Lab2 实验报告

姓名：陈翎玺 学号：523030910039 班级：电院 2302

1 实验概览

本次实验主要学习了图像边缘的提取和 Canny 算法的实现。

图像的大部分信息都存在于图像的边缘中, 主要表现为图像局部特征的不连续性, 即图像中灰度变化比较剧烈的地方。因此定义图象的边缘为图象局部区域亮度变化显著的部分，该区域的灰度剖面一般可以看作是一个阶跃，即灰度值在很小的区域内急剧的变化。

Canny 算法是一种经典的边缘检测算法，由 John Canny 在 1986 年提出，主要用于图像处理中的边缘提取。它的设计目标是最大限度地减少噪声的干扰，同时保持边缘的精确性。广泛用于目标检测、图像分割、物体识别等领域，是计算机视觉中的重要工具。

Canny 算法的优点包括：

1. 抗噪性强：高斯滤波减少了噪声的干扰。
2. 检测精确：通过梯度计算和非极大值抑制，确保边缘精度高。
3. 双阈值处理：增强了边缘检测的灵活性。

Canny 算法的主要步骤包括灰度化，高斯滤波，梯度计算，非极大值抑制，双阈值处理和边缘连接。其具体步骤将在2中展开。

2 练习题的解决思路

2.1 灰度化

通常摄像机获取的是彩色图像，而检测的首要步骤是进行灰度化，以 RGB 格式彩图为例，一般的灰度化方法有两种：

方法一：

$$Gary = (R + G + B)/3$$

方法二：

$$Gary = 0.299R + 0.587G + 0.114B$$

(参数考虑人眼的生理特点)

考虑本题已给出图片，使用 opencv 的 imread(..., cv2.IMREAD_GRAYSCALE) 直接读取灰度图即可。

2.2 高斯滤波

为了降低图像中的噪声，首先对图像进行高斯滤波。通过卷积操作平滑图像，以减少后续边缘检测时的误检。

图像高斯滤波的实现可以用两个一维高斯核分别两次加权实现，也可以通过一个二维高斯核一次卷积实现离散化的一维高斯函数与二维高斯函数如下：

$$K = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

离散一维高斯函数

$$K = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

离散二维高斯函数

具体而言，高斯滤波器核的生成方程式由下式给出：

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp(-\frac{(i-(k+1))^2 + (j-(k+1)^2)}{2\sigma^2}); 1 \le i, j \le (2k+1)$$

本次实验取 $k = 1, \sigma = 1.4$，得到对应的高斯滤波器核（归一化后）的结果为

$$H = \begin{bmatrix} 0.0924 & 0.1192 & 0.0924 \\ 0.1192 & 0.1538 & 0.1192 \\ 0.0924 & 0.1192 & 0.0924 \end{bmatrix}$$

若图像中一个 3x3 的窗口为 A，要滤波的像素点为 e，则经过高斯滤波之后，像素点 e 的亮度值为：

$$e = H*A = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{22} & H_{32} & H_{33} \end{bmatrix} * \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = sum\left(\begin{bmatrix} a \times H_{11} & b \times H_{12} & c \times H_{13} \\ d \times H_{21} & e \times H_{22} & f \times H_{23} \\ g \times H_{22} & h \times H_{32} & i \times H_{33} \end{bmatrix} \right)$$

其中 * 为卷积符号，sum 表示矩阵中所有元素相加求和

2.3 梯度计算

关于图像灰度值得梯度可使用一阶有限差分来进行近似，这样就可以得图像在 x 和 y 方向上偏导数的两个矩阵。常用的梯度算子有如下几种（本次实验使用 Sobel 算子）：

2.3.1 Roberts 算子

$$s_x = \frac{1}{2}\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad s_y = \frac{1}{2}\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

定义窗口

$$A = \begin{bmatrix} f_{i,j} & f_{i,j+1} \\ f_{i+1,j} & f_{i+1,j+1} \end{bmatrix}$$

对应的梯度矩阵为

$$G_{x,i,j} = sum\left(s_x * A\right)$$

$$G_{y,i,j} = sum\left(s_y * A\right)$$

$$G_{i,j} = |G_{x,i,j}| + |G_{y,i,j}|$$

2.3.2 Sobel 算子

$$s_x = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad s_y = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & 1 \end{bmatrix}$$

定义窗口

$$A = \begin{bmatrix} f_{i-1,j-1} & f_{i-1,j} & f_{i-1,j+1} \\ f_{i,j-1} & f_{i,j} & f_{i,j+1} \\ f_{i+1,j-1} & f_{i+1,j} & f_{i+1,j+1} \end{bmatrix}$$

对应的梯度矩阵为

$$G_{x,i,j} = sum\left(s_x * A\right)$$

$$G_{y,i,j} = sum\left(s_y * A\right)$$

$$G_{i,j} = \sqrt{G_{x,i,j}^2 + G_{y,i,j}^2}$$

2.3.3 Prewitt 算子

$$s_x = \frac{1}{6}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad s_y = \frac{1}{6}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & 1 \end{bmatrix}$$

定义窗口

$$A = \begin{bmatrix} f_{i-1,j-1} & f_{i-1,j} & f_{i-1,j+1} \\ f_{i,j-1} & f_{i,j} & f_{i,j+1} \\ f_{i+1,j-1} & f_{i+1,j} & f_{i+1,j+1} \end{bmatrix}$$

对应的梯度矩阵为

$$G_{x,i,j} = sum\left(s_x * A\right)$$

$$G_{y,i,j} = sum\left(s_y * A\right)$$

$$G_{i,j} = \sqrt{G_{x,i,j}^2 + G_{y,i,j}^2}$$

2.3.4 Canny 算子

$$s_x = \frac{1}{4}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \quad s_y = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

定义窗口

$$A = \begin{bmatrix} f_{i,j} & f_{i,j+1} \\ f_{i+1,j} & f_{i+1,j+1} \end{bmatrix}$$

对应的梯度矩阵为

$$G_{x,i,j} = sum\left(s_x * A\right)$$

$$G_{y,i,j} = sum\left(s_y * A\right)$$

$$G_{i,j} = \sqrt{G_{x,i,j}^2 + G_{y,i,j}^2}$$

2.4 非极大值抑制

图像梯度幅值矩阵中的元素值越大，说明图像中该点的梯度值越大。在 Canny 算法中，非极大值抑制是进行边缘检测的重要步骤，是指寻找像素点局部最大值，将非极大值点所对应的灰度值置为 0，从而可以剔除掉一大部分非边缘点。

如图：

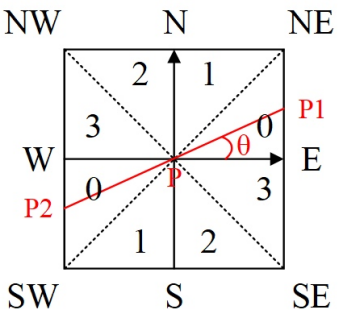

图 1: 非极大值抑制示意图

由2.3.2得到的 $G_{x,i,j}$(向右) 和 $G_{y,i,j}$（向上）梯度，可以计算得该点的梯度方向。红色线条方向为 C 点的梯度方向，C 点局部的最大值则分布在这条线上。

由于像素是离散存储的，梯度方向不一定恰好命中邻域中的点（如本图中的 P1，P2）。为了得到对应点的像素值，我们可以通过插值的方法计算 P1，P2 的值。

对于图中这种情况,

$$|G_{y,i,j}| \leq |G_{x,i,j}|, G_{x,i,j} \times G_{y,i,j} > 0$$

计算方法如下所示如下所示：

$$k = G_{y,i,j}/G_{x,i,j}$$

$$P1 = NE \times k + E \times (1 - k)$$

$$P2 = SW \times k + E \times (1 - k)$$

其中，$NW = G_{i-1,j-1}, N = G_{i-1,j}, \ldots, SE = G_{i+1,j+1}$ 如果 $G_{i,j} \geq P1$ 且 $G_{i,j} \geq P2$，那么说明点 (i, j) 是梯度极大值点，保留。否则丢弃它。

对于其他情况，同理计算并插值即可，具体方法可见代码。

2.5 双阈值处理和边缘连接

Canny 算法中减少假边缘数量的方法是采用双阈值法。选择两个阈值，根据高阈值得到一个边缘图像，这样一个图像含有很少的假边缘，但是由于阈值较高，产生的图像边缘可能不闭合，为解决此问题采用了另外一个低阈值。对非极大值抑制图像作用两个阈值 th1 和 th2，两者关系 th1=0.4th2

在高阈值图像中把边缘链接成轮廓，当到达轮廓的端点时，该算法会在断点邻域点中寻找满足低阈值的点，再根据此点收集新的边缘，直到整个图像边缘闭合。

3 代码运行结果和结果分析

3.1 灰度化和高斯模糊

图 2: 原始图

图 3: 灰度图

图 4: 高斯模糊

由图3和图4的对比可以看出，高斯模糊滤去了一部分高频细节，但同时模糊了噪声，更利于边缘提取，减少假边缘。如左图的草帽的纹理，中图直升机螺旋桨的边缘平滑等。

3.2 梯度可视化和非极大值抑制

图 5: Sobel 算子的梯度

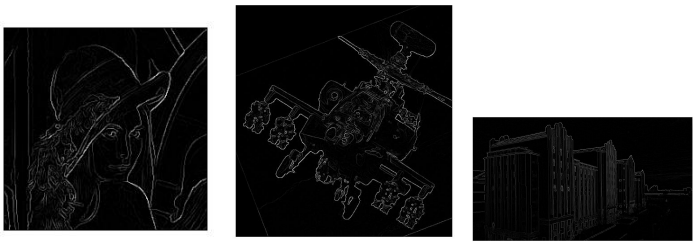

图 6: 非极大值抑制结果

可以看到，经过非极大值抑制，图片的边缘明显变细，精度变好。

3.3 双阈值检测和连接边缘

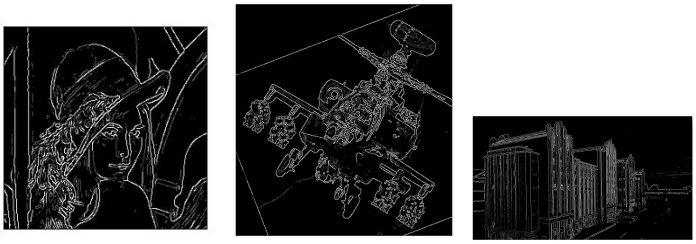

图 7: 双阈值检测结果

这里设置高阈值为 12.5，低阈值为 5。高低阈值分别对应亮暗像素点，可以发现部分单独暗像素点，对应假边缘；且部分区域亮点并不能构成闭合回路，因而需要进一步处理图像。

可以看到，经过连接边缘处理后，图形边缘基本连接在了一起，且假边缘较少。

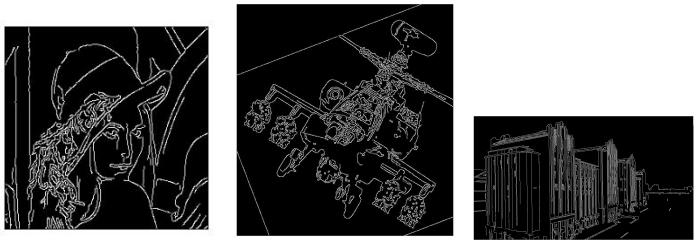

图 8: 连接边缘结果

3.4 不同阈值及内置函数的处理效果

从左至右依次为 (10, 4), (12.5, 5), (20, 8) 和内置函数 (参数为 (200, 80)) 代表设定的高低阈值。

图 9: 1.jpg 处理结果对比

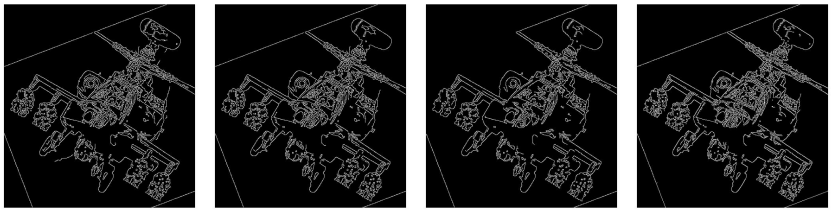

图 10: 2.jpg 处理结果对比

图 11: 3.jpg 处理结果对比

结果分析：总体来看，(10,4) 参数组仍存在较为明显的假边缘保留情况如 (2.jpg 组左下角的细线) 而 (20, 8) 参数组虽然较好地保留了主要细节，但边缘连接情况较差，对于非明显梯度变化边缘 (如 1.jpg 中的帽檐) 识别情况不好。内置函数组可能因计算方法不完全一样，采用同样的阈值得到结果较差，因此选择单独设置参数，在该参数 (200, 80) 的情况下其特征保留的结果介于 (10, 32) 和 (12.5, 5) 组的结果之间。

3.5 其他对比

3.5.1 不同梯度幅值算子性能比较

这里采用了阈值参数为 (12.5, 5) 进行控制变量，分别使用了 Sobel，Prewitt 和 Canny 算子（下图从左至右）。其定义见2.3，为节约篇幅不再重复声明, 且仅以 1.jpg 的结果作为展示，其余结果可以在 name/contrast 中找到。

图 12: 使用不同算子的结果

分析：由于算子选取的不同，对于边缘的特征提取有优劣之分，总体来看 Sobel 算子效果较好。

3.5.2 连接边缘定义联通数的影响

在边缘连接时，理想情况下我们希望得到一根细线，用于较好的刻画边缘。但同时我们也希望它能连接在一起，而非停止在某个端点。此时，如何

定义边缘线联通需要引入一个参数 cnt，表示该点及周围八个点中有多少点在图像中显示。对于一个高阈值点而言，如果 cnt 过小，说明该点大概率是一个端点，需要继续扩展；否则，认为这个点在内部，不需要继续扩展并导致边缘线变粗。

以下就 1.jpg，阈值参数为 (12.5, 5)，cnt 分界值为 2，3，4 的结果进行展示并分析。

图 13: 使用不同 cnt 分界值的结果

分析：可以看到，采用 cnt 分界值为 2 得到的图连通性较差，而分界值为 3 和 4 的效果几乎一样。可能原因是部分分界线为斜线，如果 cnt 分界值为 2，可能出现某端点的一侧及角落同时显示，此时该点被误认为是内部点而不继续扩展，导致连通性较差。

4 实验感想

本次实验学习使用了 Canny 算法检测图形边缘，让我对边缘的认识从肉眼直观上升到了数学层面。多种参数的细微调整和效果比较，也让我明白需要结合实际应用灵活设置参数。这次学习让我对图像处理的基本概念和细节有了更深的理解，同时理解了图像处理的多方法、多学科结合的综合性、多样性特点。

5 代码

可见 name/src/edge.py

```
1 import numpy as np
2 import cv2
3 import matplotlib.pyplot as plt
```

```
4  import math
5
6  from jedi.api.helpers import filter_follow_imports
7  from matplotlib import font_manager
8  import os
9  import math
10
11 font_path = 'C:\\Windows\\Fonts\\msyh.ttc' # Windows下
       引入微软雅黑字体
12 font_prop = font_manager.FontProperties(fname=
       font_path)
13
14 # 高斯滤波器生成
15 def gauss_filter(sigma=1.4, k=1):
16     ax = np.arange(-k, k + 1)
17     x, y = np.meshgrid(ax, ax)
18     gauss_mat = np.exp(-(x ** 2 + y ** 2) / (2 * sigma
           ** 2))
19     gauss_mat /= 2 * np.pi * sigma ** 2
20     return gauss_mat / gauss_mat.sum()
21
22 # 读取灰度图像
23 for num in range(1, 4):
24     image_grey = cv2.imread("../dataset/" + str(num) +
           '.jpg', cv2.IMREAD_GRAYSCALE)
25     H, W = image_grey.shape
26     # 高斯滤波
27     gauss_core = gauss_filter()
28     filter_image = cv2.filter2D(image_grey, -1,
           gauss_core)
29
30     gx, gy = [], []
31
```

```
    # Sobel 算子计算梯度
    sx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]) / 8
    sy = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]]) / 8

    # Prewitt 算子计算梯度
    # sx = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]]) / 6
    # sy = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]]) / 6
    #
    for i in range(0, H):
        gx.append([])
        gy.append([])
        for j in range(0, W):
            if i == 0 or i == H - 1 or j == 0 or j == W - 1:
                gx[-1].append(0)
                gy[-1].append(0)
                continue
            mat = filter_image[i - 1: i + 2, j - 1: j + 2]
            gx[-1].append(np.sum(mat * sx))
            gy[-1].append(np.sum(mat * sy))
    gx = np.array(gx)
    gy = np.array(gy)
    g = np.sqrt(gx ** 2 + gy ** 2)

    # Canny 算子计算梯度
    # sx = np.array([[-1, 1], [-1, 1]]) / 4
    # sy = np.array([[1, 1], [-1, -1]]) / 4
    # for i in range(H):
```

```
59      #     gx.append([])
60      #     gy.append([])
61      #     for j in range(W):
62      #         if i == 0 or i == H - 1 or j == 0 or j
            == W - 1:
63      #             gx[-1].append(0)
64      #             gy[-1].append(0)
65      #             continue
66      #         mat = filter_image[i : i + 2, j : j + 2]
67      #         gx[-1].append(np.sum(mat * sx))
68      #         gy[-1].append(np.sum(mat * sy))
69      # gx = np.array(gx)
70      # gy = np.array(gy)
71      # g = np.sqrt(gx ** 2 + gy ** 2)
72
73      # 非极大值抑制
74      g0 = np.zeros_like(g)
75
76      for i in range(1, H - 1):
77          for j in range(1, W - 1):
78              if(gx[i][j] == 0 and gy[i][j] == 0):
79                  continue
80              if(gx[i][j] == 0 and gy[i][j] != 0):
81                  tmp1 = g[i - 1][j]
82                  tmp2 = g[i + 1][j]
83                  if (g[i][j] >= tmp1 and g[i][j] >=
                        tmp2):
84                      g0[i][j] = g[i][j]
85                  continue
86              if(gx[i][j] != 0 and gy[i][j] == 0):
87                  tmp1 = g[i][j - 1]
88                  tmp2 = g[i][j + 1]
89                  if (g[i][j] >= tmp1 and g[i][j] >=
```

```
                    tmp2):
90                      g0[i][j] = g[i][j]
91                  continue
92              theta = math.atan(gy[i][j] / gx[i][j]) /
                    math.pi * 180 + 90
93              tmp1, tmp2 = 0, 0
94              if 0 <= theta < 45:
95                  k = abs(gx[i][j] / gy[i][j])
96                  tmp1 = g[i - 1][j] * (1 - k) + g[i -
                        1][j - 1] * k
97                  tmp2 = g[i + 1][j] * (1 - k) + g[i +
                        1][j + 1] * k
98              if 45 <= theta < 90:
99                  k = abs(gy[i][j] / gx[i][j])
100                 tmp1 = g[i][j - 1] * (1 - k) + g[i -
                        1][j - 1] * k
101                 tmp2 = g[i][j + 1] * (1 - k) + g[i +
                        1][j + 1] * k
102             if 90 <= theta < 135:
103                 k = abs(gy[i][j] / gx[i][j])
104                 tmp1 = g[i][j + 1] * (1 - k) + g[i -
                        1][j + 1] * k
105                 tmp2 = g[i][j - 1] * (1 - k) + g[i +
                        1][j - 1] * k
106             if 135 <= theta < 180:
107                 k = abs(gx[i][j] / gy[i][j])
108                 tmp1 = g[i - 1][j] * (1 - k) + g[i -
                        1][j + 1] * k
109                 tmp2 = g[i + 1][j] * (1 - k) + g[i +
                        1][j - 1] * k
110             if (g[i][j] >= tmp1 and g[i][j] >= tmp2):
111                 g0[i][j] = g[i][j]
112     gradmap = np.zeros_like(g)
```

```
113     img_bool = np.zeros_like(g)
114
115     # 设置高低阈值
116     high = 12.5
117     low = 0.4 * high
118     low = int(low)
119
120     que = []
121     for i in range(1, H - 1):
122         for j in range(1, W - 1):
123             if g0[i][j] >= high:
124                 gradmap[i][j] = high
125                 img_bool[i][j] = True
126                 que.append([i, j])
127             elif g0[i][j] >= low:
128                 gradmap[i][j] = low
129
130     # 边缘连接
131     vis = np.zeros_like(g)
132     while(len(que) != 0):
133         u = que.pop(0)
134         x = u[0]
135         y = u[1]
136         if vis[x][y]:
137             continue
138         img_bool[x][y] = True
139         vis[x][y] = True
140         cnt = 0
141         for i in range(-1, 2):
142             for j in range(-1, 2):
143                 if img_bool[x + i][y + j]:
144                     cnt += 1
145         if cnt <= 4:
```

```
            for i in range(-1, 2):
                for j in range(-1, 2):
                    if gradmap[x + i][y + j] >= low:
                        que.append([x + i, y + j])

    # folder_path = "../contrast"
    # os.makedirs(folder_path, exist_ok=True)
    plt.figure(figsize = (W / 100, H / 100), dpi =
        100)
    plt.imshow(img_bool, cmap='gray')
    # img_canny = cv2.Canny(image_grey, 80, 200)
    # plt.imshow(img_canny, cmap='gray')
    plt.axis('off')
    plt.tight_layout()
    # plt.savefig(folder_path + "/"+str(num)+ "_canny"
        + ".jpg")
    plt.show()
    plt.close()
```