

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 $a_A = 16 \text{ cm}$, 焦点距離 $f_A = 12 \text{ cm}$ を代入すると以下の通りとなる.

$$\begin{aligned}\frac{1}{16} + \frac{1}{b_A} &= \frac{1}{12} \\ \frac{1}{b_A} &= \frac{1}{12} - \frac{1}{16} = \frac{4}{48} - \frac{3}{48} = \frac{1}{48} \\ b_A &= 48 \text{ cm}\end{aligned}$$

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

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

■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.

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

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

$$\begin{aligned}\frac{1}{15} + \frac{1}{b_B} &= \frac{1}{10} \\ \frac{1}{b_B} &= \frac{1}{10} - \frac{1}{15} = \frac{3}{30} - \frac{2}{30} = \frac{1}{30} \\ b_B &= 30 \text{ cm}\end{aligned}$$

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

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

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.

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 Python スクリプト

```
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
11 # 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
    {ksize}x{ksize})')
121
122 # 保存
123 cv2.imwrite(IMG_DIR + 'denoised_image.png', dst)
124
125 # -----
126 # 課題4: Canny法によるエッジ検出
127 # -----
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 = 100
134 th2 = 130
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)
```