画像処理・画像処理工学 レポート課題 1

クラス: ○○　番号: ○○　氏名: ○○○○

2025 年 11 月 21 日

1　問題 1

1.1　複合レンズによる像の位置と大きさ

1.1.1　計算・導出過程

■凸レンズ A による中間像　レンズの公式 $\frac{1}{a}+\frac{1}{b}=\frac{1}{f}$ より，中間像の位置 b_A を求める．物体距離 $a_A = 16\,\mathrm{cm}$，焦点距離 $f_A = 12\,\mathrm{cm}$ を代入すると以下の通りとなる．

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

$$\frac{1}{b_A}=\frac{1}{12}-\frac{1}{16}=\frac{4}{48}-\frac{3}{48}=\frac{1}{48}$$

$$b_A = 48\,\mathrm{cm}$$

倍率 m_A は以下の通りである．

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

物体の大きさ $h = 2.0\,\mathrm{cm}$ より，中間像の大きさ h'_A は次式となる．

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

■凸レンズ B による最終像　中間像を凸レンズ B の物体とみなす．AB 間の距離は 63 cm であるため，凸レンズ B における物体距離 a_B は以下となる．

$$a_B = 63 - b_A = 63 - 48 = 15\,\mathrm{cm}$$

凸レンズ B の焦点距離 $f_B = 10\,\mathrm{cm}$ より，最終像の位置 b_B を求める．

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

$$\frac{1}{b_B}=\frac{1}{10}-\frac{1}{15}=\frac{3}{30}-\frac{2}{30}=\frac{1}{30}$$

$$b_B = 30\,\mathrm{cm}$$

倍率 m_B は以下の通りである．

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

中間像の大きさ $h'_A = 6.0\,\mathrm{cm}$ より，最終像の大きさ h'_B は次式となる．

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

結果

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

1.2　各デバイスの ppi 計算と比較

1.2.1　計算・導出過程

ppi は，画面の画素数 (w_p, h_p) および対角線長 d_i (インチ) より次式で定義される．

$$\text{ppi} = \frac{\sqrt{w_p^2 + h_p^2}}{d_i}$$

■A. Google Pixel 10　$(w_p = 1080,\ h_p = 2424,\ d_i = 6.3)$

$$\text{ppi} = \frac{\sqrt{1080^2 + 2424^2}}{6.3} = \frac{\sqrt{1166400 + 5875776}}{6.3} \approx \frac{2653.7}{6.3} \approx 421.22$$

■B. iPad (A16)　$(w_p = 2360,\ h_p = 1640,\ d_i = 10.9)$

$$\text{ppi} = \frac{\sqrt{2360^2 + 1640^2}}{10.9} = \frac{\sqrt{5569600 + 2689600}}{10.9} \approx \frac{2873.9}{10.9} \approx 263.66$$

■C. EIZO EV2740S　$(w_p = 3840,\ h_p = 2160,\ d_i = 27.0)$

$$\text{ppi} = \frac{\sqrt{3840^2 + 2160^2}}{27.0} = \frac{\sqrt{14745600 + 4665600}}{27.0} \approx \frac{4405.8}{27.0} \approx 163.18$$

結果

各デバイスの ppi（整数値）を以下に示す．

デバイス名	ppi
Google Pixel 10	421 ppi
iPad (A16)	264 ppi
EIZO EV2740S	163 ppi

1.3　8 近傍鮮鋭化フィルタの導出

1.3.1　導出過程

鮮鋭化フィルタ W_S は，単位オペレータ W_I とラプラシアンフィルタ W_L を用いて次式で表される．

$$W_S = W_I - W_L$$

以下，ラプラシアンフィルタ W_L を 2 次差分より導出する．

■各方向の 2 次差分　注目画素を $f(i, j)$ とし，隣接画素との差分（1 次微分）の差分により 2 次微分を近似する．

- **x 方向 2 次差分 (f_{xx})**

$$f_{xx} = f(i+1, j) - 2f(i, j) + f(i-1, j)$$

- **y 方向 2 次差分 (f_{yy})**

$$f_{yy} = f(i, j+1) - 2f(i,j) + f(i, j-1)$$

- **斜め方向 1 (左上-右下) 2 次差分 (f_{d1})**

$$f_{d1} = f(i+1, j+1) - 2f(i,j) + f(i-1, j-1)$$

- **斜め方向 2 (右上-左下) 2 次差分 (f_{d2})**

$$f_{d2} = f(i-1, j+1) - 2f(i,j) + f(i+1, j-1)$$

■8 近傍ラプラシアンフィルタ (W_L)　8 近傍ラプラシアン $\nabla^2 f$ は，上記 4 方向の 2 次差分の総和と定義される．

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

式を整理すると，注目画素 $f(i,j)$ の係数は-8，周囲 8 画素の係数は 1 となるため，以下のカーネルが得られる．

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

■8 近傍鮮鋭化フィルタ (W_S)　$W_S = W_I - W_L$ より計算する．

$$W_S = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

結果

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

1.4　Sobel フィルタによるエッジ解析

1.4.1　計算・導出過程

Sobel オペレータを S_x, S_y とし，エッジ強度 G と方向 θ を以下で定義する．

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

■注目画素 A

$$G_x = 200, \quad G_y = 200$$

$$G_A = \sqrt{200^2 + 200^2} \approx 282.84, \quad \theta_A = \arctan(1) = 45^\circ$$

■注目画素 B

$$G_x = 400, \quad G_y = -400$$

$$G_B = \sqrt{400^2 + (-400)^2} \approx 565.69, \quad \theta_B = \arctan(-1) = -45^\circ$$

結果と考察

- **エッジ強度**: 画素 B の強度は A の約 2 倍であり，B 周辺の方がより急峻な輝度変化を有している．
- **エッジ方向**: A は 45°，B は $-45°$ 方向へのエッジであり，傾きが直交に近い関係にある．

2　問題 2

2.1　線形変換関数による濃度変換

■理論　図 A-4 より，入力濃度 f と出力濃度 g の関係式を導出する．グラフは点 $(0, 15)$ と 点 $(200, 255)$ を結ぶ直線，および $f > 200$ における定数値である．直線の傾き a は以下の通りとなる．

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

したがって，求める線形変換関数は次式で表される．

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

■結果　「gray_image.png」に対し上記変換を適用した結果を図 1 に示す．

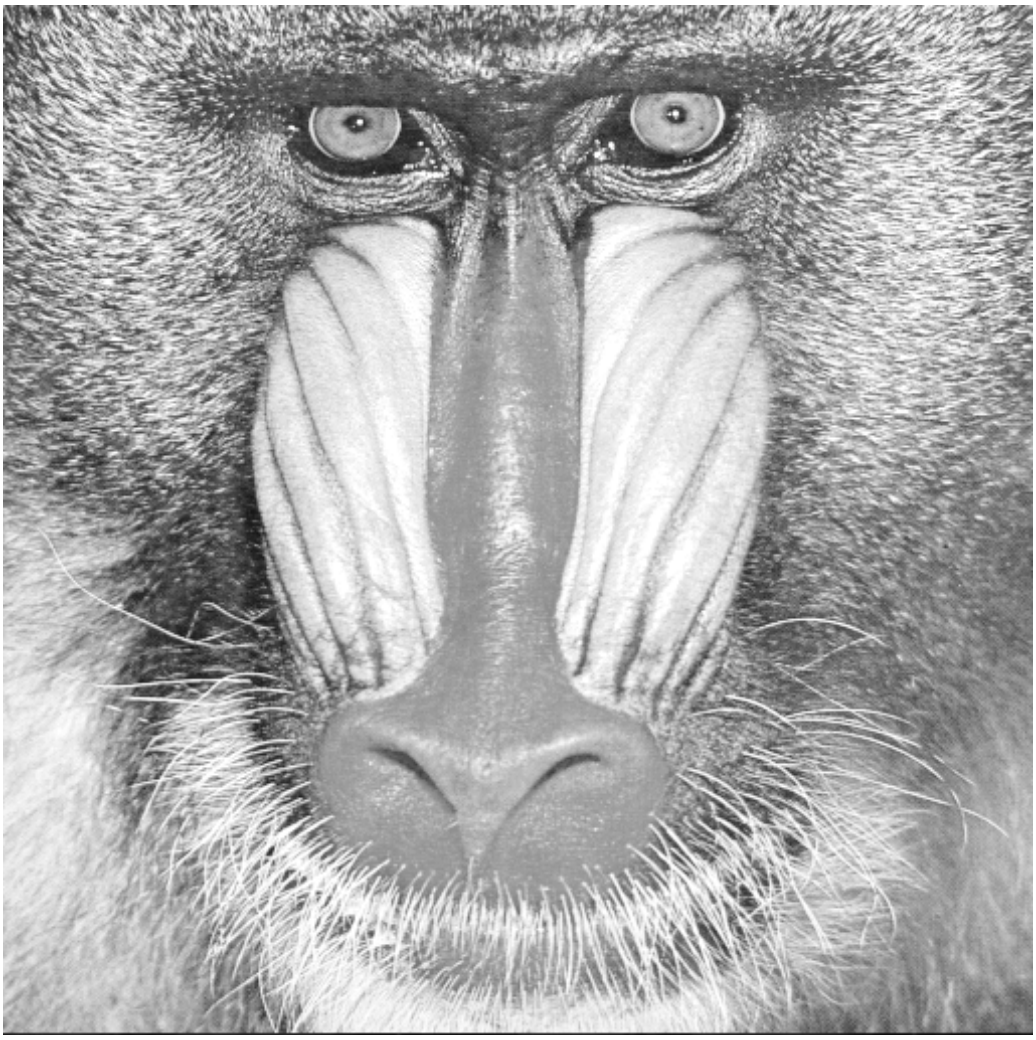

図 1　線形変換の結果 (左: 元画像, 右: 変換後画像) および各ヒストグラム

■考察　ヒストグラムを確認すると，元画像では 0 付近にあった低輝度の画素が，変換後には 15 付近にシフトしている．また，輝度分布の幅が広がり（コントラストが強調され），全体的に画像が明

るくなっている．これは，変換式の傾きが 1.2（増幅）であり，かつ切片 15（オフセット）が加算されていることと整合する．

2.2　輝度反転

■理論　8bit 濃淡画像の輝度反転における濃度変換関数は，入力 f が 0 のとき出力 g が 255，入力 255 のとき出力 0 となる直線である．

$$g(f) = -f + 255 \quad (= 255 - f)$$

グラフは傾き-1，切片 255 の右下がりの直線となる．

■結果　「building.png」に対し輝度反転を適用した結果を図 2 に示す．

図 2　輝度反転の結果とヒストグラム

■考察　画像の明暗が完全に反転している．ヒストグラムにおいても，分布の形状が左右（高輝度・低輝度）で反転していることが確認できる．これは変換式 $g = 255 - f$ により，画素値の大小関係が逆転した結果である．

2.3　空間フィルタリングによるノイズ除去

■使用したフィルタ

- **種類**: メディアンフィルタ (Median Filter)
- **サイズ**: 5×5
- **選定理由**: 元画像「noisy_image.png」には，白黒の斑点状のノイズ（ごま塩ノイズ／インパルスノイズ）が確認された．メディアンフィルタは注目画素周辺の中央値を採用するため，平均化フィルタ等と比較して，このような極端な値を持つ外れ値（ノイズ）の除去能力が高く，かつエッジを保存しやすい特性があるため採用した．

■結果　ノイズ除去の結果を図 3 に示す．

図 3　メディアンフィルタによるノイズ除去結果

■考察　処理後の画像では，建物の輪郭などのエッジ情報を大きく損なうことなく，ごま塩ノイズが良好に除去されている．これにより，後段のエッジ検出処理に適した画像が得られたといえる．

2.4　Canny 法によるエッジ検出

■処理概要　前項でノイズ除去を行った画像に対し，Canny のエッジ検出アルゴリズムを適用した．物体の輪郭を適切に抽出するため，ヒステリシスしきい値処理における 2 つのしきい値を調整した．

■結果

- **設定したしきい値**: threshold1 = 100, threshold2 = 130

エッジ検出の結果を図 4 に示す．

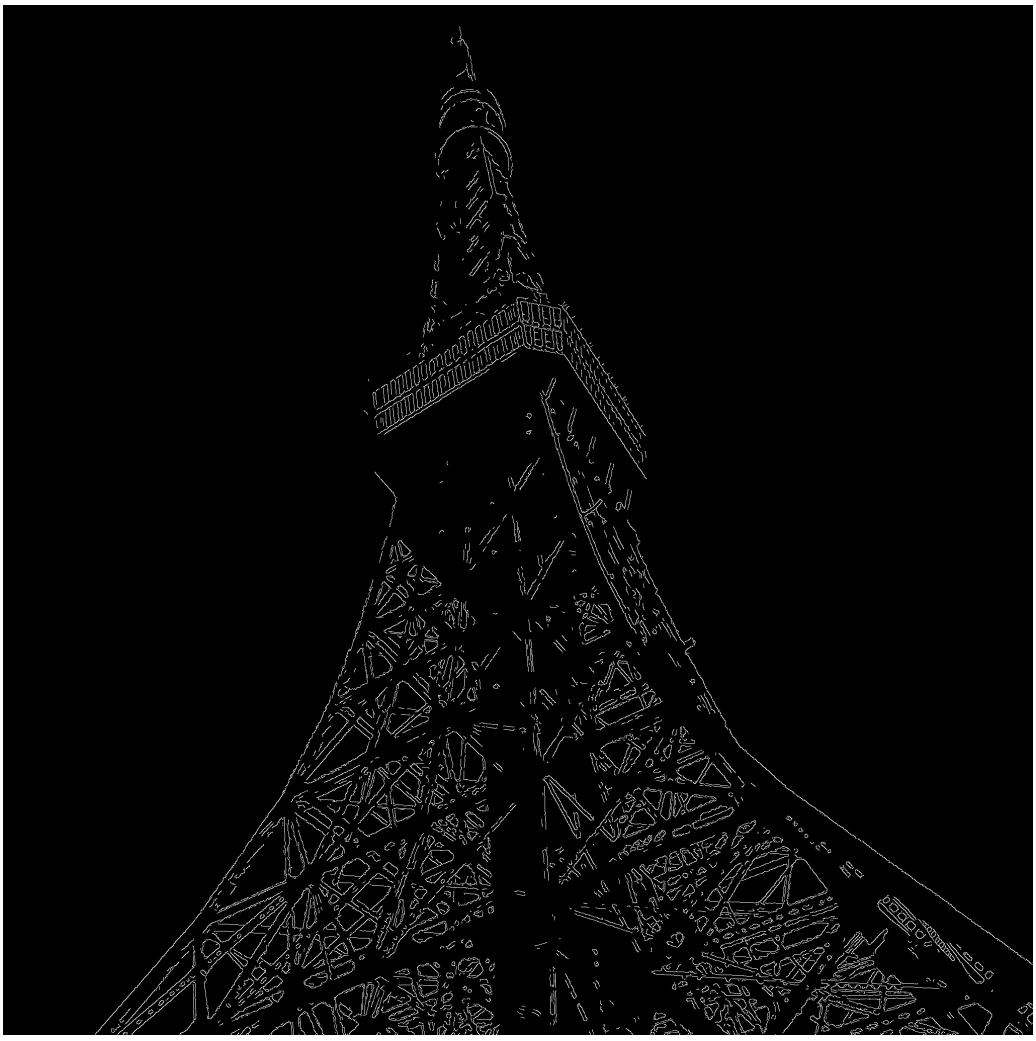

図 4　Canny 法によるエッジ検出結果

■考察　適切なしきい値設定を行うことで，建物の主要な輪郭線や窓の枠組みが明瞭に検出された．しきい値をこれより低く設定するとノイズによる微細な線が多数検出され，高く設定すると建物の輪郭が途切れる現象が確認されたため，上記の値が最適であると判断した．

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.

```python
# モジュールのインポート
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive

# ドライブのマウント
drive.mount('/content/drive')

# 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
IMG_DIR = '/content/drive/MyDrive/img2025/image/'

def show_result(src, dst, title_src='Original', title_dst='Result'):
    """
    変換前後の画像を比較表示する関数
    src: 元画像 (Source)
    dst: 変換後画像 (Destination)
    """
    plt.figure(figsize=(10, 5))

    # 元画像
    plt.subplot(1, 2, 1)
    plt.title(title_src)
    plt.imshow(src, cmap='gray', vmin=0, vmax=255)
    plt.axis('off')

    # 結果画像
    plt.subplot(1, 2, 2)
    plt.title(title_dst)
    plt.imshow(dst, cmap='gray', vmin=0, vmax=255)
    plt.axis('off')

    plt.tight_layout()
    plt.show()

def plot_hist(img, title='Histogram'):
    """
    画像のヒストグラムを表示する関数
    """
    # OpenCVでヒストグラムを算出
    hist = cv2.calcHist([img], [0], None, [256], [0, 256])

    plt.figure(figsize=(6, 4))
```

```
    plt.title(title)
    plt.plot(hist, color='black')
    plt.xlabel('Pixel Value')
    plt.ylabel('Frequency')
    plt.xlim([0, 255])
    plt.grid(True, linestyle='--', alpha=0.6) # グリッドを少し見やすく調整
    plt.tight_layout()
    plt.show()

# ------------------------------------------------------------------
# 課題1: 線形変換による濃度変換
# ------------------------------------------------------------------

# 画像読み込み (Source)
src = cv2.imread(IMG_DIR + 'gray_image.png', cv2.IMREAD_GRAYSCALE)

# 出力画像 (Destination) の初期化
h, w = src.shape
dst = np.zeros((h, w), dtype=np.uint8)

# 画素ごとの処理
for y in range(h):
    for x in range(w):
        val = src[y, x]

        # 条件分岐による線形変換
        if val <= 200:
            new_val = 1.2 * val + 15
        else:
            new_val = 255

        dst[y, x] = int(new_val)

# 結果表示
show_result(src, dst, title_dst='Transformed Image')

# ヒストグラム表示
plot_hist(src, 'Histogram (Original)')
plot_hist(dst, 'Histogram (Transformed)')

# 保存
cv2.imwrite(IMG_DIR + 'transformed_image.png', dst)

# ------------------------------------------------------------------
# 課題2: 輝度反転
# ------------------------------------------------------------------

# 画像読み込み
src = cv2.imread(IMG_DIR + 'building.png', cv2.IMREAD_GRAYSCALE)
```

```python

# 線形変換 (alpha=-1, beta=255 で反転)
dst = cv2.convertScaleAbs(src, alpha=-1, beta=255)

# 結果表示
show_result(src, dst, title_dst='Inverted Image')

# ヒストグラム表示
plot_hist(src, 'Histogram (Original)')
plot_hist(dst, 'Histogram (Inverted)')

# 保存
cv2.imwrite(IMG_DIR + 'inverted_image.png', dst)

# ---------------------------------------------------------------
# 課題3: メディアンフィルタによるノイズ除去
# ---------------------------------------------------------------

# 画像読み込み
src = cv2.imread(IMG_DIR + 'noisy_image.png', cv2.IMREAD_GRAYSCALE)

# フィルタ適用 (ksize: Kernel Size)
ksize = 5
dst = cv2.medianBlur(src, ksize)

# 結果表示
show_result(src, dst, title_src='Noisy Image', title_dst=f'Denoised (Median
    {ksize}x{ksize})')

# 保存
cv2.imwrite(IMG_DIR + 'denoised_image.png', dst)

# ---------------------------------------------------------------
# 課題4: Canny法によるエッジ検出
# ---------------------------------------------------------------

# ノイズ除去済み画像の読み込み
src = cv2.imread(IMG_DIR + 'denoised_image.png', cv2.IMREAD_GRAYSCALE)

# しきい値設定 (Threshold)
th1 = 50
th2 = 85

# エッジ検出
edges = cv2.Canny(src, th1, th2)

# 結果表示
plt.figure(figsize=(6, 6))
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 (report_image_analysis3.py)