

画像処理・画像処理工学 レポート課題 1

画像処理工学科 学籍番号: 21239 組番号: 234 5E 氏名: 柳原 魁人

2025 年 11 月 24 日

本レポートの作成にあたり、GitHub Copilot により作成した TeX の雛形を使用しました。また、Markdown 形式の原稿を TeX 構文に変換する際に、Google AI Studio および GitHub Copilot 内の生成 AI を活用しました。

1 問題 1

1.1 1) 複合レンズ系による結像

本問では、凸レンズの公式 $\frac{1}{a} + \frac{1}{b} = \frac{1}{f}$ を用い、レンズ A による像を求めた後、その像をレンズ B の物体として再度計算を行う。

1.1.1 凸レンズ A による像（中間像）の計算

レンズ A について、物体距離 $a_A = 16\text{ cm}$ 、焦点距離 $f_A = 12\text{ cm}$ である。像の位置 b_A を求める。

$$\frac{1}{16} + \frac{1}{b_A} = \frac{1}{12}$$

$$\frac{1}{b_A} = \frac{1}{12} - \frac{1}{16}$$

$$\frac{1}{b_A} = \frac{4}{48} - \frac{3}{48}$$

$$\frac{1}{b_A} = \frac{1}{48}$$

よって、 $b_A = 48\text{ cm}$ である。これはレンズ A の後方 48 cm の位置に実像ができるることを示す。

次に倍率 m_A を求める。

$$m_A = -\frac{b_A}{a_A} = -\frac{48}{16} = -3$$

物体の大きさ $h = 2.0\text{ cm}$ より、中間像の大きさ h'_A は以下のようになる。

$$h'_A = |m_A| \times h = 3 \times 2.0 = 6.0\text{ cm}$$

1.1.2 凸レンズ B による像（最終像）の計算

レンズ A とレンズ B の間隔は 63 cm である。レンズ A による像（レンズ A の後方 48 cm ）をレンズ B に対する物体とみなす。レンズ B から見た物体距離 a_B は以下の通りである。

$$a_B = 63 - 48 = 15\text{ cm}$$

レンズ B の焦点距離 $f_B = 10\text{ cm}$ を用い、最終像の位置 b_B を求める。

$$\frac{1}{15} + \frac{1}{b_B} = \frac{1}{10}$$

移項して通分を行う。

$$\frac{1}{b_B} = \frac{1}{10} - \frac{1}{15}$$

$$\frac{1}{b_B} = \frac{3}{30} - \frac{2}{30}$$

$$\frac{1}{b_B} = \frac{1}{30}$$

よって、 $b_B = 30\text{ cm}$ である。

倍率 m_B を求める。

$$m_B = -\frac{b_B}{a_B} = -\frac{30}{15} = -2$$

最終的な像の大きさ h'_B は、中間像の大きさ h'_A に倍率を乗じて求められる。

$$h'_B = |m_B| \times h'_A = 2 \times 6.0 = 12.0\text{ cm}$$

解答

- 像の位置: 凸レンズ B の後方 **30 cm**
 - 像の大きさ: **12.0 cm**
-

1.2 2) デバイスの ppi 比較

画素密度 ppi (pixels per inch) は、画面解像度の幅 w 、高さ h 、および対角線インチ数 d を用いて以下の式で定義される。

$$\text{ppi} = \frac{\sqrt{w^2 + h^2}}{d}$$

1.2.1 Google Pixel 10

$d = 6.3\text{ インチ}$, $w = 1080$, $h = 2424$ を代入する。まず対角線の画素数を計算する。

$$\sqrt{1080^2 + 2424^2} = \sqrt{1166400 + 5875776} = \sqrt{7042176} \approx 2653.71$$

インチ数で除算する。

$$\text{ppi} = \frac{2653.71}{6.3} \approx 421.22$$

1.2.2 iPad (A16)

$d = 10.9\text{ インチ}$, $w = 2360$, $h = 1640$ を代入する。

$$\sqrt{2360^2 + 1640^2} = \sqrt{5569600 + 2689600} = \sqrt{8259200} \approx 2873.88$$

$$\text{ppi} = \frac{2873.88}{10.9} \approx 263.65$$

1.2.3 EIZO EV2740S

$d = 27.0\text{ インチ}$, $w = 3840$, $h = 2160$ を代入する。

$$\sqrt{3840^2 + 2160^2} = \sqrt{14745600 + 4665600} = \sqrt{19411200} \approx 4405.81$$

$$\text{ppi} = \frac{4405.81}{27.0} \approx 163.17$$

解答

整数値で比較すると以下の通りとなる。

- Google Pixel 10: **421 ppi**
 - iPad (A16): **264 ppi**
 - EIZO EV2740S: **163 ppi**
-

1.3 3) 8 近傍鮮鋭化フィルタの導出

1.3.1 ラプラシアンフィルタの導出

注目画素を $f(i, j)$ とし、各方向の 2 次差分を以下のように定義する。

- x 方向 (f_{xx}): 隣接画素との 1 次差分の差分をとる。

$$\begin{aligned} f_{xx} &= \{f(i+1, j) - f(i, j)\} - \{f(i, j) - f(i-1, j)\} \\ &= f(i-1, j) + f(i+1, j) - 2f(i, j) \end{aligned}$$

- y 方向 (f_{yy}):

$$\begin{aligned} f_{yy} &= \{f(i, j+1) - f(i, j)\} - \{f(i, j) - f(i, j-1)\} \\ &= f(i, j-1) + f(i, j+1) - 2f(i, j) \end{aligned}$$

- 斜め方向 1 (左上-右下, f_{d1}):

$$\begin{aligned} f_{d1} &= \{f(i+1, j+1) - f(i, j)\} - \{f(i, j) - f(i-1, j-1)\} \\ &= f(i-1, j-1) + f(i+1, j+1) - 2f(i, j) \end{aligned}$$

- 斜め方向 2 (右上-左下, f_{d2}):

$$\begin{aligned} f_{d2} &= \{f(i+1, j-1) - f(i, j)\} - \{f(i, j) - f(i-1, j+1)\} \\ &= f(i-1, j+1) + f(i+1, j-1) - 2f(i, j) \end{aligned}$$

8 近傍ラプラシアン $\nabla^2 f$ は、これら 4 方向の 2 次差分の総和である。

$$\nabla^2 f = f_{xx} + f_{yy} + f_{d1} + f_{d2}$$

上式に各差分式を代入して整理すると、注目画素 $f(i, j)$ の係数は $-2 \times 4 = -8$ となり、周囲 8 近傍の画素 $f(i \pm 1, j \pm 1)$ 等の係数はすべて 1 となる。したがって、対応するオペレータ H_L は以下の通り導出される。

$$H_L = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

1.3.2 鮮鋭化フィルタの導出

鮮鋭化（アンシャープマスキング）は、元画像からラプラシアン（エッジ成分の 2 次微分）を引くことで行われる。

$$g(x, y) = f(x, y) - \nabla^2 f(x, y)$$

これを行列演算として記述する。元画像を表す単位オペレータを H_I とすると、鮮鋭化フィルタ H_S は $H_I - H_L$ で求められる。

$$H_S = H_I - H_L = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

各要素ごとの引き算を行う。

$$H_S = \begin{pmatrix} 0-1 & 0-1 & 0-1 \\ 0-1 & 1-(-8) & 0-1 \\ 0-1 & 0-1 & 0-1 \end{pmatrix} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

解答

導出された 8 近傍鮮鋭化フィルタのオペレータ：

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

1.4 4) Sobel フィルタによるエッジ解析

Sobel フィルタのカーネル S_x (水平微分)、 S_y (垂直微分) を以下のように定義する。

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, \quad S_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

エッジ強度 G と方向 θ は次式で求められる。

$$G = \sqrt{G_x^2 + G_y^2}, \quad \theta = \arctan\left(\frac{G_y}{G_x}\right)$$

1.4.1 注目画素 A (1,1) の計算

図 A-3 より、注目画素 A を中心とする 3×3 近傍領域の画素値を以下のように読み取る。

$$\text{Image}_A = \begin{pmatrix} 50 & 50 & 100 \\ 50 & 50 & 100 \\ 100 & 100 & 150 \end{pmatrix}$$

■x 方向微分 G_x 右列と左列の差分を計算する。

$$\begin{aligned} G_x &= (1 \cdot 100 + 2 \cdot 100 + 1 \cdot 150) - (1 \cdot 50 + 2 \cdot 50 + 1 \cdot 100) \\ &= (100 + 200 + 150) - (50 + 100 + 100) \\ &= 450 - 250 = 200 \end{aligned}$$

■y 方向微分 G_y 下行と上行の差分を計算する。

$$\begin{aligned} G_y &= (1 \cdot 100 + 2 \cdot 100 + 1 \cdot 150) - (1 \cdot 50 + 2 \cdot 50 + 1 \cdot 100) \\ &= (100 + 200 + 150) - (50 + 100 + 100) \\ &= 450 - 250 = 200 \end{aligned}$$

■エッジ強度と方向

$$G_A = \sqrt{200^2 + 200^2} = \sqrt{80000} \approx 282.84$$

$$\theta_A = \arctan\left(\frac{200}{200}\right) = \arctan(1) = 45^\circ$$

1.4.2 注目画素 B (3,3) の計算

同様に、注目画素 B を中心とする近傍領域を以下のように読み取る。

$$\text{Image}_B = \begin{pmatrix} 150 & 250 & 250 \\ 150 & 150 & 250 \\ 100 & 100 & 200 \end{pmatrix}$$

■x 方向微分 G_x

$$\begin{aligned} G_x &= (1 \cdot 250 + 2 \cdot 250 + 1 \cdot 200) - (1 \cdot 150 + 2 \cdot 150 + 1 \cdot 100) \\ &= (250 + 500 + 200) - (150 + 300 + 100) \\ &= 950 - 550 = 400 \end{aligned}$$

■y 方向微分 G_y

$$\begin{aligned} G_y &= (1 \cdot 100 + 2 \cdot 100 + 1 \cdot 200) - (1 \cdot 150 + 2 \cdot 250 + 1 \cdot 250) \\ &= (100 + 200 + 200) - (150 + 500 + 250) \\ &= 500 - 900 = -400 \end{aligned}$$

■エッジ強度と方向

$$G_B = \sqrt{400^2 + (-400)^2} = \sqrt{320000} \approx 565.69$$

$$\theta_B = \arctan\left(\frac{-400}{400}\right) = \arctan(-1) = -45^\circ$$

比較結果

- 強度: 画素 B のエッジ強度 (≈ 566) は、画素 A (≈ 283) の約 2 倍である。
- 方向: 画素 A は 45° (右下がり方向の輝度変化)、画素 B は -45° (右上がり方向の輝度変化) のエッジを示している。

2 問題 2

2.1 1) 線形変換によるコントラスト強調

図 A-4 のグラフから、入力画素値 f と出力画素値 g の関係式（トーンカーブ）を導出した。グラフは入力範囲 $0 \leq f \leq 200$ において直線的に増加し、 $200 < f \leq 255$ では飽和している。直線の始点は $(0, 15)$ 、終点は $(200, 255)$ であるため、傾き a は以下のようになる。

$$a = \frac{255 - 15}{200 - 0} = \frac{240}{200} = 1.2$$

切片は 15 であるため、変換式は以下の通りである。

$$g(f) = \begin{cases} 1.2f + 15 & (0 \leq f \leq 200) \\ 255 & (200 < f \leq 255) \end{cases}$$

この変換を適用した結果、元画像で低輝度に集中していた分布が全体的に明るい方向へシフトし、ヒストグラムの幅が広がった。これにより画像のコントラストが強調され、視認性が向上したことが確認できた。

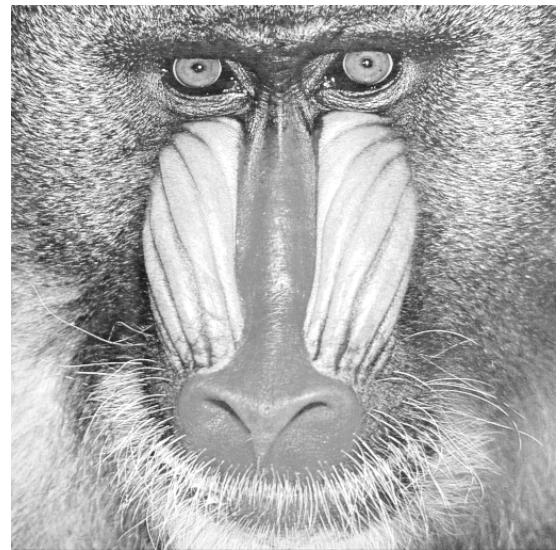
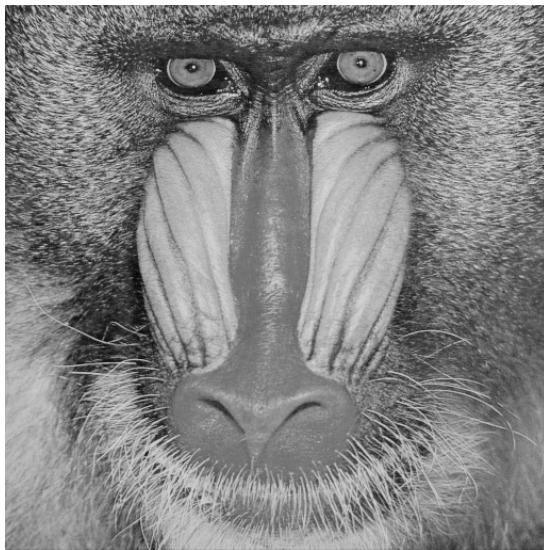


図 1 元画像と変換後画像

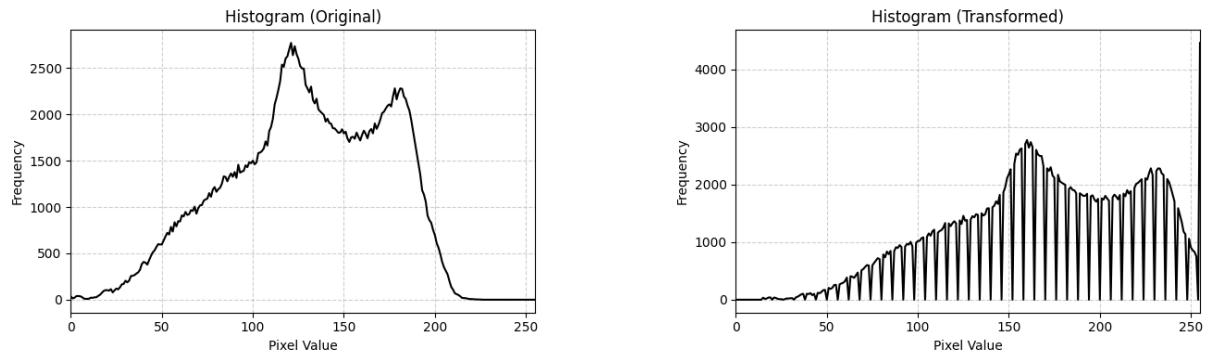


図 2 元画像のヒストグラムと変換後画像のヒストグラム

2.2 2) 輝度反転

画像の輝度反転を行うため、以下の変換式を用いた。

$$g(f) = 255 - f$$

これは8ビット画像の最大輝度255から入力値を引く操作であり、黒(0)は白(255)に、白(255)は黒(0)に変換される。処理結果として、元画像の明暗が完全に逆転した画像が得られた。ヒストグラムにおいても、輝度分布が左右反転（高輝度側と低輝度側を入れ替わる）していることが確認でき、理論通りの結果となった。



図3 輝度反転の比較: 元画像 (building.png) と反転後 (inverted_image.png)

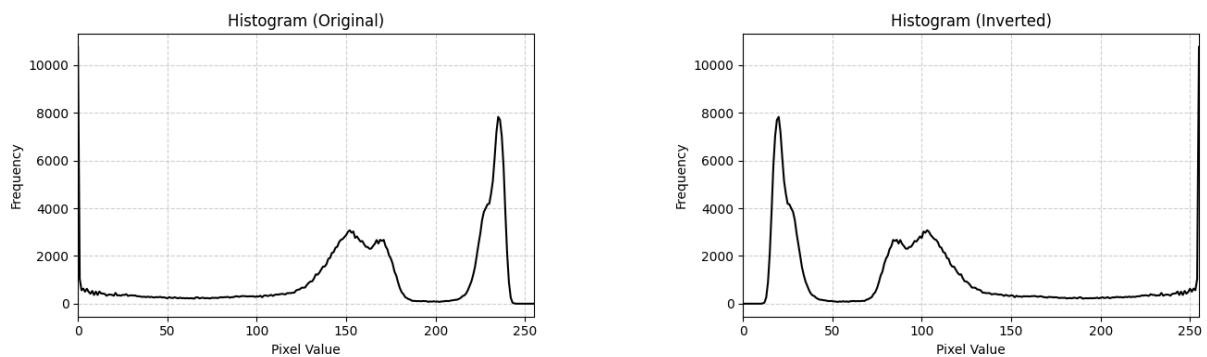


図4 載度反転前後のヒストグラム比較: 元画像のヒストグラムと反転後のヒストグラム

2.3 3) 空間フィルタリングによるノイズ除去

ノイズ除去にはメディアンフィルタ（サイズ 3×3 ）を採用した。選択の理由は、対象画像に含まれるノイズが、ランダムに白や黒の点が現れる「ごま塩ノイズ（インパルスノイズ）」であったため

である。平均化フィルタではノイズ成分が周囲にぼやけて広がってしまうのに対し、メディアンフィルタは注目画素周辺の中央値を出力するため、突発的な外れ値であるインパルスノイズを効果的に除去しつつ、物体の輪郭（エッジ）を比較的保存できる特性がある。適用結果として、エッジの鋭さを保ったままノイズのみがきれいに除去された画像が得られた。



図 5 ノイズ除去の比較: 処理前と処理後

2.4 4) Canny 法によるエッジ検出

ノイズ除去後の画像に対し、Canny 法によるエッジ検出を行った。本手法では 2 つのしきい値 (Threshold 1, Threshold 2) の設定が重要となるため、値を変化させて比較を行った。しきい値を極端に低くすると、壁面の微細な模様やノイズ、空の模様まで過剰に検出された。逆に高くしすぎると、建物の主要な輪郭線が途切れてしまった。試行錯誤の結果、**Threshold 1 = 75, Threshold 2 = 92** に設定した際、ノイズを拾わず、かつ建物の形状を正確に抽出できる最適な結果が得られた。ただし、まだより最適な数値が存在する可能性はあるものの、今回はここで妥協した。違いが分かりづらい部分があるため、今後はエッジ検出の精度を定量的に評価する調整方法を調査したい。

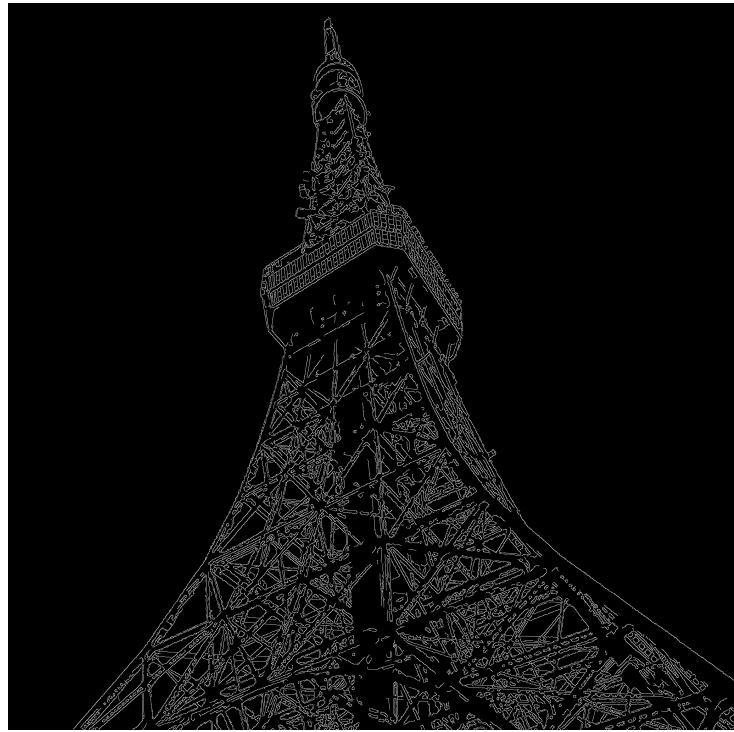


図 6 Canny 法によるエッジ検出結果

付録: プログラムリスト

本レポートの課題 2 で使用した Python プログラムを以下に示す。

```
1 # モジュールのインポート
2 import cv2
3 import numpy as np
4 import matplotlib.pyplot as plt
5 from google.colab import drive
6
7 # ドライブのマウント
8 drive.mount('/content/drive')
9
10 # 画像ディレクトリパス
# 一般的に定数は大文字、または短縮形で書くことが多いです
12 IMG_DIR = '/content/drive/MyDrive/img2025/image/'
13
14 def show_result(src, dst, title_src='Original', title_dst='Result'):
15     """
16     変換前後の画像を比較表示する関数
17     src: 元画像 (Source)
18     dst: 変換後画像 (Destination)
19     """
20     plt.figure(figsize=(10, 5))
21
22     # 元画像
```

```

23     plt.subplot(1, 2, 1)
24     plt.title(title_src)
25     plt.imshow(src, cmap='gray', vmin=0, vmax=255)
26     plt.axis('off')
27
28     # 結果画像
29     plt.subplot(1, 2, 2)
30     plt.title(title_dst)
31     plt.imshow(dst, cmap='gray', vmin=0, vmax=255)
32     plt.axis('off')
33
34     plt.tight_layout()
35     plt.show()
36
37 def plot_hist(img, title='Histogram'):
38     """
39     画像のヒストグラムを表示する関数
40     """
41     # OpenCVでヒストグラムを算出
42     hist = cv2.calcHist([img], [0], None, [256], [0, 256])
43
44     plt.figure(figsize=(6, 4))
45     plt.title(title)
46     plt.plot(hist, color='black')
47     plt.xlabel('Pixel Value')
48     plt.ylabel('Frequency')
49     plt.xlim([0, 255])
50     plt.grid(True, linestyle='--', alpha=0.6) # グリッドを少し見やすく調整
51     plt.tight_layout()
52     plt.show()
53
54 # -----
55 # 課題1: 線形変換による濃度変換
56 # -----
57
58 # 画像読み込み (Source)
59 src = cv2.imread(IMG_DIR + 'gray_image.png', cv2.IMREAD_GRAYSCALE)
60
61 # 出力画像 (Destination) の初期化
62 h, w = src.shape
63 dst = np.zeros((h, w), dtype=np.uint8)
64
65 # 画素ごとの処理
66 for y in range(h):
67     for x in range(w):
68         val = src[y, x]
69
70         # 条件分岐による線形変換
71         if val <= 200:

```

```

72         new_val = 1.2 * val + 15
73     else:
74         new_val = 255
75
76     dst[y, x] = int(new_val)
77
78 # 結果表示
79 show_result(src, dst, title_dst='Transformed Image')
80
81 # ヒストグラム表示
82 plot_hist(src, 'Histogram (Original)')
83 plot_hist(dst, 'Histogram (Transformed)')
84
85 # 保存
86 cv2.imwrite(IMG_DIR + 'transformed_image.png', dst)
87
88 # -----
89 # 課題2: 輝度反転
90 #
91
92 # 画像読み込み
93 src = cv2.imread(IMG_DIR + 'building.png', cv2.IMREAD_GRAYSCALE)
94
95 # 線形変換 (alpha=-1, beta=255 で反転)
96 dst = cv2.convertScaleAbs(src, alpha=-1, beta=255)
97
98 # 結果表示
99 show_result(src, dst, title_dst='Inverted Image')
100
101 # ヒストグラム表示
102 plot_hist(src, 'Histogram (Original)')
103 plot_hist(dst, 'Histogram (Inverted)')
104
105 # 保存
106 cv2.imwrite(IMG_DIR + 'inverted_image.png', dst)
107
108 # -----
109 # 課題3: メディアンフィルタによるノイズ除去
110 #
111
112 # 画像読み込み
113 src = cv2.imread(IMG_DIR + 'noisy_image.png', cv2.IMREAD_GRAYSCALE)
114
115 # フィルタ適用 (ksize: Kernel Size)
116 ksize = 5
117 dst = cv2.medianBlur(src, ksize)
118
119 # 結果表示
120 show_result(src, dst, title_src='Noisy Image', title_dst=f'Denoised (Median

```

```

121     '{ksize}x{ksize})')
122
123 # 保存
124 cv2.imwrite(IMG_DIR + 'denoised_image.png', dst)
125
126 # -----
127 # 課題4: Canny法によるエッジ検出
128
129 # ノイズ除去済み画像の読み込み
130 src = cv2.imread(IMG_DIR + 'denoised_image.png', cv2.IMREAD_GRAYSCALE)
131
132 # しきい値設定 (Threshold)
133 th1 = 75
134 th2 = 92
135
136 # エッジ検出
137 edges = cv2.Canny(src, th1, th2)
138
139 # 結果表示
140 plt.figure(figsize=(6, 6))
141 plt.imshow(edges, cmap='gray')
142 plt.title(f'Canny Edges (th1={th1}, th2={th2})')
143 plt.axis('off')
144 plt.show()
145
146 # 保存
147 cv2.imwrite(IMG_DIR + 'canny_edge_image.png', edges)

```

Listing 1 画像処理プログラム