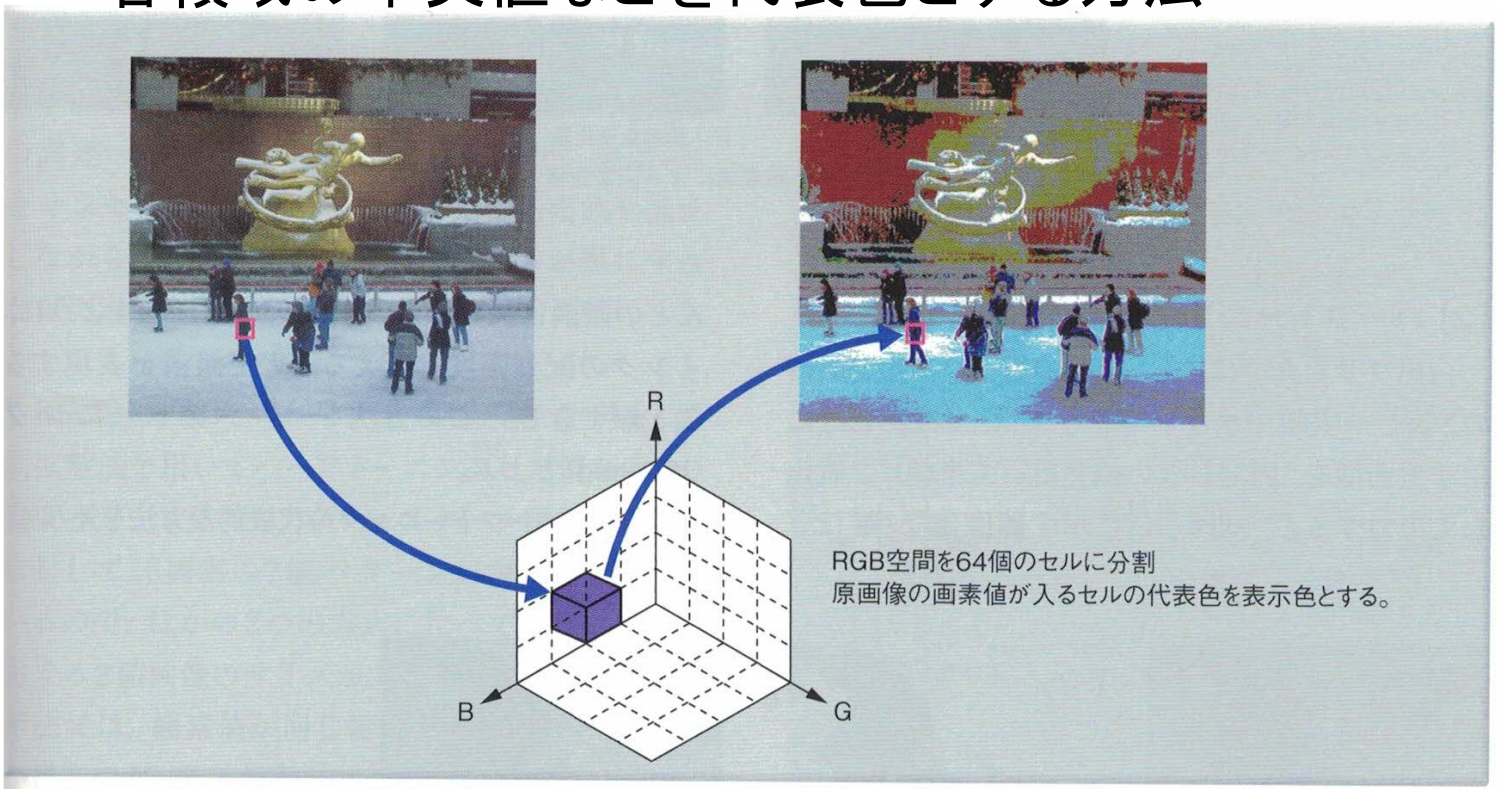
# 濃淡画像処理:
# 画像表示のための処理

# 限定色表示

- カラー画像をディスプレイで表示するときに，ディスプレイによっては表現できる色数が画像が持つ色数よりも少ないことがある
- このような場合，画像の色数を限定して出力する**限定色表示**が用いられる
  - **均等量子化法**
  - **頻度法**
  - **メディアンカット量子化法**

# 均等量子化法

- 入力画像の色空間を**均等に分割**し，
  各領域の中央値などを代表色とする方法

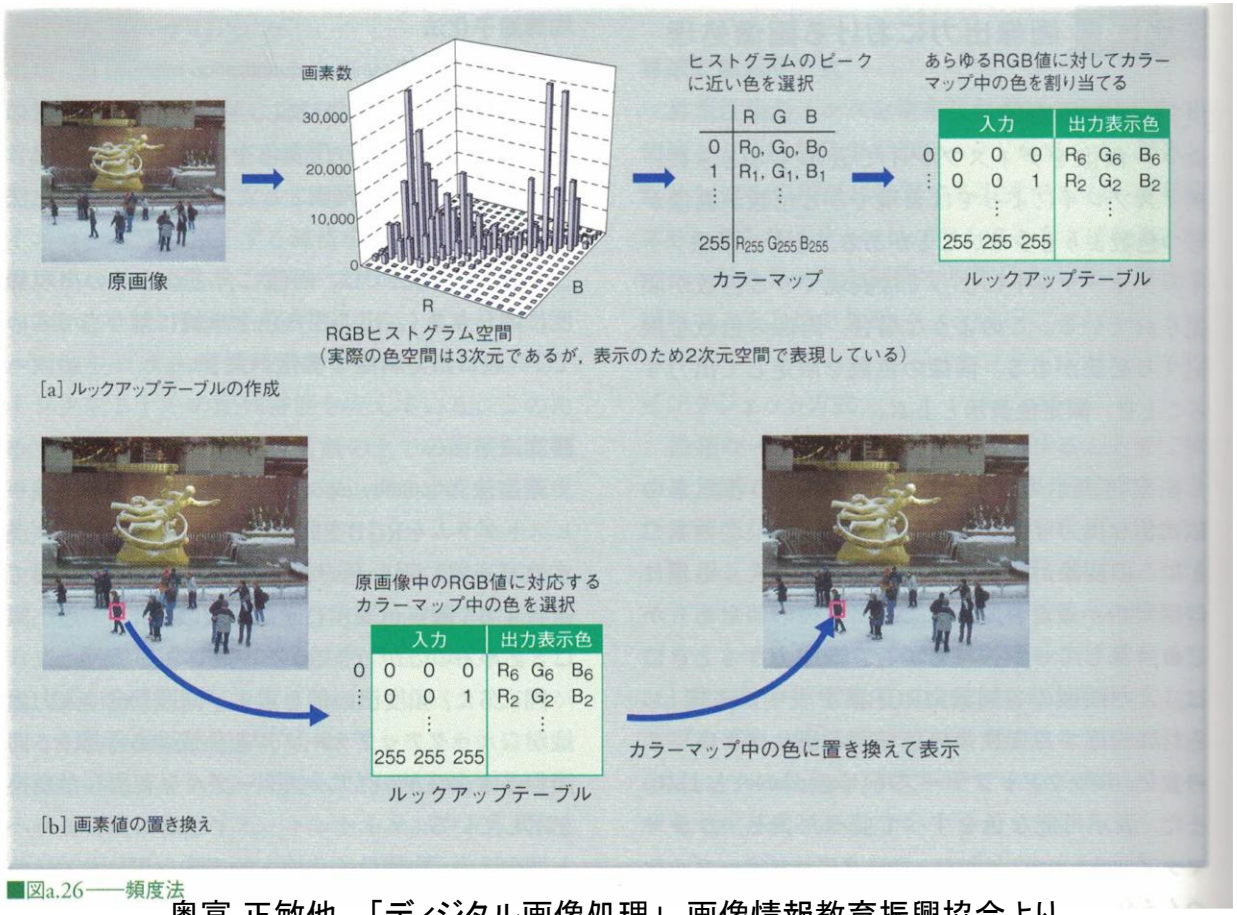



■図a.25——RGB各4階調での均等量子化法

奥富 正敏他，「ディジタル画像処理」，画像情報教育振興協会より

# 頻度法

- <u>出現頻度の高い</u>色から順次代表色として選定する方法

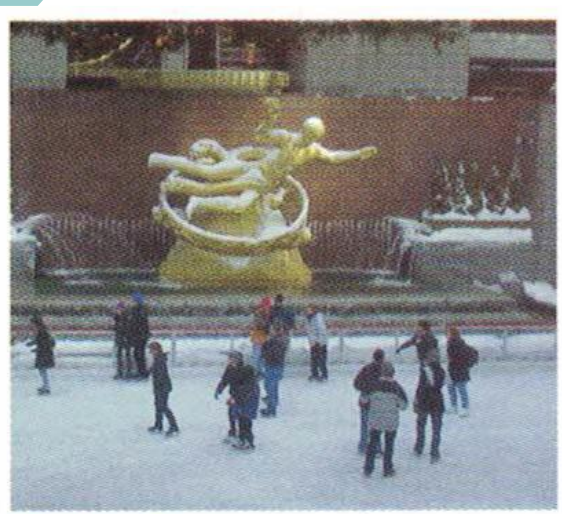



■図a.26——頻度法

奥富 正敏他，「ディジタル画像処理」，画像情報教育振興協会より

# 均等量子化法と頻度法の比較

- どちらも27色の限定色表示
- 頻度法のほうが**劣化の少ない**表示が可能



[a] 原画像

[b] 均等量子化法

[c] 頻度法

■図a.27——均等量子化法と頻度法による限定色表示の例

奥富 正敏他，「ディジタル画像処理」，画像情報教育振興協会より

# メディアンカット量子化法

- **中央値**で領域を分割していく方法
  各領域の**中央値**を代表色として選択

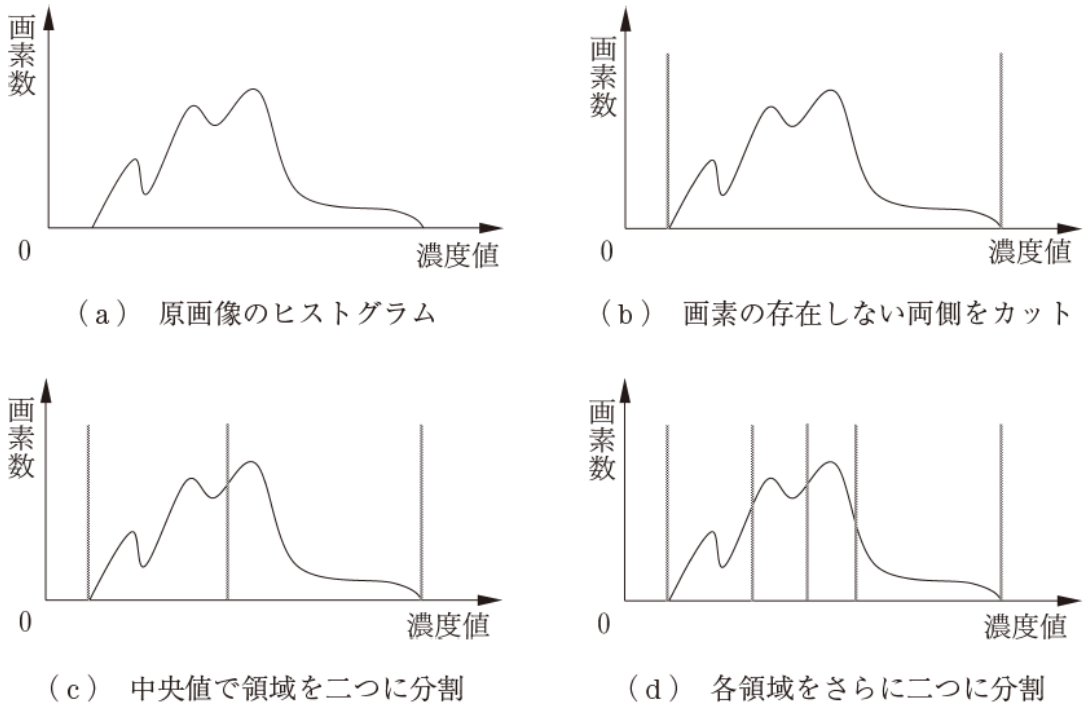
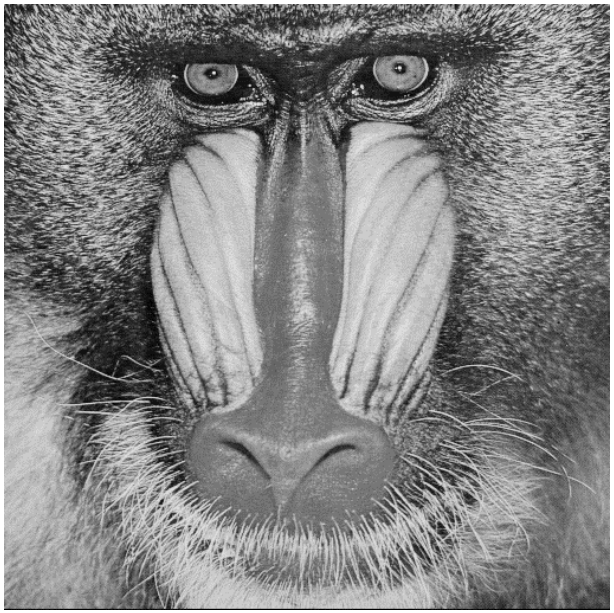



**図 5.20** メディアンカット量子化法

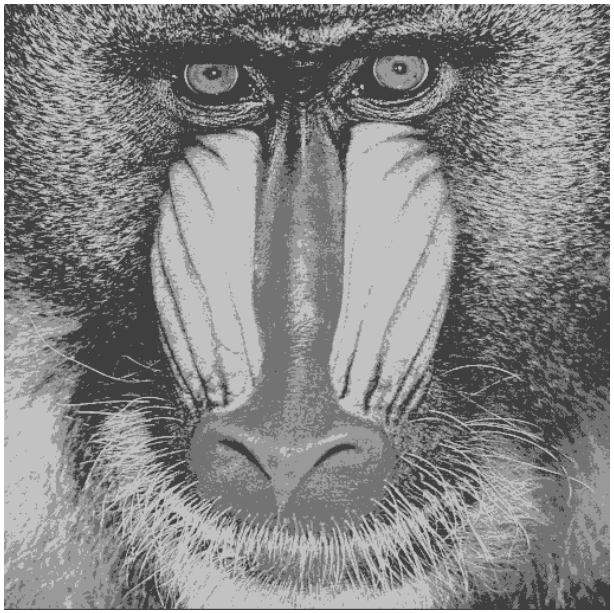
山田 宏，末松 良一 「画像処理と画像認識 ―AI時代の画像処理入門」 コロナ社より

# メディアンカット量子化法

- 256色のグレースケール画像を4色で限定色表示



256色



4色

# 2値化処理

# 2値化処理

- **高速に画像処理を行いたい**場合，多値画像を2値画像に変換してから処理を行うことがある
- 濃度画像を2値化する場合，**しきい値（閾値）**を決めて白か黒を判定する
- しきい値をいくつにするか決定するために，様々手法が提案されている
  - **モード法**
  - **微分ヒストグラム法**
  - **P-タイル法**
  - **判別分析法（大津の2値化）**

# モード法

- 画像の**濃度ヒストグラムが谷を持っている**場合に，その**谷の濃度値**をしきい値とする方法
- ノイズの多い画像や，自然画像などのようにヒストグラムに谷が生じないような画像には不向き

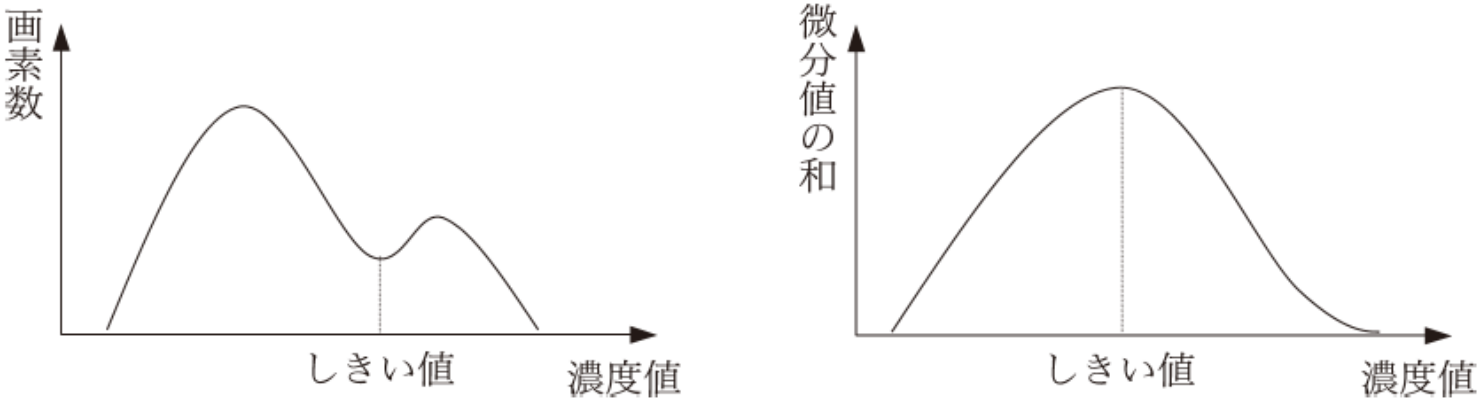



**図 6.1** 2値化処理

# 微分ヒストグラム法

- **濃度値の変化の大きいところ**が
  対象図形の輪郭である可能性が強いことを利用
- **微分ヒストグラムの値が最大値**となる濃度値を
  しきい値とする方法

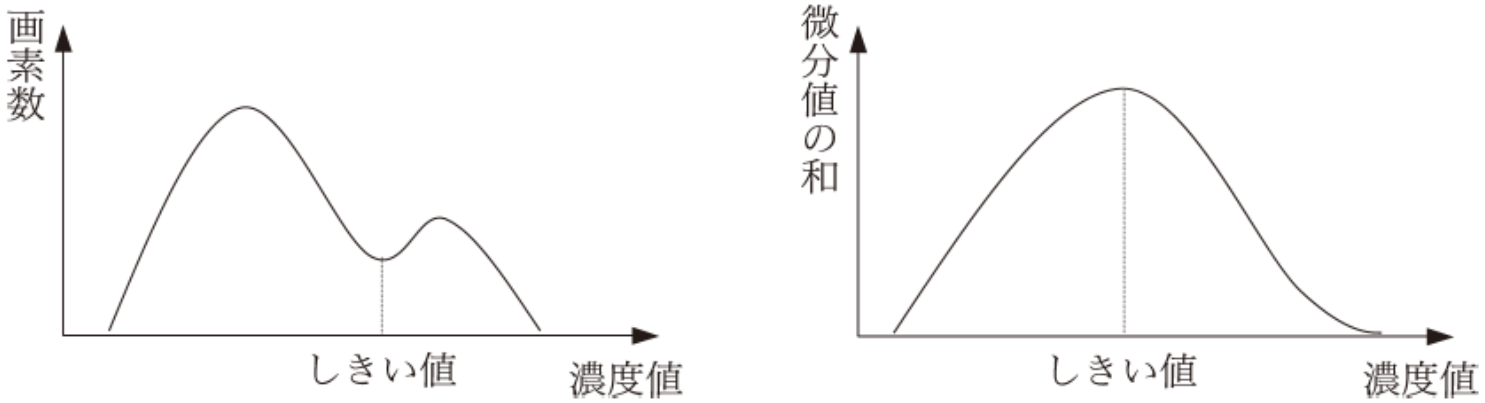
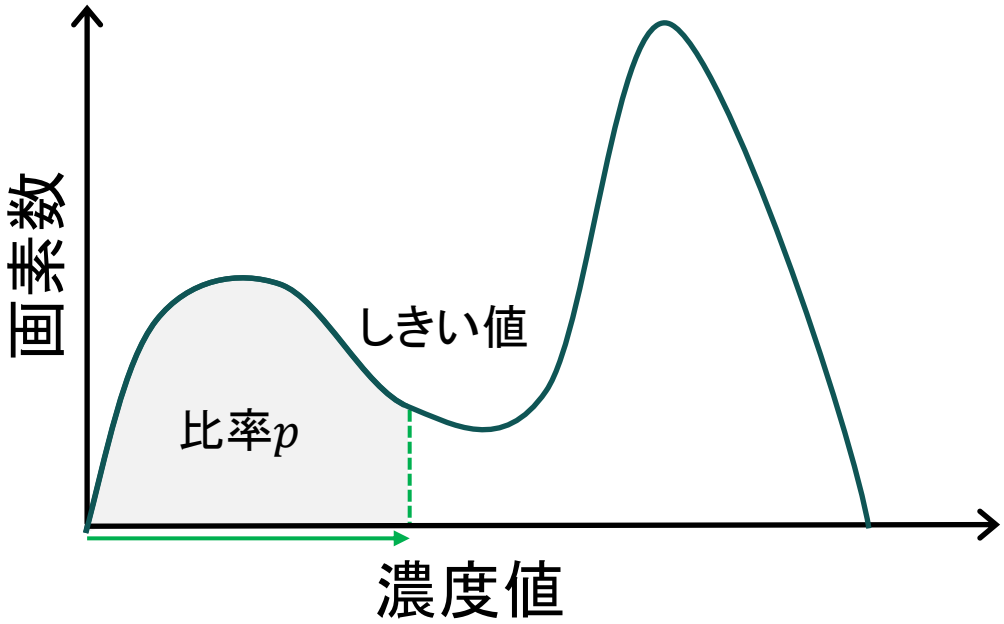



図 6.1　2 値化処理

# P-タイル法

- 画像の中から**切り出したい被写体の面積に対する比率$p$**を決める
- 濃度ヒストグラムを求め，濃度値の低い方から画素数を積算し，**比率$p$に達した時点の濃度値**をしきい値とする



# 判別分析法（大津の2値化）

- **統計的処理**により，しきい値を決定する方法
- しきい値$t$で画像を**2つのクラス**にわける
- **分散を用いたクラス同士の分離が最も良くなる**ようにしきい値$t$を決定する

各クラスの画素数を$n_1, n_2$，濃度値の平均を$\overline{f_1}, \overline{f_2}$とすると

クラス1の分散 $\sigma_1^2 = \frac{1}{n_1}\sum_{f=0}^{t-1} n_f\left(f - \overline{f_1}\right)^2$

クラス2の分散 $\sigma_2^2 = \frac{1}{n_2}\sum_{f=t}^{255} n_f\left(f - \overline{f_2}\right)^2$

$n_f$：濃度$f$を持つ画素の数

# 判別分析法（大津の2値化）

このとき，**クラス内分散**$\sigma_w^2$および**クラス間分散**$\sigma_b^2$は

クラス内分散　$\sigma_w^2 = \dfrac{n_1\sigma_1^2 + n_2\sigma_2^2}{n_1 + n_2}$

クラス間分散　$\sigma_b^2 = \dfrac{n_1\left(\overline{f_1} - \overline{f_t}\right)^2 + n_2\left(\overline{f_2} - \overline{f_t}\right)^2}{n_1 + n_2}$

ここで，$\overline{f_t}$は全画素の濃度値の平均である

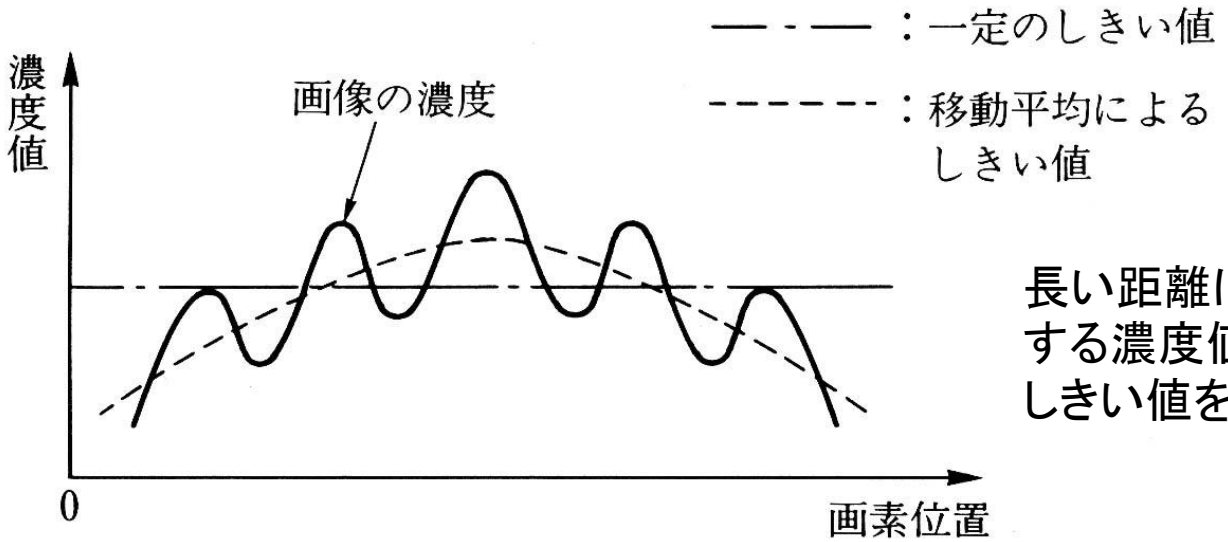
$$\overline{f_t} = \frac{n_1\overline{f_1} + n_2\overline{f_2}}{n_1 + n_2}$$

クラス内分散$\sigma_w^2$が小さく，クラス間分散$\sigma_b^2$が大きいほど両者を良好に分離できるため，

分散比 $\dfrac{\sigma_b^2}{\sigma_w^2}$ が**最大**となるようにしきい値$t$を求める

# 可変しきい値法

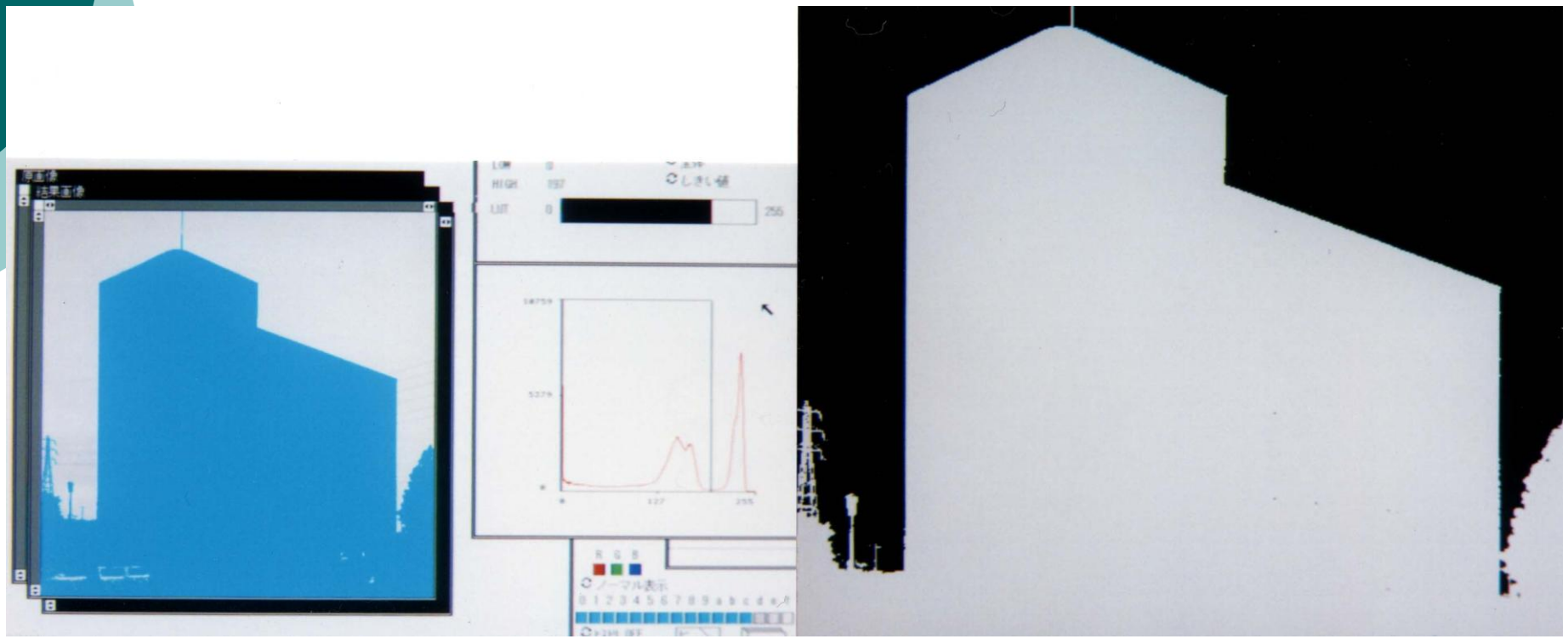
- 照明のあたり方などの問題で，**一定のしきい値**では良好な2値画像が得られない場合がある
- 画像を適当な大きさの小領域に分割し，小領域ごとにしきい値を決定する方法が有効である
- 移動平均によりしきい値を決定する方法もある（下図）




長い距離に対して変化する濃度値に従って，しきい値を変化させる．

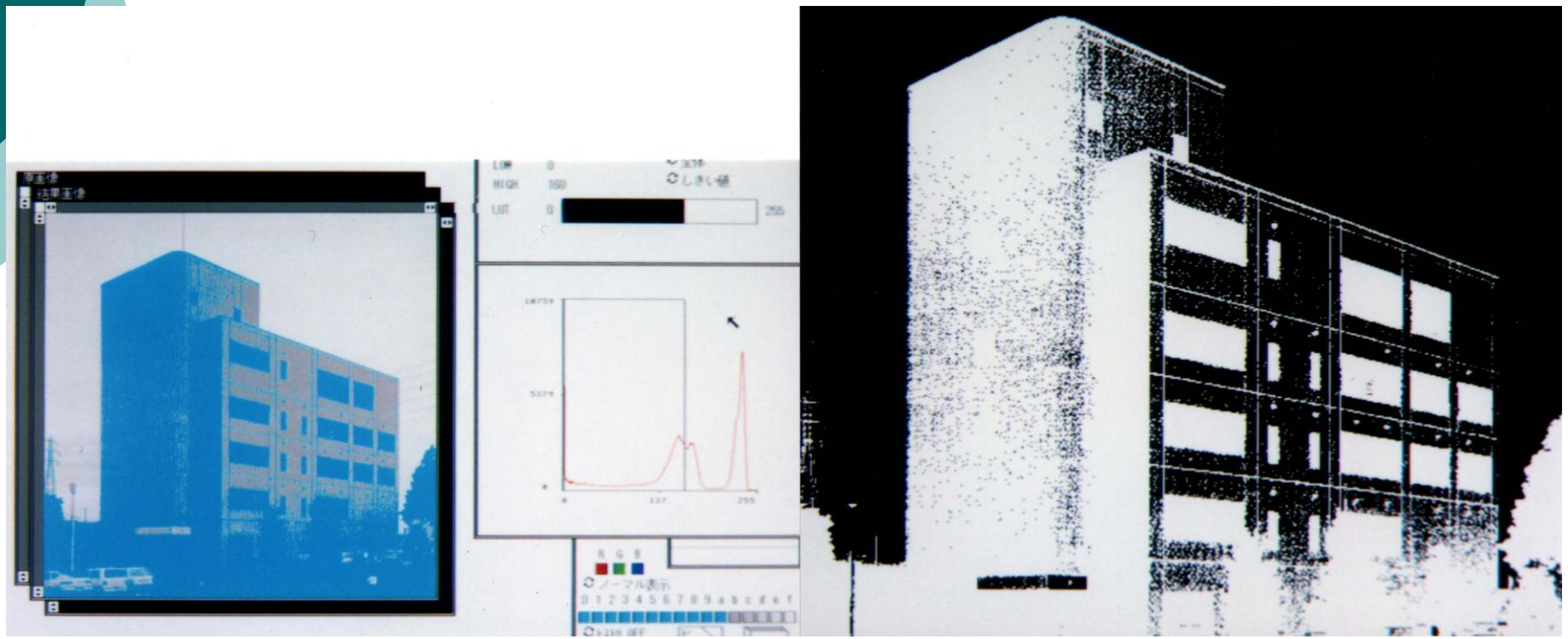
# 入力画像

# 2値化 （レベル1）



濃度ヒストグラムをとった結果，複数の分布の谷があり，それぞれの谷の濃度値をしきい値として2値化を行なった例である.
最も高い位置での濃度値の谷をしきい値として2値化した結果，空の部分と建物を含むそれ以外の部分が分離された.

# 2値化 （レベル2）



上から二番目の濃度値の谷をしきい値として2値化した結果，
建物の左側面と右側の壁面部分が分離された．

# 2値化 （レベル3）

一番小さい濃度値の長い谷の右の部分をしきい値として2値化した結果，
建物の窓および木々の部分が分離された．

# 原画像，輝度反転，2値化（固定しきい値，可変しきい値）

原画像

輝度反転
（ネガ）

固定しきい値

4x4の領域分割による
可変しきい値

# 特殊な2値化処理：疑似濃淡表示

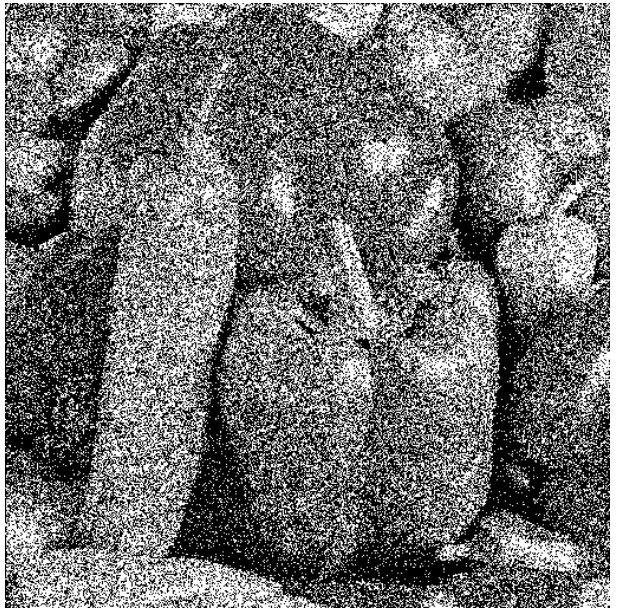
# 疑似濃淡表示(ディザ法)

- 2値化におけるしきい値を画素ごとに変化
  白と黒で疑似的に濃淡画像を再現する方法
- プリンタなどの印刷技術で使用される
- しきい値の決め方によって手法は様々
  - **ランダムディザ法**
  - **組織的ディザ法**

# ランダムディザ法

- **各画素に対するしきい値をランダムに与える**
- 実装は簡単だが,
  雑音が入ったざらついた画像になりやすい

原画像



ランダムディザ法

# 組織的ディザ法

- ディザパターン（ディザマトリックス）を用意する
- ディザパターンの値と画像の画素値を比較して2値化

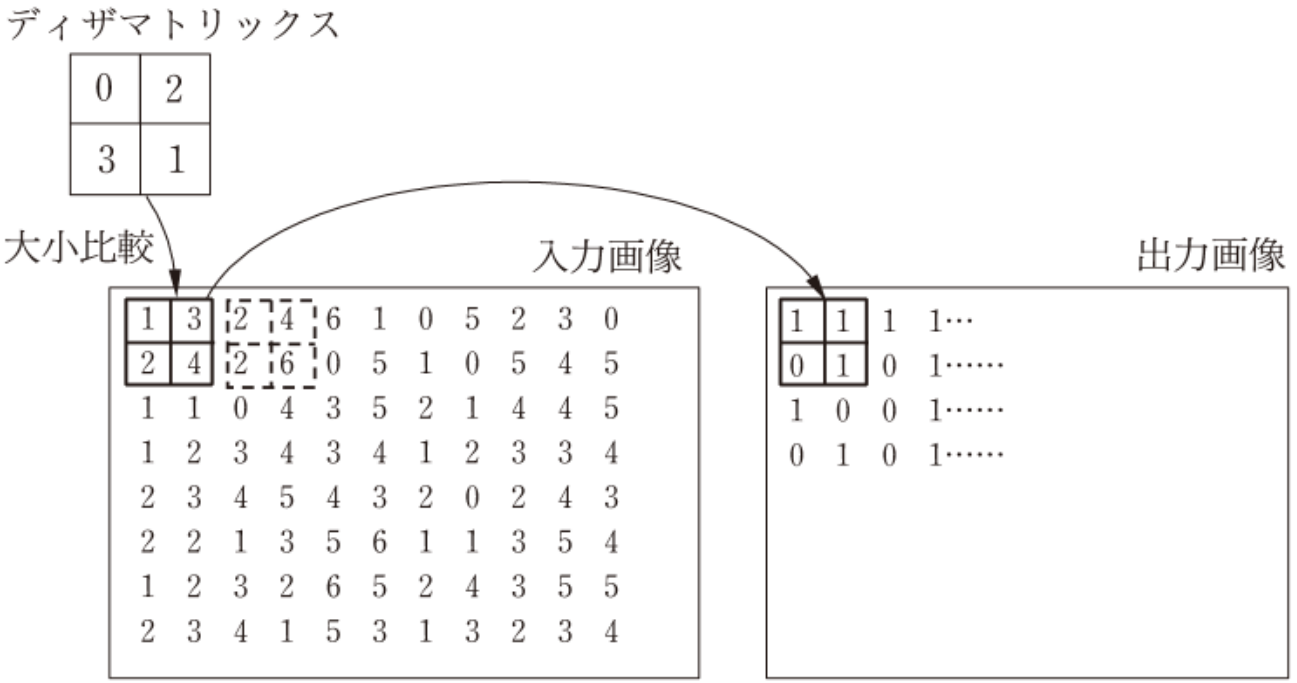


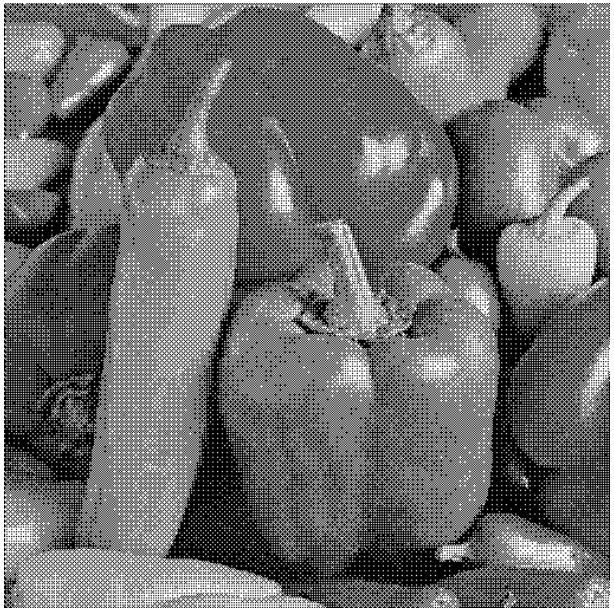

**図 5.23**　組織的ディザ法の処理手順

# 組織的ディザ法

- ４×４画素のディザマトリックスを用いて2値化
- 周期的な偽パターンが発生しやすい

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

Bayer型のディザパターン



組織的ディザ法

# 組織的ディザ法

- ディザパターンとしてはBayer型の他にも，渦巻型や網点型などがある
- 入力グレースケール画像の階調数に合わせてスケーリングを行って使用する

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

渦巻型

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

網点型

# 画像処理プログラミング４
# 2値化処理

# 手動でしきい値を設定して2値化

- thresholdに指定した値をしきい値とする
- cv2.threshold()関数で画像を2値化できる
  戻り値の第1変数はここでは使わないので'_'を記載

```
# 画像を読み込む
gray_img= cv2.imread(common_path+ 'building.png',
cv2.IMREAD_GRAYSCALE)

# しきい値を手動で設定して2値化
threshold = 127
_, binary_img= cv2.threshold(gray_img, threshold, 255,
cv2.THRESH_BINARY)

# 実行結果の表示
show_images(gray_img, binary_img, 'Original Image', 'Binary
Image')
```

# 手動でしきい値を設定して2値化

- thresholdの値で出力画像も変化する

Original Image

Binary Image

# P-タイル法によるしきい値の決定

```
#P-タイル法
def p_tile_method(image, p=50):
    # ヒストグラムを計算
    hist = cv2.calcHist([image], [0], None, [256], [0, 256])
    #総画素数を取得
    total_pixels = image.size
    #しきい値を初期化
    threshold = 0
    #累積和を初期化
    cumulative_sum = 0

    #濃度値の低いところから累積和を求める
    #比率p%以上となったときの濃度値をしきい値とする
    for i in range(256):
        cumulative_sum += hist[i]
        if cumulative_sum >= total_pixels * (p / 100.0):
            threshold = i
            break

    return threshold
```

# P-タイル法による2値化

```
# P-タイル法を実行
threshold_ptile = p_tile_method(gray_img, p=40)

# しきい値を表示
print(f'P-タイル法で求めたしきい値：{threshold_ptile}')

# P-タイル法で求めたしきい値で2値化
_, ptile_img = cv2.threshold(gray_img, threshold_ptile, 255,
cv2.THRESH_BINARY)

# 実行結果の表示
show_images(gray_img,ptile_img, 'Original Image’, 'P-Tile Binary
Image')
```

# P-タイル法による2値化

- 比率p=40%のとき，しきい値154

Original Image

P-Tile Binary Image

# 判別分析法（大津の2値化）

- cv2.threshold()の2番目の引数を0とし，
  4番目の引数をcv2.THRESH_OTSUとする
- cv2.threshold()の戻り値として，しきい値が出力
  されるのでこれをthreshold_otsuに格納し表示

```
# 判別分析法を適用
threshold_otsu, otsu_img = cv2.threshold(gray_img, 0, 255,
cv2.THRESH_OTSU)

#しきい値を表示する
print(f'判別分析法で決定されたしきい値：{threshold_otsu}')

# 実行結果の表示
show_images(gray_img, otsu_img, 'Original Image','Otsu Binary
Image')
```

# 判別分析法（大津の2値化）

- しきい値を自動で決定して2値化処理

Original Image

Otsu Binary Image