画像処理・画像処理工学 レポート課題 1 解答

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　$(w_p = 1080,\ h_p = 2424,\ d_i = 6.3)$

$$\text{ppi} = \frac{\sqrt{1080^2 + 2424^2}}{6.3} = \frac{\sqrt{1166400 + 5875776}}{6.3} = \frac{\sqrt{7042176}}{6.3} \approx \frac{2653.7}{6.3} \approx 421.22$$

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

$$\text{ppi} = \frac{\sqrt{2360^2 + 1640^2}}{10.9} = \frac{\sqrt{5569600 + 2689600}}{10.9} = \frac{\sqrt{8259200}}{10.9} \approx \frac{2873.9}{10.9} \approx 263.66$$

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

$$\text{ppi} = \frac{\sqrt{3840^2 + 2160^2}}{27.0} = \frac{\sqrt{14745600 + 4665600}}{27.0} = \frac{\sqrt{19411200}}{27.0} \approx \frac{4405.8}{27.0} \approx 163.18$$

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})**

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

- **y 方向 2 次差分 (f_{yy})**

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

- **斜め方向 1 (左上-右下) 2 次差分 (f_{d1})**

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

- **斜め方向 2 (右上-左下) 2 次差分 (f_{d2})**

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

■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．

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

結果

導出された 8 近傍鮮鋭化フィルタは以下の通り．

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

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 オペレータを以下とする．

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

エッジ強度 G と方向 θ の定義式は以下の通りである．

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

■注目画素 A

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

- 一次微分:
  $G_x = (100 \cdot 1 + 100 \cdot 2 + 150 \cdot 1) - (50 \cdot 1 + 50 \cdot 2 + 100 \cdot 1) = 450 - 250 = 200$
  $G_y = (100 \cdot 1 + 100 \cdot 2 + 150 \cdot 1) - (50 \cdot 1 + 50 \cdot 2 + 100 \cdot 1) = 450 - 250 = 200$
- エッジ強度:
  $G_A = \sqrt{200^2 + 200^2} = 200\sqrt{2} \approx 282.84$
- エッジ方向:
  $\theta_A = \arctan\left(\frac{200}{200}\right) = \arctan(1) = 45^\circ$

■注目画素 B

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

- 一次微分:
  $G_x = (250 \cdot 1 + 250 \cdot 2 + 200 \cdot 1) - (150 \cdot 1 + 150 \cdot 2 + 100 \cdot 1) = 950 - 550 = 400$
  $G_y = (100 \cdot 1 + 100 \cdot 2 + 200 \cdot 1) - (150 \cdot 1 + 250 \cdot 2 + 250 \cdot 1) = 500 - 900 = -400$
- エッジ強度:
  $G_B = \sqrt{400^2 + (-400)^2} = 400\sqrt{2} \approx 565.69$
- エッジ方向:
  $\theta_B = \arctan\left(\frac{-400}{400}\right) = \arctan(-1) = -45^\circ$

結果と考察

計算結果の比較を以下に示す.

	注目画素 A	注目画素 B
x 方向微分 G_x	200	400
y 方向微分 G_y	200	-400
エッジ強度 G	約 282.8	約 565.7
エッジ方向 θ	45°	-45°

両者を比較すると以下の特徴が認められる.

- **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 は左下から右上 (45°)，B は左上から右下 (-45°) 方向へのエッジであり，傾きが直交に近い関係にある.

付録: 実験で使用した Python コード

以下は課題 1〜4 で使用したプログラムのリストである.

Listing 1　report_image_analysis2 用 Python スクリプト

```python
# report_image_analysis2.py
# Generated from report_image_analysis2.ipynb
# NOTE: Includes google.colab mount lines; remove if running locally.

# 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')

# 共通のディレクトリパス
common_path = '/content/drive/MyDrive/img2025/image/'


# 変換前後の画像を並べて表示する関数
def show_images(img1, img2, title1='Original', title2='Result', cmap='gray', vmin=0, vmax=255):
    plt.figure(figsize=(10, 5))

    # 変換前の画像を表示
    plt.subplot(1, 2, 1)
    plt.title(title1)
    plt.imshow(img1, cmap=cmap, vmin=vmin, vmax=vmax)
    plt.axis('off')

    # 変換後の画像を表示
    plt.subplot(1, 2, 2)
    plt.title(title2)
    plt.imshow(img2, cmap=cmap, vmin=vmin, vmax=vmax)
    plt.axis('off')

    plt.tight_layout()
    plt.show()


# ヒストグラムを描画する関数
def plot_grayscale_histogram(image, title='Histogram'):
    # 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=(6, 4))
    plt.title(title)
    plt.plot(hist, color='black')
    plt.xlabel('Pixel Value')
    plt.ylabel('Frequency')
    plt.xlim([0, 255])
    plt.grid(True)
    plt.tight_layout()
    plt.show()


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

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

# 出力用配列の初期化
H, W = gray_img.shape
adjusted_img = np.zeros((H, W), dtype=np.uint8)

# 画素ごとの濃度変換処理
for y in range(H):
    for x in range(W):
        f = gray_img[y, x]

        # 変換式: f <= 200 なら 1.2f + 15, それ以外は 255
        if f <= 200:
            g = 1.2 * f + 15
        else:
            g = 255

        adjusted_img[y, x] = int(g)

# 結果画像を表示（関数を使用）
show_images(gray_img, adjusted_img, 'Original Image', 'Transformed Image')

# ヒストグラムを表示（関数を使用）
plot_grayscale_histogram(gray_img, 'Original Histogram')
plot_grayscale_histogram(adjusted_img, 'Transformed Histogram')

# 画像保存
cv2.imwrite('transformed_image.png', adjusted_img)


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

```python
# ------------------------------------------------------------

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

# 線形変換関数による輝度変換（反転: alpha=-1, beta=255）
adjusted_img = cv2.convertScaleAbs(gray_img, alpha=-1, beta=255)

# 結果画像を表示
show_images(gray_img, adjusted_img, 'Original Image', 'Inverted Image')

# ヒストグラムを表示
plot_grayscale_histogram(gray_img, 'Original Histogram')
plot_grayscale_histogram(adjusted_img, 'Inverted Histogram')

# 画像保存
cv2.imwrite('inverted_image.png', adjusted_img)


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

# 画像の読み込み
noisy_img = cv2.imread(common_path + 'noisy_image.png', cv2.IMREAD_GRAYSCALE)

# メディアンフィルタの適用（サイズ5）
median_img = cv2.medianBlur(noisy_img, 5)

# 実行結果の表示
show_images(noisy_img, median_img, 'Noisy Image', 'Denoised Image (Median 5x5)')

# 画像保存
cv2.imwrite(common_path + 'denoised_image.png', median_img)


# ------------------------------------------------------------------
# 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(common_path + 'denoised_image.png', cv2.IMREAD_GRAYSCALE)

# しきい値の設定
threshold1 = 100
threshold2 = 130

# Cannyアルゴリズムを使用してエッジ検出を行う
canny_edges = cv2.Canny(src, threshold1, threshold2)
```

```
141
142 # 実行結果の画像を表示する
143 plt.figure(figsize=(6, 6))
144 plt.imshow(canny_edges, cmap='gray', vmin=0, vmax=255)
145 plt.title(f'Canny Edges (th1={threshold1}, th2={threshold2})')
146 plt.axis('off')
147 plt.show()
148
149 # 画像保存
150 cv2.imwrite(common_path + 'canny_edge_image.png', canny_edges)
```