画像処理・画像処理工学 レポート課題 1

画像処理工学科　学生番号: 25X0000　氏名: 画像 太郎

2025 年 11 月 24 日

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 $a_A = 16\,\mathrm{cm}$、焦点距離 $f_A = 12\,\mathrm{cm}$ である。像の位置 b_A を求める。

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

移項して通分を行う。

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

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

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

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

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

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

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 $63\,\mathrm{cm}$ である。レンズ A による像（レンズ A の後方 $48\,\mathrm{cm}$）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。

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

レンズ B の焦点距離 $f_B = 10\,\mathrm{cm}$ 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、$b_B = 30\,\mathrm{cm}$ 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。

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

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。

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

1.2.1 Google Pixel 10

$d = 6.3$ 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。

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

1.2.2 iPad (A16)

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

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

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

1.2.3 EIZO EV2740S

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

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

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

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

画像の 2 次微分（ラプラシアン）$\nabla^2 f$ は、注目画素を中心とした近傍画素との差分によって近似できる。8 近傍の場合、水平 (x)、垂直 (y) に加え、対角 2 方向 (d_1, d_2) の 2 階差分の和をとる。

$$\nabla^2 f \approx \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial d_1^2} + \frac{\partial^2 f}{\partial d_2^2}$$

各方向の 2 階差分係数は $(1, -2, 1)$ であるため、中心画素 (i, j) における係数の総和は $-2 \times 4 = -8$ となる。周辺の 8 画素はそれぞれ係数 1 を持つ。よって、8 近傍ラプラシアンフィルタのカーネル H_L は以下のようになる。

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

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。元画像を表す単位インパルスフィルタ（恒等写像）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

図 A-3 より、各注目画素の近傍領域における画素値分布を読み取り、計算を行う。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。

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

1.4.1　注目画素 A (1,1) の計算

画素 A の中心画素値は 50 である。図より、この領域は左右で輝度が変化する縦エッジ（左が暗く右が明るい）であると読み取れるため、以下の画素値を仮定して計算する。

$$\text{近傍}_A = \begin{pmatrix} 0 & 50 & 100 \\ 0 & 50 & 100 \\ 0 & 50 & 100 \end{pmatrix}$$

x 方向微分 g_x の計算（畳み込み）:

$$g_x = (-1 \cdot 0) + (0 \cdot 50) + (1 \cdot 100) + (-2 \cdot 0) + (0 \cdot 50) + (2 \cdot 100) + (-1 \cdot 0) + (0 \cdot 50) + (1 \cdot 100)$$

$$g_x = 100 + 200 + 100 = 400$$

y 方向微分 g_y の計算: 上段と下段の画素値が等しいため、相殺されて 0 となる。

$$g_y = (-1 \cdot 0 - 2 \cdot 50 - 1 \cdot 100) + (1 \cdot 0 + 2 \cdot 50 + 1 \cdot 100) = -200 + 200 = 0$$

エッジ強度 G と方向 θ:

$$G_A = \sqrt{400^2 + 0^2} = 400$$

$$\theta_A = \arctan\left(\frac{0}{400}\right) = 0^\circ \quad \text{(垂直エッジ)}$$

1.4.2　注目画素 B (3,3) の計算

画素 B の中心画素値は 150 である。図より、この領域は上下で輝度が変化する横エッジ（上が暗く下が明るい）であると読み取れるため、以下の画素値を仮定する。

$$\text{近傍}_B = \begin{pmatrix} 50 & 50 & 50 \\ 150 & 150 & 150 \\ 250 & 250 & 250 \end{pmatrix}$$

x 方向微分 g_x の計算: 左列と右列の画素値が等しいため、相殺されて 0 となる。

$$g_x = (-1 \cdot 50 + 1 \cdot 50) + (-2 \cdot 150 + 2 \cdot 150) + (-1 \cdot 250 + 1 \cdot 250) = 0$$

y 方向微分 g_y の計算:

$$g_y = (-1 \cdot 50 - 2 \cdot 50 - 1 \cdot 50) + (0) + (1 \cdot 250 + 2 \cdot 250 + 1 \cdot 250)$$

$$g_y = (-50 - 100 - 50) + (250 + 500 + 250) = -200 + 1000 = 800$$

エッジ強度 G と方向 θ:

$$G_B = \sqrt{0^2 + 800^2} = 800$$

$$\theta_B = \arctan\left(\frac{800}{0}\right) = 90^\circ \quad \text{(水平エッジ)}$$

比較結果

- **強度**: 画素 B のエッジ強度 (800) は、画素 A(400) の 2 倍である。画素 B 周辺の方が輝度変化が急峻である。
- **方向**: 画素 A は 0°（垂直方向のエッジ)、画素 B は 90°（水平方向のエッジ）である。

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。

2.2　2) 輝度反転

画像のネガポジ反転を行うため、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が入れ替わる）していることが確認でき、理論通りの結果となった。

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。

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 法を用いてエッジ検出を行った。Canny 法は、ガウシアンフィルタによる平滑化、Sobel フィルタによる勾配計算、非極大抑制、およびヒステリシス閾値処理の 4 段階からなるアルゴリズムである。本課題では、ヒステリシス処理における 2 つの閾値（min_val,

max_val）の調整が重要であった。実験の結果、**min_val=100, max_val=200** に設定した際に、建物の主要な輪郭線が途切れることなく、かつ不要なテクスチャや微細なノイズを拾わずに検出できた。この結果から、Canny 法はパラメータ調整により、目的とする構造的特徴のみを抽出する能力に優れていることが確認できた。

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)

# ---------------------------------------------------------------
```

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

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

# 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 = 30
th2 = 80

# 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 (report_image_analysis3.py)