濃淡画像処理:濃度変換

濃度変換

- 表示装置が表示できる濃度値に対して，画像のダイナミックレンジが狭い場合，**コントラストが低く**なる

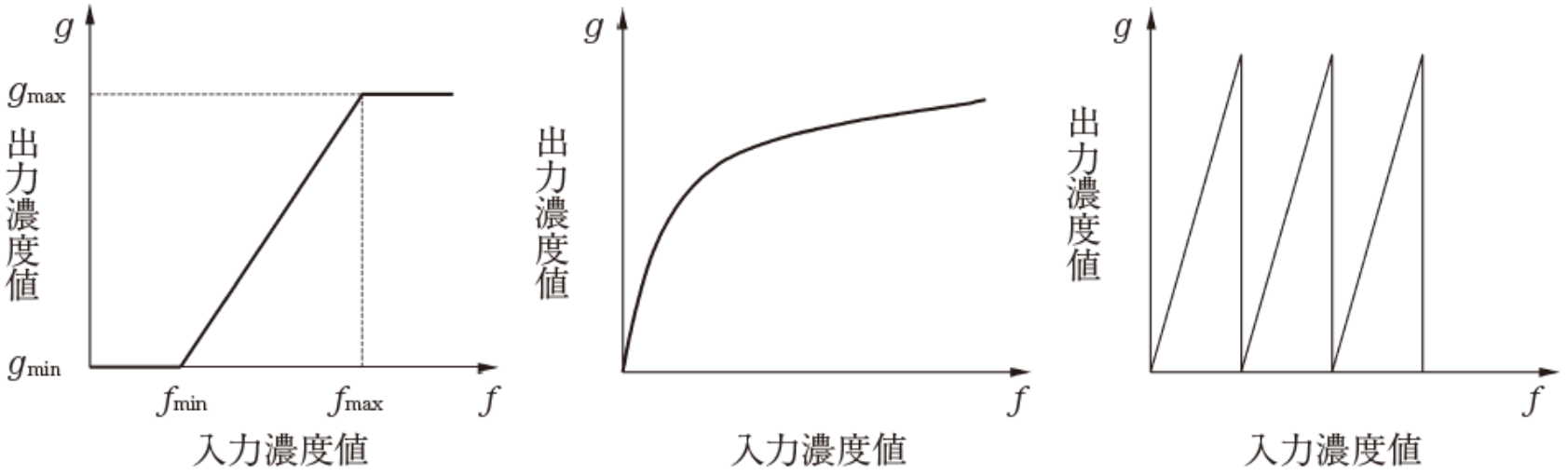
（a） コントラストが悪い画像　　（b） コントラストを改善した画像

図 5.1 濃度変換によるコントラストの改善例

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

濃度変換

- 濃度変換関数(トーンカーブ)の例



（a） 線形変換関数の例　（b） 非線形変換関数の例　（c） のこぎり波状関数の例

図 5.2 濃度変換関数の例

濃度ヒストグラム

- 横軸：**画像の濃度値**（8bit量子化の場合は0～255）
  縦軸：**その濃度値を持つ画素数**

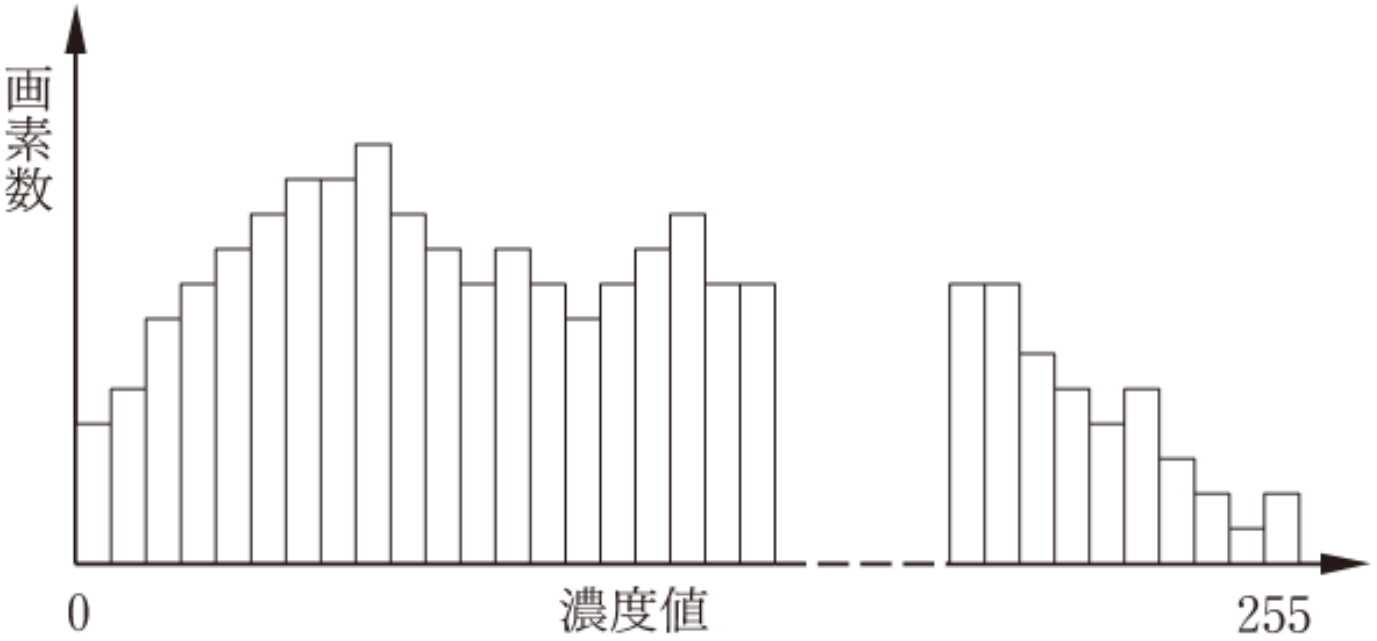
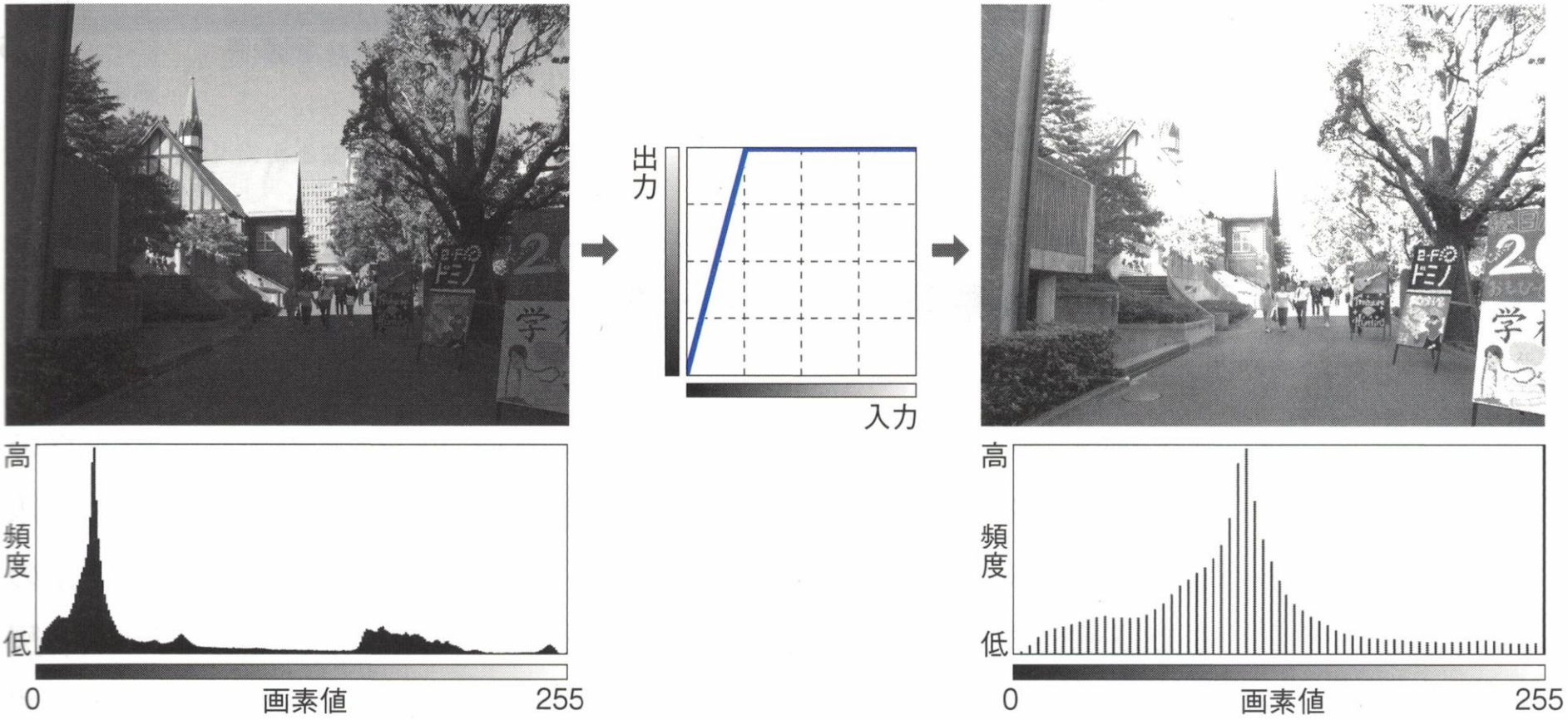


図 5.3　ヒストグラム

線形変換関数による変換例

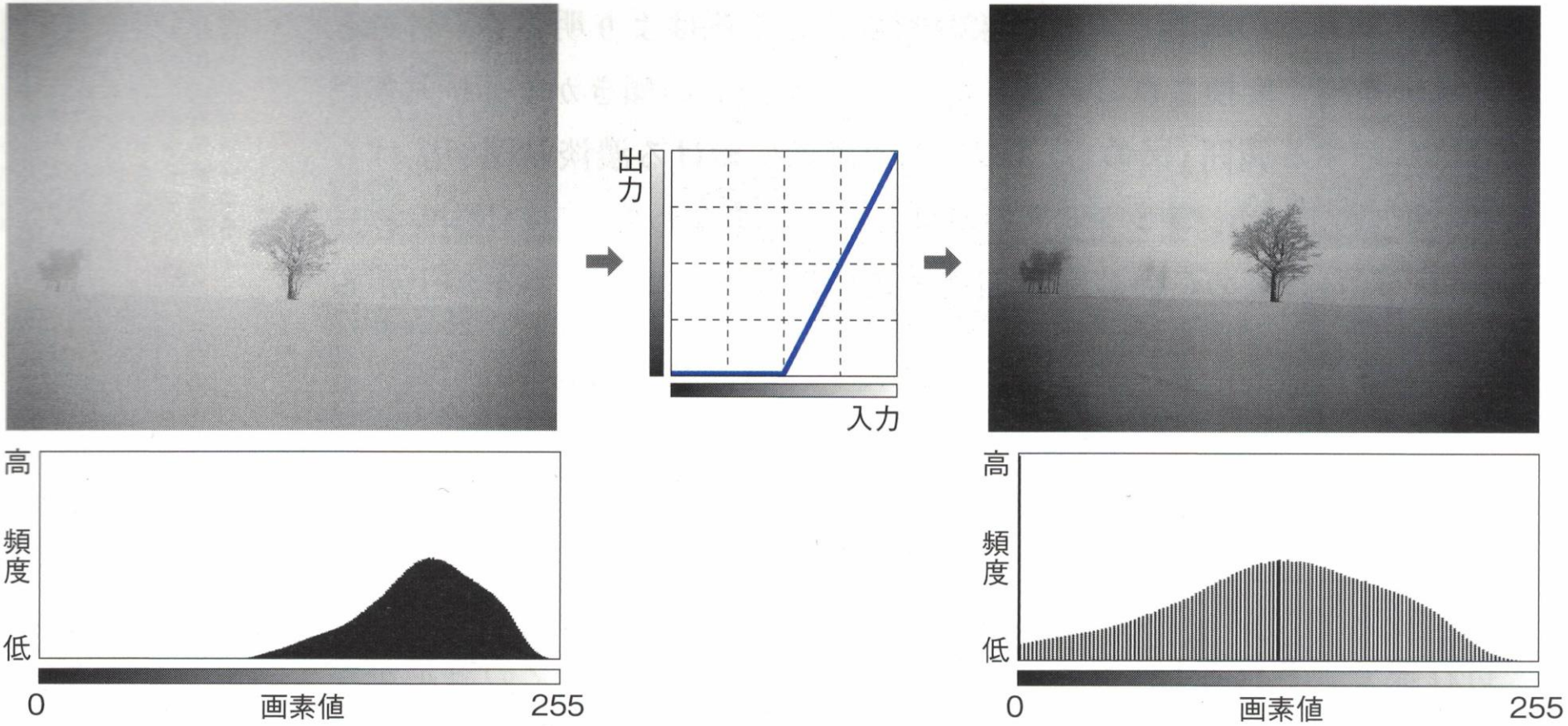


[a] 入力画像　[b] 出力画像

■図4.2——折れ線型トーンカーブによる変換（1）

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

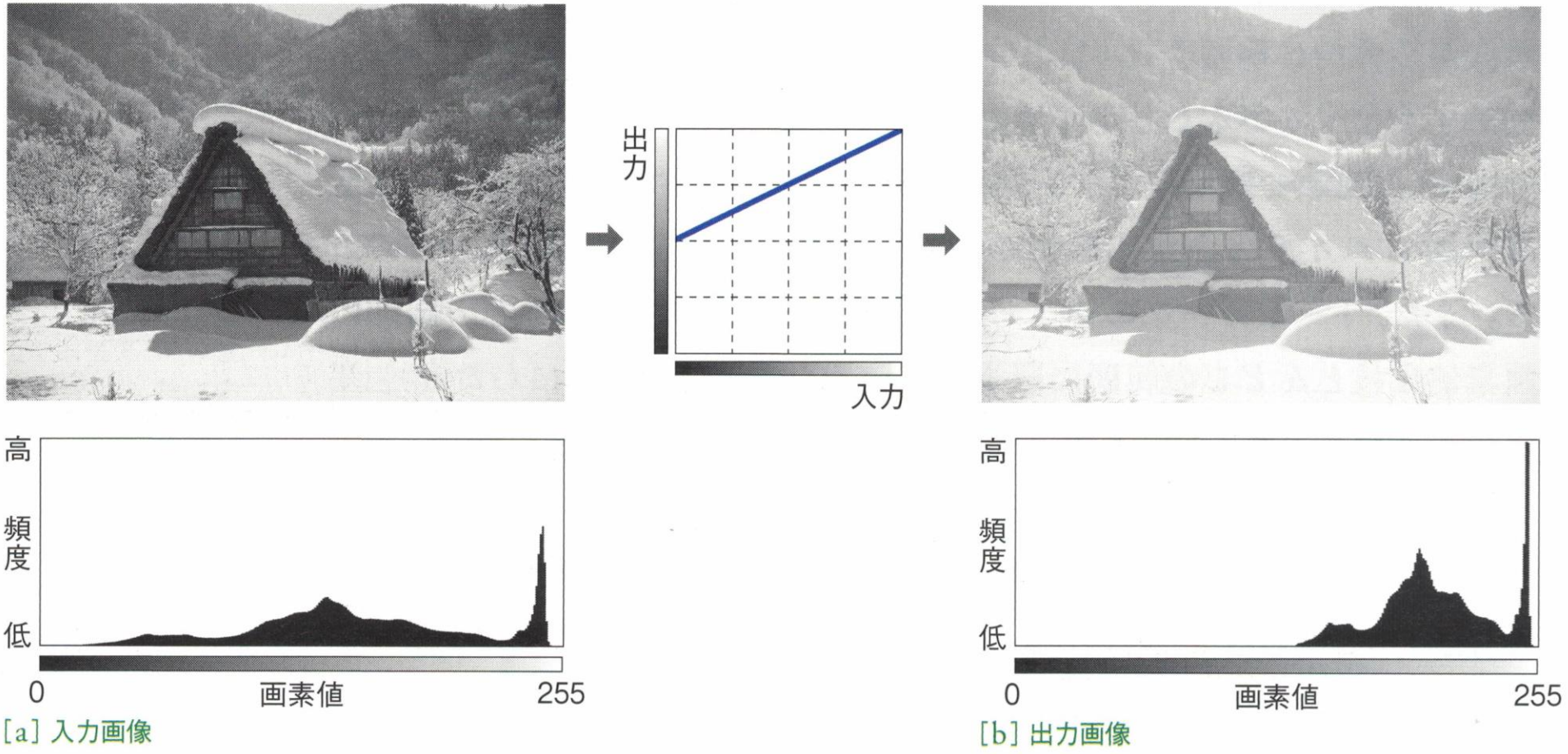
線形変換関数による変換例2



■図4.3——折れ線型トーンカーブによる変換（2）

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

線形変換関数による変換例3



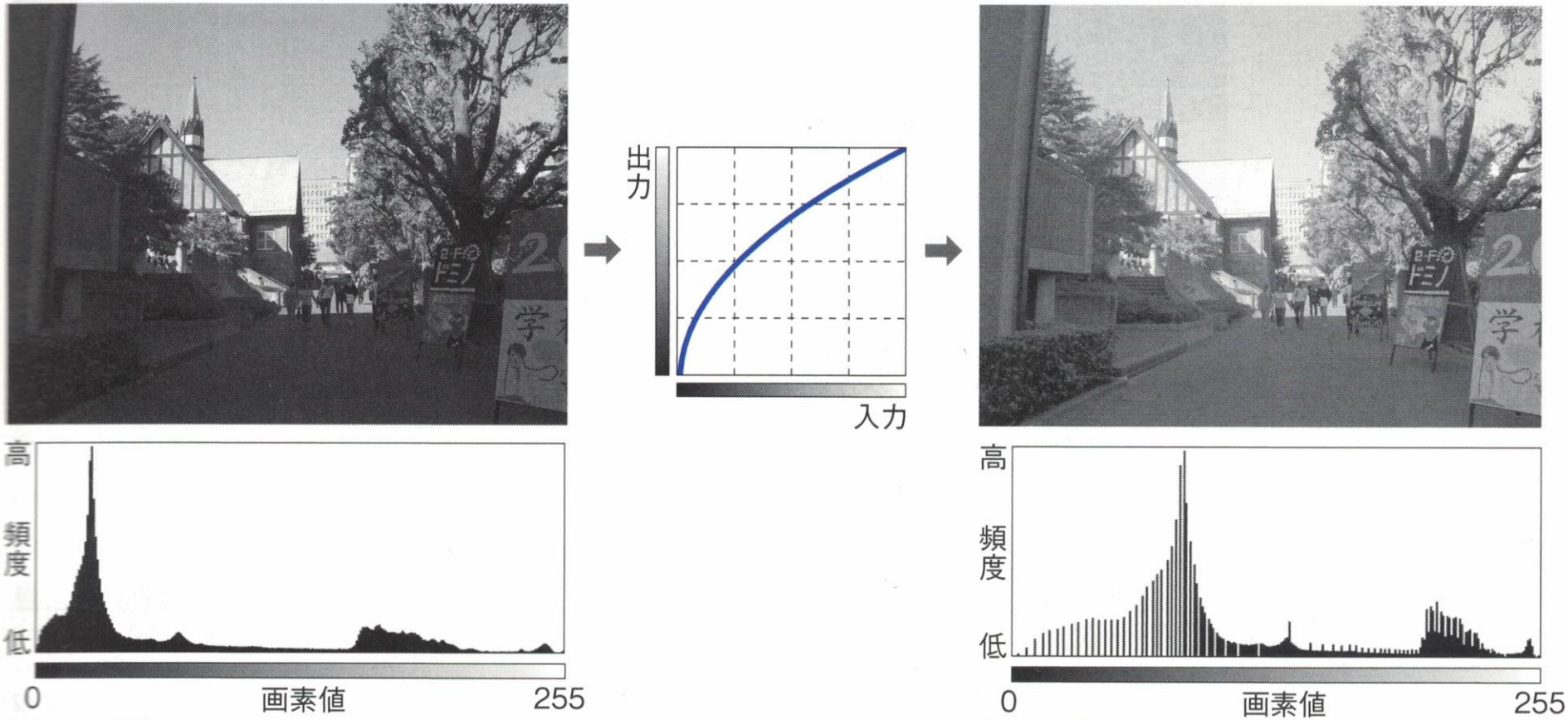
[a] 入力画像

[b] 出力画像

■図4.4——折れ線型トーンカーブによる変換（3）

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

非線形変換関数による変換例



[a] 入力画像

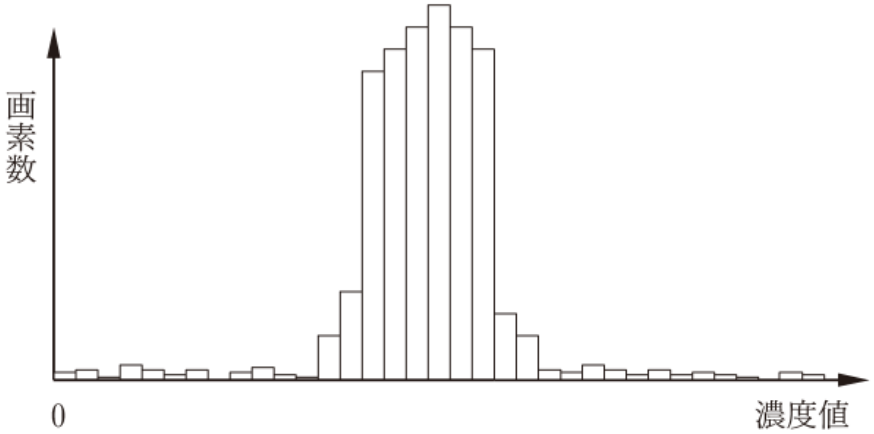
[b] 出力画像

■図4.6—— $\gamma=2$ による変換

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

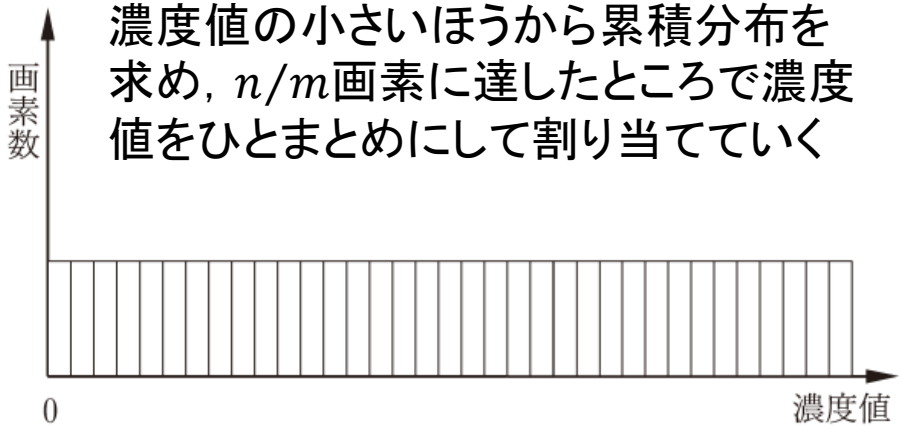
ヒストグラム均一化

- ヒストグラムに大きな偏りがあるとコントラストが悪い
- **濃度値に対する画素の出現率**を均一化することでコントラストを改善



（a） コントラストの悪い画像のヒストグラム

全画素数がn画素，階調数がmとき，濃度値の小さいほうから累積分布を求め，n/m画素に達したところで濃度値をひとまとめにして割り当てていく



（b） 均一化されたヒストグラム

図 5.4 ヒストグラムの均一化

実際のヒストグラム均一化

- 実際には，濃度値の小さい方から累積分布関数CDFを求めて均一化が行われる

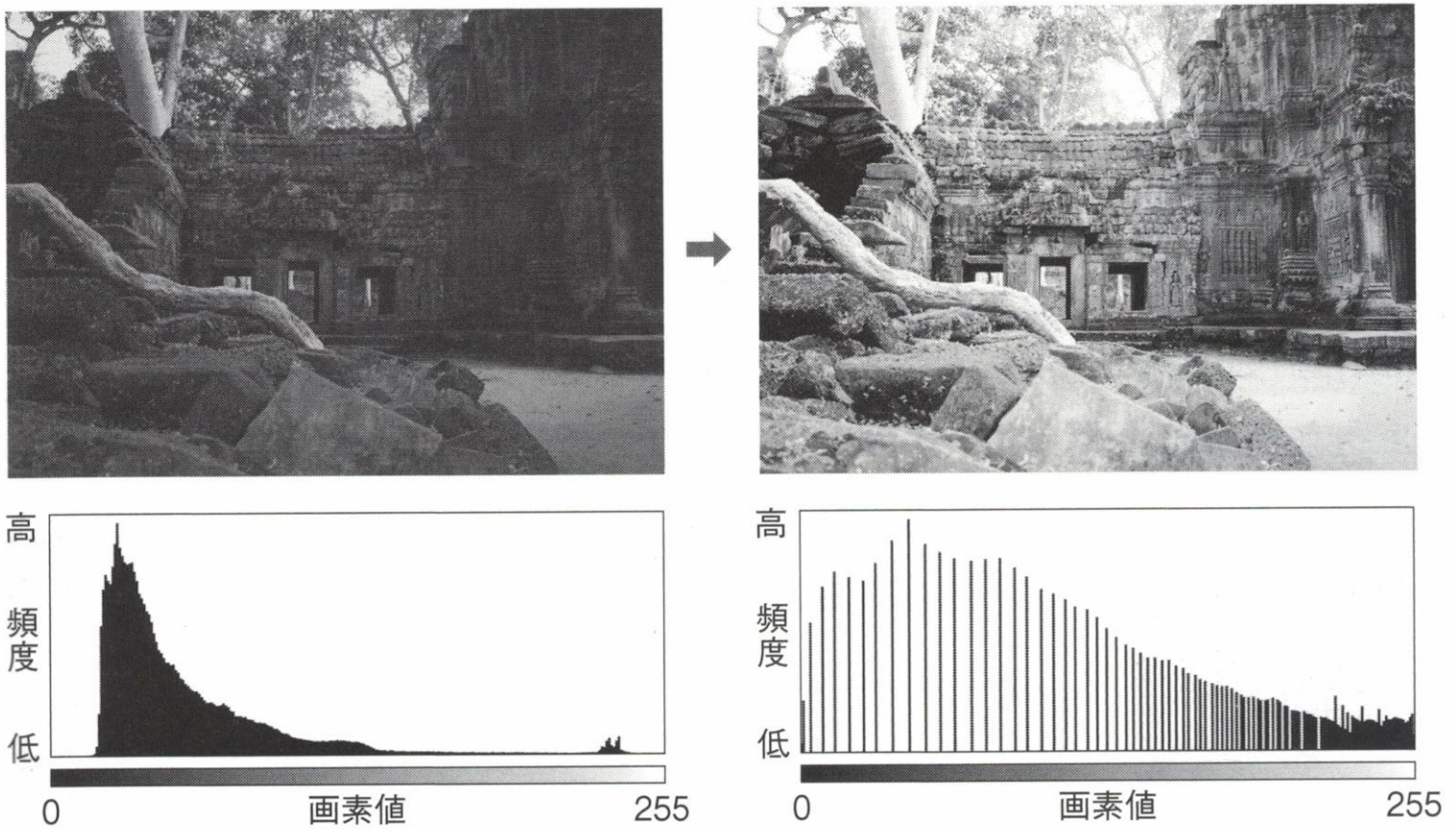
CDF：Cumulative Distribution Function

- 累積分布関数を0～255の整数値に正規化することで均一化が行われる

$$h(v) = \mathrm{round}\left(\frac{CDF(v) - CDF_{min}}{N - CDF_{min}} \times 255\right)$$

$CDF(v)$：濃度値vにおける累積分布関数
CDF_{min}：累積分布関数の最小値（0以外）
N：画像の全画素数
$h(v)$：均一化後の濃度値

実際のヒストグラム平坦化の例



[a] 入力画像　[b] 出力画像

■図4.9—ヒストグラム平坦化の結果

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

濃淡画像処理:平滑化

画像の平滑化

- 画像に乗った**ランダムなノイズを除去**したり，**濃度値の細かい変動を少なくして滑らかな画像にする**処理
- 線形フィルタによる平滑化
  - 移動平均フィルタ・加重平均フィルタ
  - ガウシアンフィルタ
- エッジを保存した平滑化（非線形フィルタ）
  - メディアンフィルタ

空間フィルタリング

- 画像とフィルタを畳み込み演算
- 注目画素を決めて，**対応する位置のフィルタ係数を乗算し，それらの和**を出力データとする

$$g(i,j) = \sum_{n=-w}^{w} \sum_{m=-w}^{w} f(i+m, j+n) h(m,n)$$

50	60	70	70
30	100	60	90
200	210	180	70
150	10	10	30

入力画像 $f(i,j)$

×

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

フィルタ $h(m,n)$

=

26	41	50	32
72	106	93	60
72	105	84	48
63	84	56	32

出力画像 $g(i,j)$

例：3×3移動平均フィルタ

50	60	70
30	100	60
200	210	180

×

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

50 × 1/9 + 60 × 1/9 + 70 × 1/9 +
30 × 1/9 + 100 × 1/9 + 60 × 1/9 +
200 × 1/9 + 210 × 1/9 + 180 × 1/9
= 106

50	60	70	70
30	100	60	90
200	210	180	70
150	10	10	30

入力画像

➡

	106		

出力画像

例：3×3移動平均フィルタ

60	70	70
100	60	90
210	180	70

×

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

60 × 1/9 + 70 × 1/9 + 70 × 1/9 +
100 × 1/9 + 60 × 1/9 + 90 × 1/9 +
210 × 1/9 + 180 × 1/9 + 70 × 1/9
= **93**

50	60	70	70
30	100	60	90
200	210	180	70
150	10	10	30

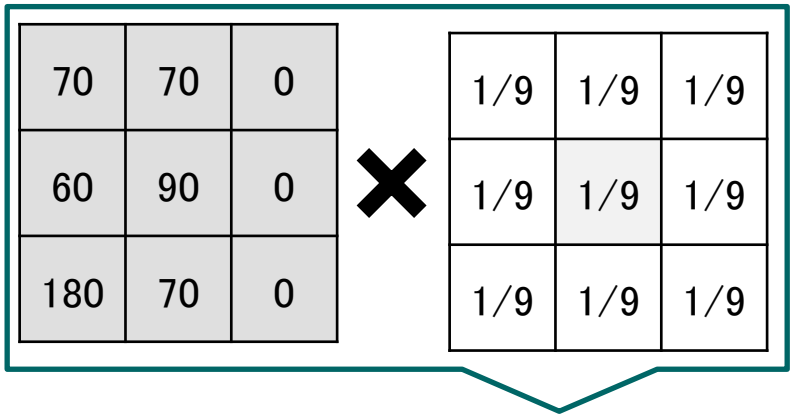
入力画像

➡

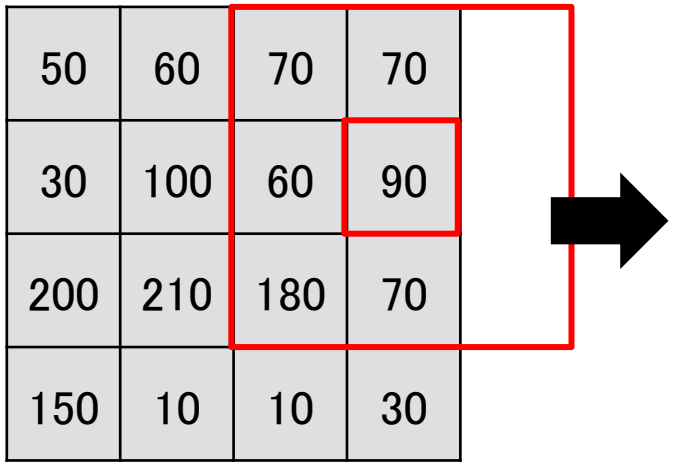
	106	93	

出力画像

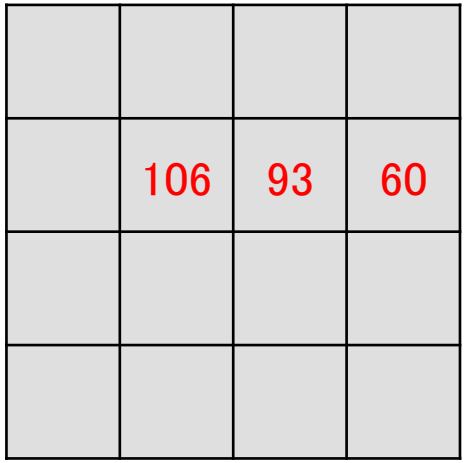
例：3×3移動平均フィルタ



$70 \times 1/9 + 70 \times 1/9 + 0 \times 1/9 + 60 \times 1/9 + 90 \times 1/9 + 0 \times 1/9 + 180 \times 1/9 + 70 \times 1/9 + 0 \times 1/9 = 60$



入力画像



出力画像

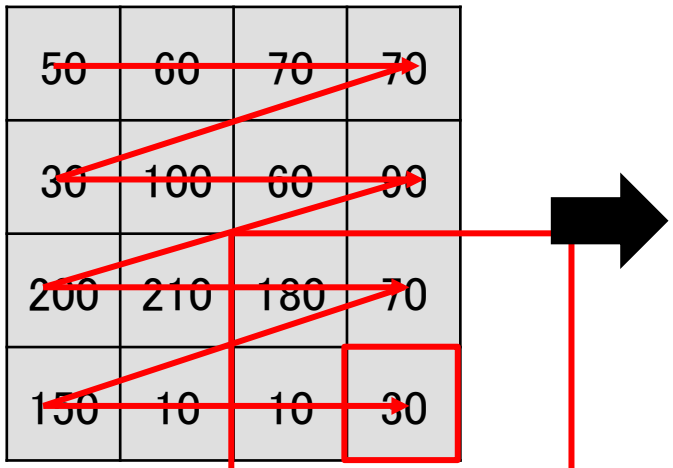
例:3×3移動平均フィルタ

180	70	0
10	30	0
0	0	0

×

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

180 × 1/9 + 70 × 1/9 + 0×1/9 +
10 × 1/9 + 30 × 1/9 + 0 × 1/9 +
0 × 1/9 + 0 × 1/9 + 0 × 1/9
= **32**



入力画像

26	41	50	32
72	106	93	60
72	105	84	48
63	84	56	32

出力画像

移動平均フィルタ

- 重み係数が全て等しいフィルタ

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

3×3

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

5×5

加重平均フィルタ

- **注目画素の近傍の重みを大きく**したフィルタ
- 移動平均フィルタと比べて**画像のぼけを抑制**できる

1/10	1/10	1/10
1/10	2/10	1/10
1/10	1/10	1/10

0	1/5	0
1/5	1/5	1/5
0	1/5	0

３×３加重平均フィルタの例

ガウシアンフィルタ

- **正規分布（ガウス分布）の重み係数**を用いたフィルタ

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

$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$
$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$
$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$

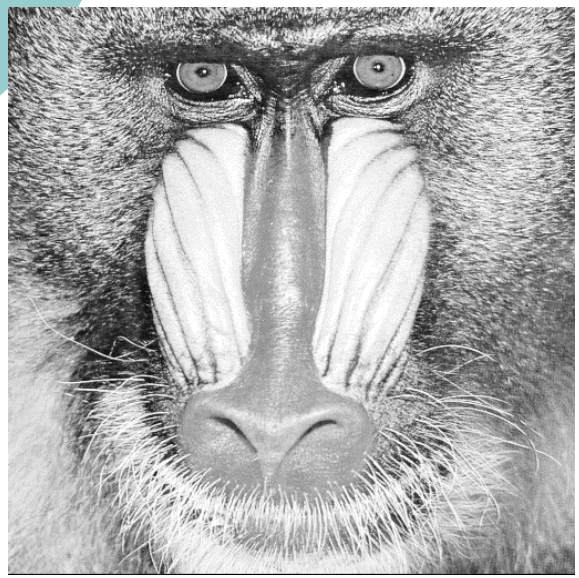
3×3

$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{6}{256}$	$\frac{24}{256}$	$\frac{36}{256}$	$\frac{24}{256}$	$\frac{6}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$

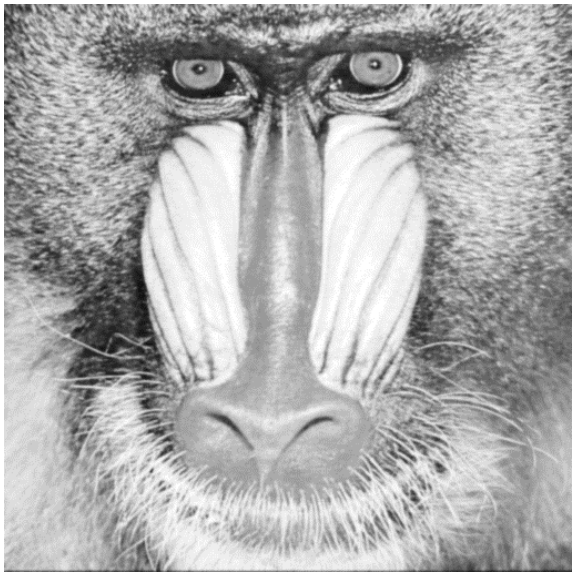
5×5

ガウシアンフィルタの効果

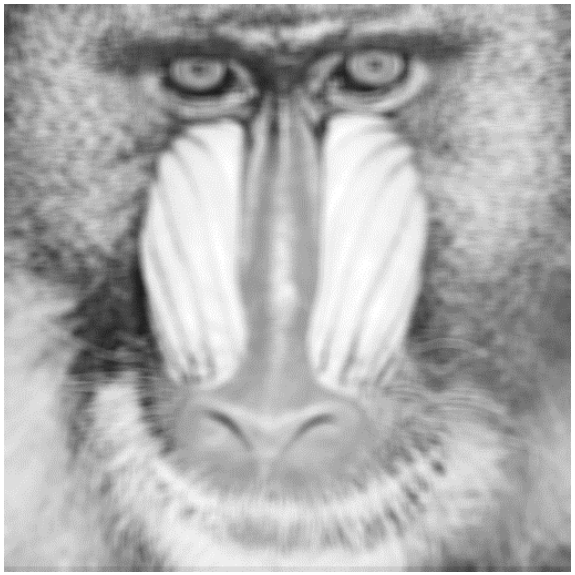
- 同じフィルタサイズで比較すると，
  ガウシアンフィルタの方が画像がぼけにくい



原画像



９×９ガウシアン
フィルタ適用後

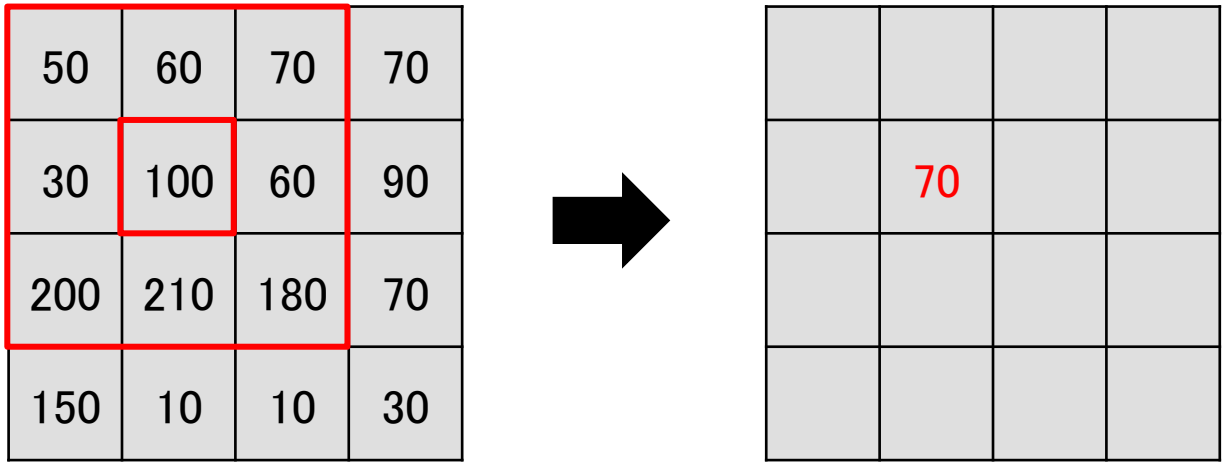


９×９移動平均
フィルタ適用後

メディアンフィルタ

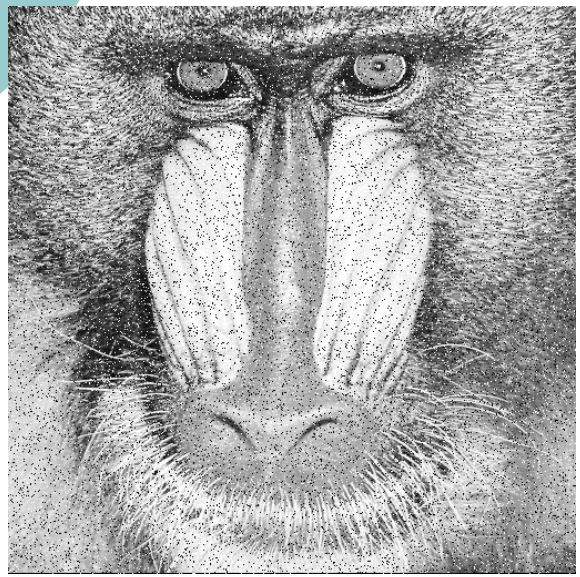
- **中央値**を取り出すフィルタ（非線形フィルタ）
- ゴマ塩ノイズを良好に除去できる
- **データのソート**が必要になるため，
  線形フィルタよりも計算に時間がかかる

[30, 50, 60, 60, 70, 100, 180, 200,210]

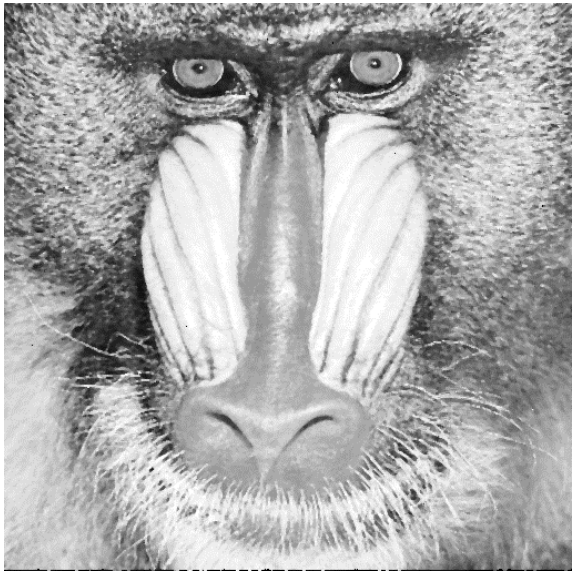


ごま塩ノイズの除去

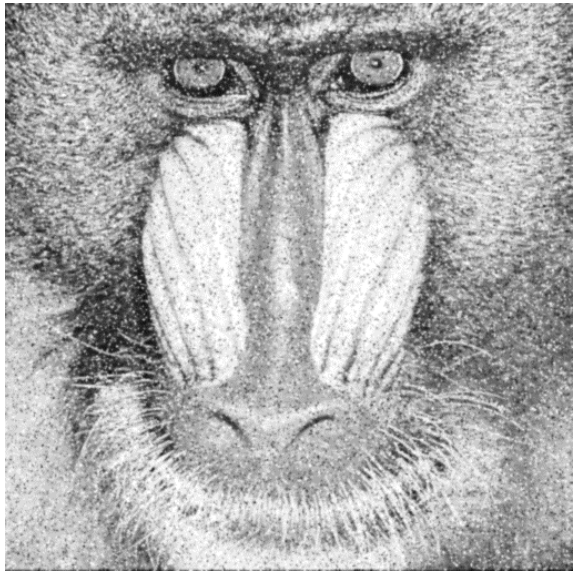
- ごま塩ノイズのようなスパイク状のノイズは，
  メディアンフィルタにより良好に除去できる



ごま塩ノイズを
付与した画像



３×３メディアン
フィルタ適用後



５×５ガウシアン
フィルタ適用後

画像処理プログラミング２
濃度変換

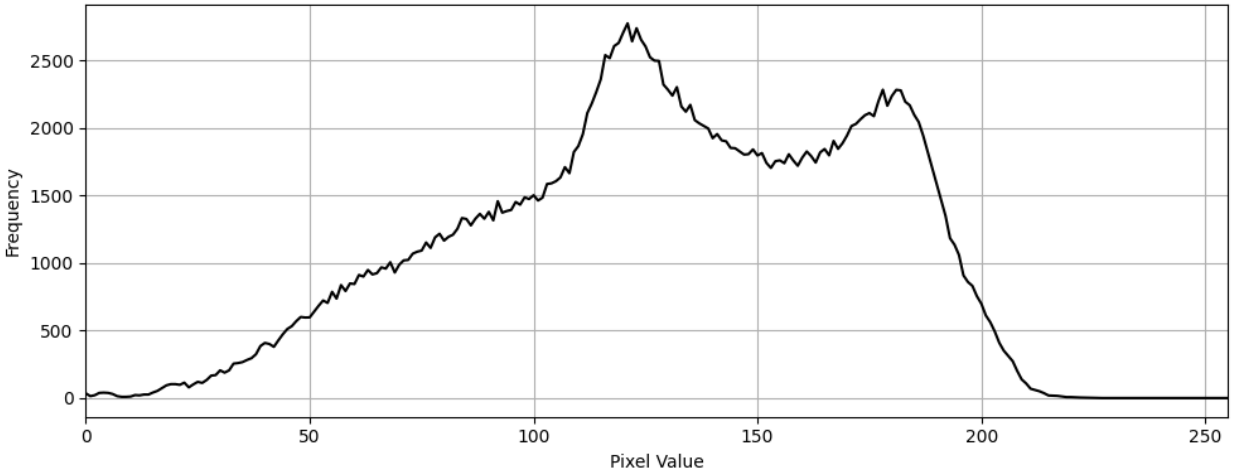
ヒストグラムを描画する関数

```
# ヒストグラムを描画する関数
def plot_grayscale_histogram(image):

    # OpenCVでヒストグラムを計算
    hist = cv2.calcHist([image], [0], None, [256], [0, 256])

    # ヒストグラムを描画
    plt.figure(figsize=(10, 4))
    plt.plot(hist, color='black')
    plt.xlabel('Pixel Value')
    plt.ylabel('Frequency')
    plt.xlim([0, 255])
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

グレースケール画像を読み込んで濃度ヒストグラムを作成

- 前回の演習で生成したグレースケール画像を読み込んでヒストグラムを表示する（再度ドライブのマウントをしてから実行）

```
#画像を読み込む
gray_img = cv2.imread(common_path + 'gray_image.png', cv2.IMREAD_GRAYSCALE)
#ヒストグラムを表示する
plot_grayscale_histogram(gray_img)
```



線形変換関数による濃度変換

- パラメータα, βを指定して変換

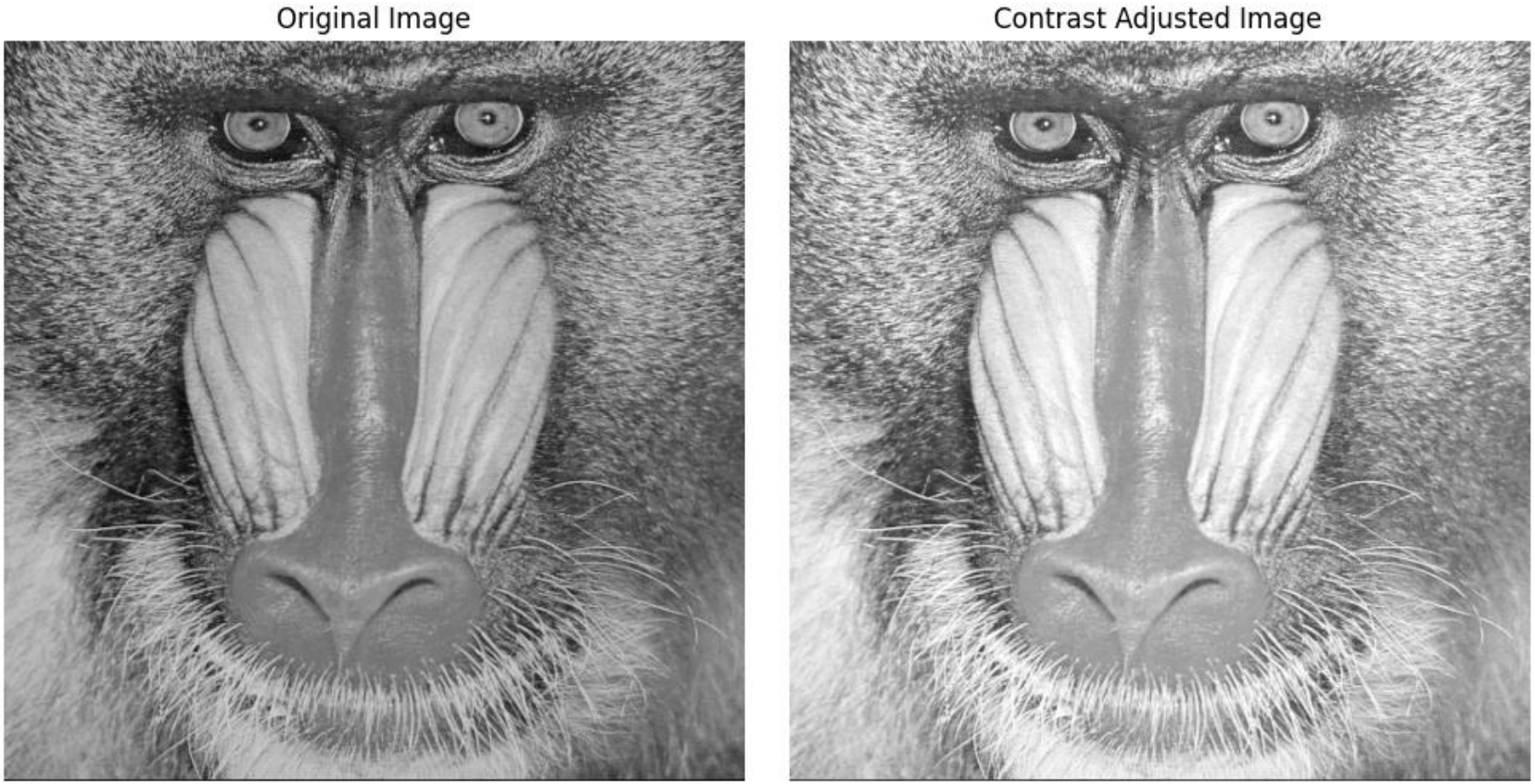
$$g = \alpha f + \beta$$

- αでコントラスト，βで明るさを調整できる
- cv2.convertScaleAbs()を用いる
  オーバーフローした値は255に値が丸め込まれる

```
#線形変換関数による濃度変換
adjusted_img = cv2.convertScaleAbs(gray_img, alpha=1.1, beta=30)

#結果を表示
show_images(gray_img, adjusted_img, 'Original Image', 'Contrast Adjusted Image')
```

線形変換関数による濃度変換

- $\alpha = 1.1, \beta = 30$の場合

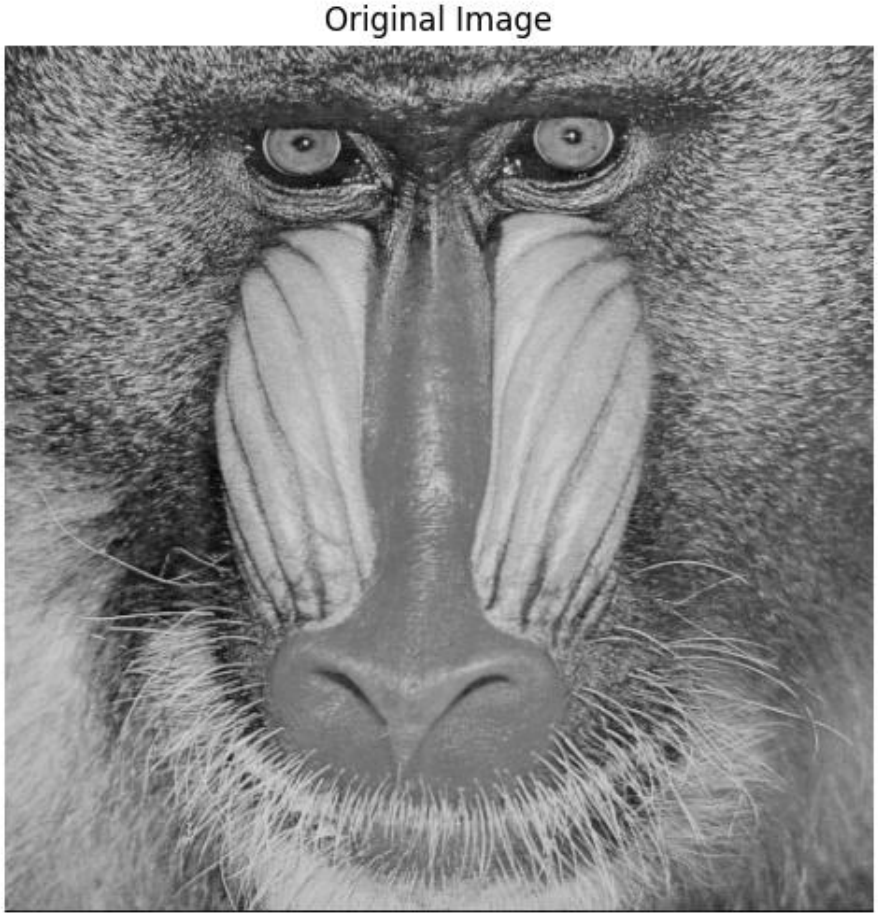


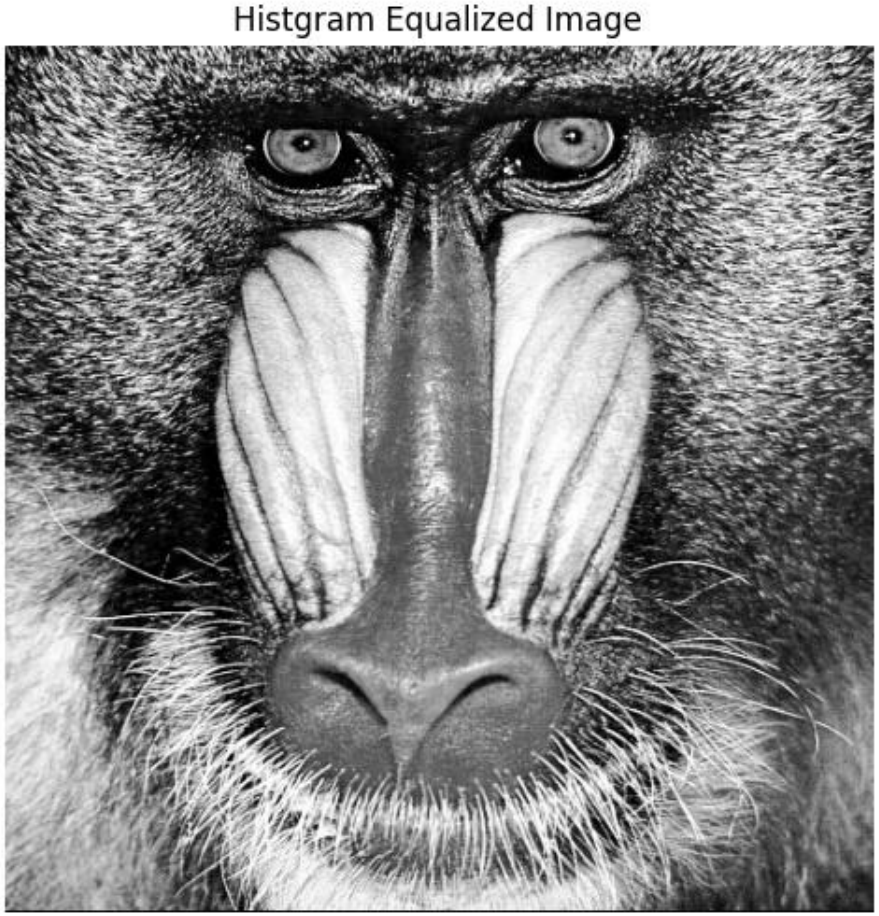
ヒストグラムの均一化

- cv2.equalizeHist関数を使ってヒストグラム均一化

```
# ヒストグラム均一化
equalized_img = cv2.equalizeHist(gray_img)

# 実行結果の表示
show_images(gray_img,equalized_img, 'Original Image',
'Histgram Equalized Image')
```

ヒストグラムの均一化

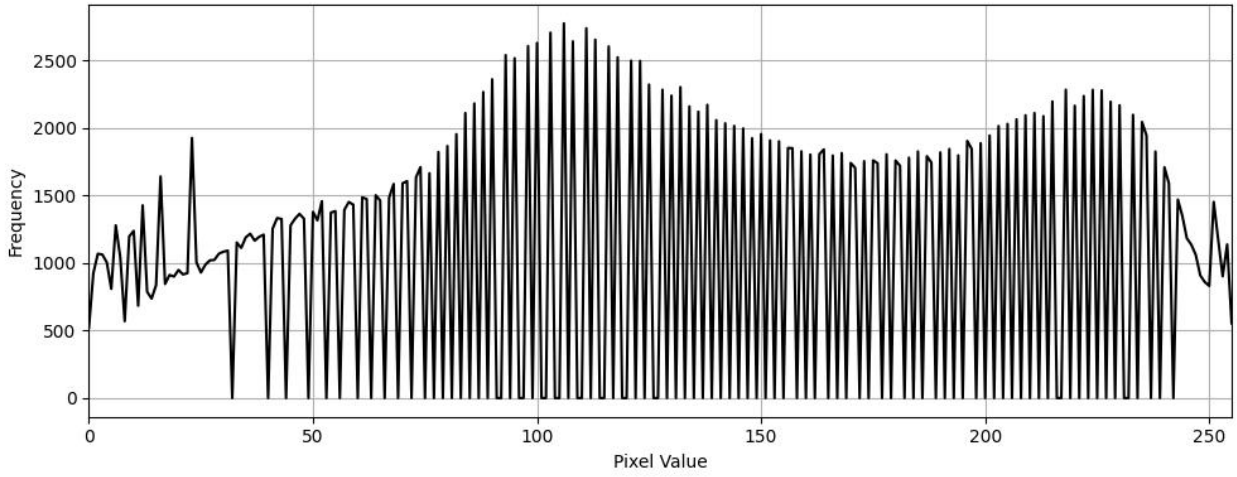




均一化後のヒストグラムの表示

- 累積分布関数を見て均一化されるため，
  濃度値あたりの画素数は必ずしも一定にはならない

```
#均一化後のヒストグラムを表示する
plot_grayscale_histogram(equalized_img)
```



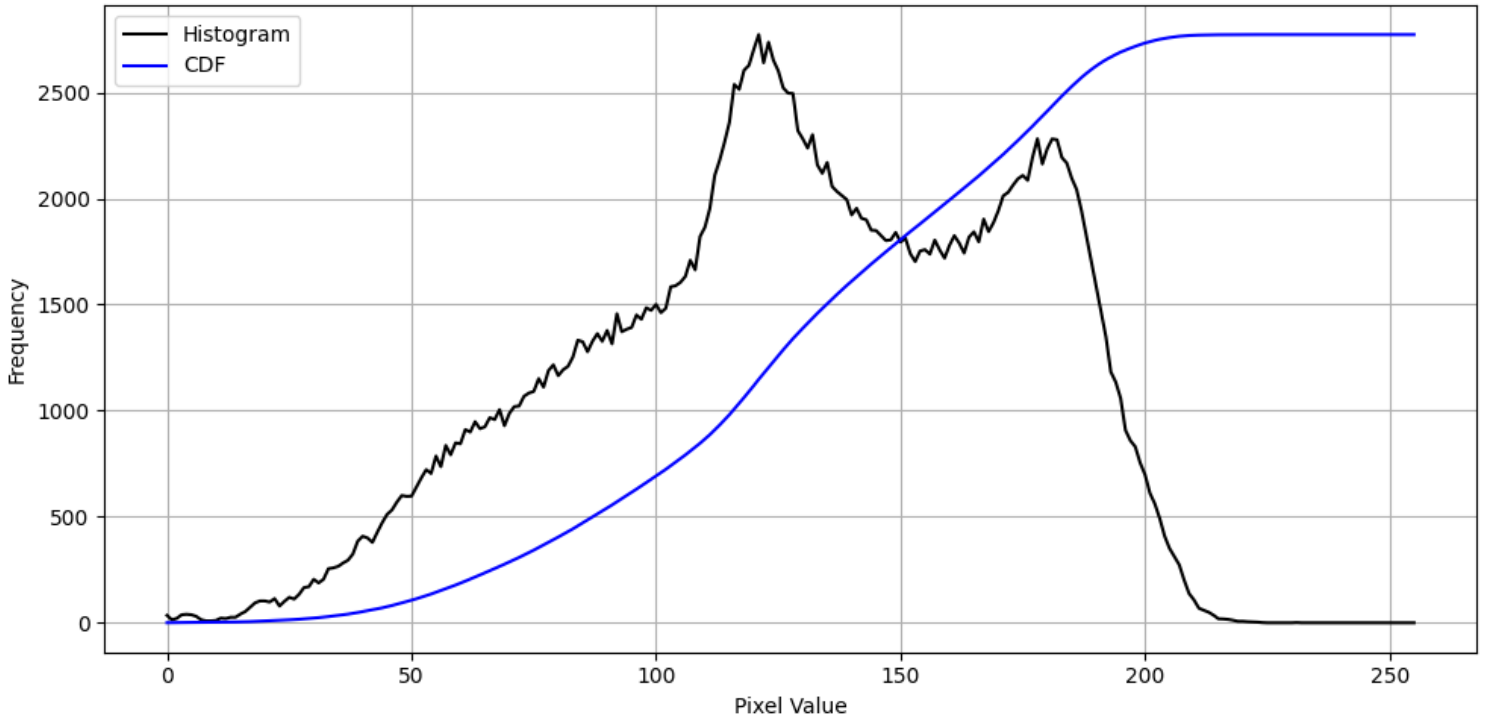
累積分布関数も合わせて表示する

```
# 累積分布関数とヒストグラムを描画する関数
def plot_grayscale_histogram_cdf(image):

    # ヒストグラムの計算
    hist = cv2.calcHist([image], [0], None, [256], [0, 256])

    # 累積分布関数(CDF)の計算
    cdf = hist.cumsum()

    # ヒストグラムと同じスケールに正規化
    cdf_normalized = cdf / cdf.max() * hist.max()

    # グラフ描画
    plt.figure(figsize=(10, 5))
    plt.plot(hist, color='black', label='Histogram')
    plt.plot(cdf_normalized, color='blue', label='CDF')
    plt.xlabel('Pixel Value')
    plt.ylabel('Frequency')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

元の画像のヒストグラムとCDF

- 累積分布関数の増加量が一定ではない

```
#グレースケール画像の累積分布関数とヒストグラムを表示
plot_grayscale_histogram_cdf(gray_img)
```



均一化後のヒストグラムとCDF

- 累積分布関数の増加量が一定となり均一化

```
#均一化後の累積分布関数とヒストグラムを表示
plot_grayscale_histogram_cdf(equalized_img)
```