東南大學

边缘检测算法 Canny

计算机视觉实验一

姓名 蒋雨初

学号 58121102

院所 人工智能学院

2023 年 10 月 28 日

目录

1 背景

1.1 题目描述

实现对彩色图像的灰度处理，并使用高斯一阶差分滤波器计算图像梯度，进而执行非极大值抑制和阈值操作及连接，从而进行 canny 边缘检测。

除上述要求外，本实验还额外做了以下工作：

- 对比和分析了使用 Sobel 滤波器、高斯一阶差分滤波器作为梯度计算器实现 Canny 算法的效果。
- 对比和分析了自己实现的算法和 opencv 标准算法的差别。

1.2 编程环境

VSCode + Python 3.11

2 原理

2.1 Canny 算法

Canny 边缘检测算子是澳洲计算机科学家约翰 • 坎尼于 1986 年开发出来的一个多级边缘检测算法 [1]。它包含四个基本的步骤

1. 平滑降噪。由于边缘检测容易受到图像中噪声的影响，因此第一步是使用 5x5 高斯滤波器去除图像中的噪声。
2. 寻找图像强度梯度。这一步使用微分滤波器在水平和垂直方向对平滑后的图像进行滤波，以获得水平方向的一阶导数 G_x 和垂直方向 G_y 的一阶导数，并计算梯度大小和方向。在 OpenCV 中使用的是 Sobel 滤波器，本次实验中要求使用的是一阶高斯差分滤波器。
3. 非极大值抑制。获得梯度大小和方向后，对图像进行全面扫描，以去除可能不构成边缘的任何不需要的像素。
4. 滞后阈值法处理。此阶段决定哪些所有边都是真正的边，哪些不是。

2.2 RGB 到灰度图

要将 RGB 图像转换为灰度图，直观来说会考虑三个通道取平均、取三个通道的最大/小值的方法，但是这并不符合人眼对不同颜色的敏感程度，会导致某些颜色看上去

过深或过浅。因此通常使用的公式是

$$Gray = 0.2126 \times R + 0.7152 \times G + 0.0722 \times B \tag{1}$$

$$Gray = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{2}$$

式 1是高清电视标准 [2]，式 2是显像管电视标准 [3]。

2.3 差分滤波器

差分滤波器（Differential Filter）是一种用于图像处理的滤波器。它的主要工作原理是计算相邻像素或信号之间的差值，以此来检测或强调变化的部分，因此常被用于检测图像中的边缘和纹理。差分滤波器通常使用一组预定义的卷积核来对图像进行卷积操作。

n 阶差分滤波器指的是滤波器计算的是 n 阶导数。常见的一阶差分滤波器有：Sobel 滤波器、Prewitt 滤波器、Roberts 滤波器等。二阶差分滤波器包括 Laplacian 滤波器、LoG（Laplacian of Gaussian）滤波器等。

2.3.1 Sobel 滤波器

Sobel 滤波器是一种常用的一阶差分滤波器，它用于边缘检测。它包含水平和垂直两个卷积核，分别计算图像在水平和垂直方向上的梯度。图 1给出了 Sobel 滤波器的卷积核。在通过 S_x, S_y 卷积之后，得到一阶导数 G_x, G_y，再通过 $|G_x| + |G_y|$ 或 $\sqrt{G_x^2 + G_y^2}$ 就可以得到最终的边缘图。对于求解同一方向的梯度而言，卷积核每一位置的值同时取负与原来的卷积核并无差别，即 S_x 与 $-S_x$ 是等效的。关于 Sobel 滤波器的推导，见附录 B。

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

(a) 水平方向卷积核 S_x

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(b) 垂直方向卷积核 S_y

图 1: Sobel 滤波器的卷积核

2.3.2 高斯一阶差分滤波器

二维高斯函数的表达式是

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

x, y 方向上的偏导分别是

$$\frac{\partial G}{\partial x} = -\frac{x}{\sigma^2} G(x, y)$$
$$\frac{\partial G}{\partial y} = -\frac{y}{\sigma^2} G(x, y)$$

一阶高斯差分滤波器即是使用 G_x, G_y 形成的两个卷积核。

注意如果使用高斯一阶差分滤波器计算梯度，那么就不再需要先进行降噪操作了，这是因为该滤波器同时包含了高斯函数降噪和求解梯度的过程。由卷积的导数性质（证明见附录 C）

$$\frac{df * g}{dx} = f * \frac{dg}{dx}$$

可知，欲对高斯平滑后的图像求梯度等同于以高斯一阶导数生成的滤波器卷积。

2.4 高斯平滑

高斯平滑运用的是高斯滤波器。高斯滤波器是一种线性平滑滤波器，它对图像进行卷积操作，使得每个像素的值都是其周围像素的加权平均值。在高斯平滑过程中，像素值受到离其越近的像素的影响更大，离其越远的像素的影响更小。

高斯平滑的数学公式可以表示为：

$$I_{\text{smoothed}}(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2 + j^2}{2\sigma^2}\right) \cdot I(x + i, y + j)$$

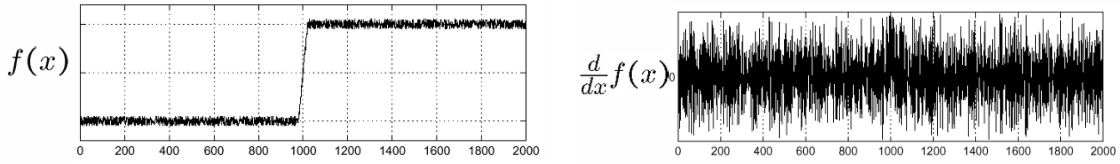
其中，$I_{\text{smoothed}}(x, y)$ 表示平滑后的图像在位置 (x, y) 处的像素值，$I(x, y)$ 表示原始图像在位置 (x, y) 处的像素值，k 是高斯滤波器的大小（通常为奇数），σ 是高斯滤波器的标准差。

在参数上，主要需要决定滤波器的大小和高斯分布的方差。较小的滤波器和较小的方差产生的模糊效果也较少，这样就可以检测较小、变化明显的细线。较大的滤波器和较大的方差产生的模糊效果也较多，将较大的一块图像区域涂成一个特定点的颜色值。这样带来的结果就是对于检测较大、平滑的边缘更加有用，例如彩虹的边缘。

高斯平滑的目的是通过应用高斯滤波器来降低图像中的高频噪声。噪声的存在会导致图像梯度震荡，不利于边界的判断。例如若直接对如图 2(a)所示的函数求导，会得到频繁震荡的导数。

2.5 非极大值抑制

非极大值抑制（Non-Maximum Suppress，NMS）算法原本来自于目标检测领域，其核心思想在于抑制非极大值的目标（去冗余），从而搜索出局部极大值的目标（找最优）。



(a) 带有噪声的图像函数

(b) 带有噪声的函数的导数

图 2: 带噪声图像的求导示意图。由于高频噪声的存在，导数频繁震荡会被算法误以为带有大量边缘。

在经过了平滑和求梯度操作之后，提取出来的边缘仍可能较为模糊，因此需要滤去局部的非极大值。

具体来说，如图 3所示，需要比较 g_a, g_5, g_b 的大小，如果 g_5 为非极大值，则抑制它（设置该点像素值为 0），否则保留它。然而，g_a 和 g_b 不一定能精准落到某个像素点上，因此需要进行近似或线性插值。

2.5.1 近似方法

近似方法是根据梯度的角度，近似到水平、竖直、主对角线、副对角线四个方向上，这也是原始论文和 OpenCV 实现中[1]使用的方法。

以图 3(a)为例，梯度会被近似到竖直方向，那么将会比较 g_2, g_5, g_8 的大小。

2.5.2 线性插值方法

线性插值的目的是为了估计出亚像素点的梯度，理论上而言它拥有比起近似方法更好的准确度。线性插值权重系数 w 由梯度分量的大小决定。以图 3(a)为例，$w = \frac{|G_x|}{|G_y|}$，那么

$$g_a = wg_1 + (1-w)g_2$$
$$g_b = wg_9 + (1-w)g_8$$

而对于图 3(c)，w 应取 $\frac{|G_y|}{|G_x|}$，那么

$$g_a = wg_1 + (1-w)g_4$$
$$g_b = wg_9 + (1-w)g_6$$

2.6 滞后阈值法

滞后阈值法（Hysteresis Thresholding）是一种常用的图像处理技术，用于图像分割和边缘检测。它基于像素的灰度值，并根据预先定义的两个阈值对像素进行分类。

[1]OpenCV 4.x Canny 算法源代码

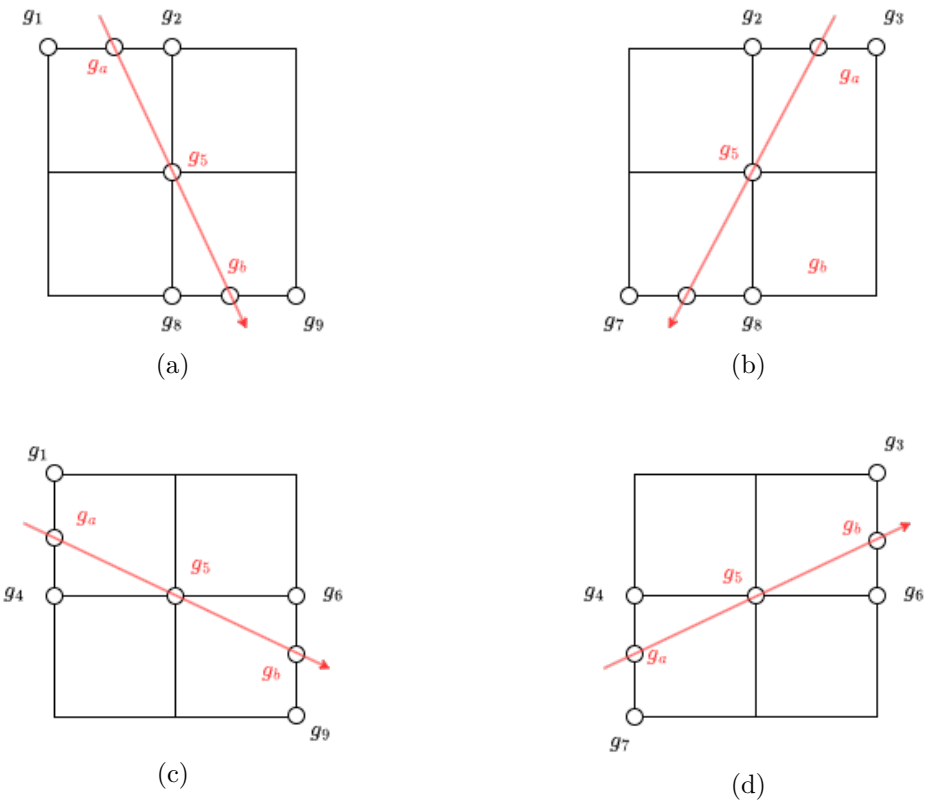


图 3: 非极大值抑制算法的四种情形，其中 g_i 表示图像对应位置的梯度大小，红色箭头表示 g_5 处的梯度，箭头方向即为梯度方向。(a)(b) 图中 $|G_y| > |G_x|$，且 (a) 图 G_y, G_x 同号，(b) 图 G_y, G_x 异号。(c)(d) 图中 $|G_y| < |G_x|$，且 (c) 图 G_y, G_x 同号，(d) 图 G_y, G_x 异号

。

滞后阈值法通过引入滞后现象来解决阈值选择时的不确定性。为此，我们需要两个阈值：minVal 和 maxVal。强度梯度大于 maxVal 的任何边缘都肯定是边缘，而低于 minVal 的边缘肯定是非边缘，因此被丢弃。位于这两个阈值之间的那些根据其连通性被分类为边缘或非边缘。如果它们连接到“确定边缘”像素，则它们被视为边缘的一部分。否则，它们也会被丢弃。

3 实现与结果

3.1 RGB 到灰度图

直接用图像和小小节 2.2中所述系数相乘

```
gray_image1_1 = np.dot(image1[..., :3], [0.299, 0.587, 0.114])
gray_image1_2 = np.dot(image1[..., :3], [0.2126, 0.7152, 0.114])
```

得到灰度图图 4。要注意的是此时得到的图像数组的数据类型是`float`类型，而灰度图应当是`uint8`，因此还需要再进行转换。

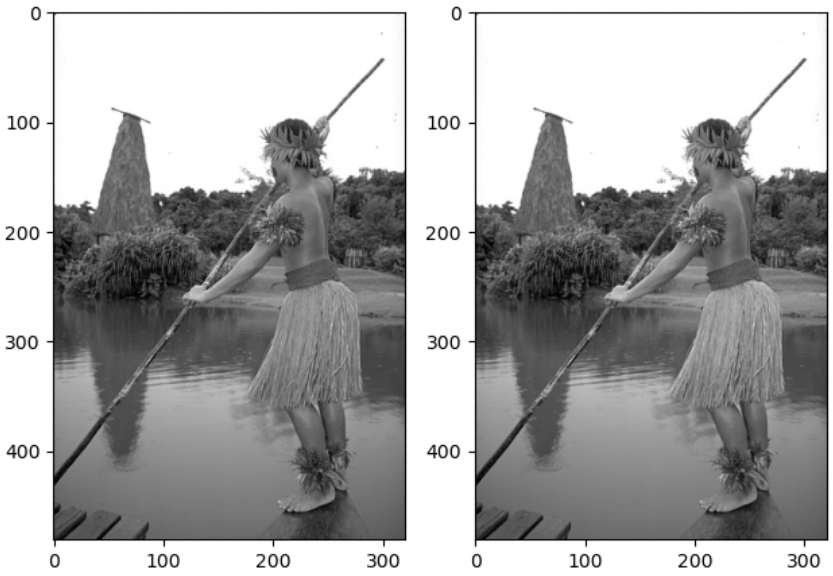


图 4: 运用公式各通道加权求和得到的灰度图像

3.2 差分滤波器

分别对图像进行高斯滤波 +Sobel 滤波器卷积与高斯一阶差分滤波器卷积。为了保证精度，在求梯度时应当把图像转到`float64`。

```
sobel_x, sobel_y = sobel_kernel()
g_kernel = gaussian_kernel(ksize=5)
dg_x, dg_y = first_derivative_gaussian_kernel(ksize=5)
```

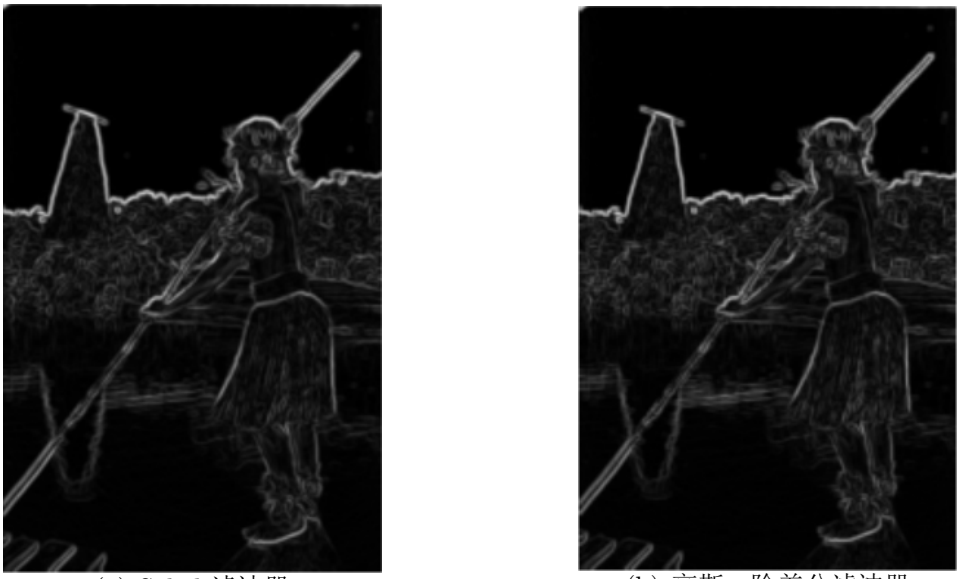
```
# sobel
sobel_grad_x, sobel_grad_y = image_gradient(
    convolve(gray_image1_1, g_kernel), kernel_x=sobel_x,
                                       kernel_y=sobel_y)
sobel_magitude = np.sqrt(sobel_grad_x**2, sobel_grad_y**2)
sobel_angle = np.atan2(sobel_grad_y, sobel_grad_x)

# first derivative gassian filter
gaussian_grad_x, gaussian_grad_y = image_gradient(
    kernel_x=dg_x, kernel_y=dg_y)
gaussian_magitude = np.sqrt(gaussian_grad_x**2, gaussian_grad_y**2)
gaussian_angle = np.atan2(gaussian_grad_y, gaussian_grad_x)
```

其中，所有卷积核大小均为 5x5，高斯卷积核的方差由经验公式

$$\sigma = 0.3 \times \left(\frac{\text{ksize}}{2} - 1 \right) + 0.8$$

确定，其中 kszie 代表卷积核的大小。得到梯度图像如图 5所示。



(a) Sobel 滤波器　　(b) 高斯一阶差分滤波器

图 5: 经过不同滤波器计算出的图像梯度

3.3 非极大值抑制

对 Sobel 滤波器计算出的梯度和高斯一阶差分滤波器计算出的梯度运用非极大值抑制，得到

(a) Sobel 滤波器

(b) 高斯一阶差分滤波器

图 6: 经过 NMS 处理后的图像

3.4 滞后阈值法

按照小小节 2.6所属，保留梯度大于 maxVal 的强边缘，剔除梯度小于 minVal 的非边缘。再以强边缘为中心，使用 dfs 遍历八个方向维护梯度介于 minVal 和 maxVal 之间的弱边缘。对于这一步，也可以使用`scipy.ndimage.label`函数标记出连接成分。最终得到图 7。

4 讨论与分析

彩色图像转灰度图

式 2和式 1其实并无区别，在不同的显示设备上自然需要有所调整。

Sobel 滤波器与高斯一阶差分滤波器的对比

从结果图 7来看，图 7(b)边缘不连续，而且出现许多横向边缘，由此观之，Sobel 滤波器的效果要好于高斯一阶差分滤波器。其原因可能是因为 Sobel 算子使用 Mahanttan 距离（见附录附录 B），这在梯度估计上更为合理。

与 OpenCV Canny 算法的对比

如图 8所示，由于自己的实现中，NMS 中采用了线性插值而不是简单的近似，所以图 8(b)有着更多的错误边缘。

(a) Sobel 滤波器

(b) 高斯一阶差分滤波器

图 7: 经过滞后阈值法处理得到的图像。其中低阈值 minVal 取 30，高阈值 maxVal 取 100。

(a) My Canny

(b) OpenCV Canny

图 8: 使用自己的 Canny 实现和 OpenCV 的 Canny 实现进行边缘提取，所有参数均保持一致。

5 心得与总结

本次实验深度探究了许多课堂上没有来得及解决的问题，最令人欣喜的莫过于理解了 Sobel 算子的起源。

我一向是一个喜欢刨根究底的人，遇到细节的问题，当别人说“你不用关心这个，这根本不重要“的时候，我总是不以为然。自由能原理提出，一个系统（比如大脑）会努力最小化其信念以及与真实世界之间的差距 [4]。大脑会首先对外部世界进行建模，形成其对外部世界的信念（可视为贝叶斯中的先验）。接着，大脑通过感受器，源源不断地汲取有关外部世界的信息，并做出预测。如果不去修正认知的偏差，大脑还如何对真实世界进行变分推理呢？

事实上本次实验大部分的时间不花在实现，而是在找问题。我注意到一开始实现的效果远不如 OpenCV，但明明流程都是相同的。我从彩色图转灰度图开始比较，最终发现是我在实现 NMS 时混淆了 x 和 y 轴的偏移量。找到问题听上去轻松，但背后是困难而艰辛的。沿途亦有收获，例如我知道了 OpenCV 中彩色图转灰度图其实使用的也是我用的公式、之所以`cv.Canny`不能指定高斯滤波的参数，其实是因为它并不包含高斯模糊。向目的地前进的路途也可观赏周围的风景。

参考文献

[1] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[2] ITU-R Recommendation BT.709. https://www.itu.int/rec/R-REC-BT.709/.

[3] ITU-R Recommendation BT.601. https://www.itu.int/rec/R-REC-BT.601/.

[4] Karl Friston, Lancelot Da Costa, Noor Sajid, Conor Heins, Kai Ueltzhöffer, Grigorios A Pavliotis, and Thomas Parr. The free energy principle made simpler but not too simple. *Physics Reports*, 1024:1–29, 2023.

附录

A 基础概念

A.1 图像梯度

在二元函数中，梯度定义如下：

$$gradf = \nabla f(x,y) = \{\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\} = f_x(x,y)\vec{i} + f_y(x,y)\vec{j}$$

如果把图像视为二元离散函数 $F(x,y)$，那么图像边缘就是函数值突变即梯度较大的位置。由于像素的最小单位是 1，所以以差分近似表示梯度

$$F_x = F(x+1,y) - F(x-1,y)$$
$$F_y = F(x,y+1) - F(x,y-1)$$

梯度的幅度和方向分别是

$$gradF = \sqrt{F_x^2 + F_y^2}$$
$$\theta = tan^{-1}(\frac{F_y}{F_x})$$

梯度幅度的另一种等价表述是

$$|gradF| = \frac{\text{像素灰度差分}}{\text{像素之间的距离}}$$

B Sobel 算子的推导[2]

B.1 Manhattan 距离

在 Sobel 滤波器中，对像素的距离的定义使用的是 Manhattan 距离而不是欧氏距离。曼哈顿距离（Manhattan distance），也称为城市街区距离或 L1 距离，是计算两个点之间的距离的一种度量方式。它得名于在城市街区网格中，两点之间的最短路径是沿着网格的边缘走，而不是直线路径。

对于二维平面上的两个点 (x_1,y_1) 和 (x_2,y_2)，曼哈顿距离可以通过以下公式计算：

$$d = |x_1 - x_2| + |y_1 - y_2|$$

B.2 卷积核推导

Sobel 算子在梯度的计算中,对于像素间的距离使用的是 Manhanttan 距离。如图 9所示，沿着四个方向求对中心 I_5 处求其梯度矢量和，可以给出 I_5 的平均梯度估计

$$\overline{gradF(I_5)} = \frac{1}{4}\left(\frac{I_3 - I_7}{4}\cdot[1,1] + \frac{I_1 - I_9}{4}\cdot[-1,1] + \frac{I_2 - I_8}{2}\cdot[0,1] + \frac{I_6 - I_4}{4}\cdot[1,0]\right)$$

式中，四个单位向量 $[1,1],[-1,1],[0,1],[1,0]$ 控制差分的方向，系数 $1/4, 1/2$ 为距离反比权重。

然而，一些典型的运算都是针对数值较小的整数的定点运算，除法会丢失低阶的重要字节, 因此消去分母，把向量乘 16，以保留低阶字节。

$$\begin{aligned} gradF' &= 16\overline{gradF(I_5)} \\ &= [I_3 + 2I_6 + I_9 - I_1 - 2I_4, I_1 + 2I_2 - I_7 - 2I_8 - I_9] \end{aligned}$$

[2]彭真明 Sobel 算子的数学基础

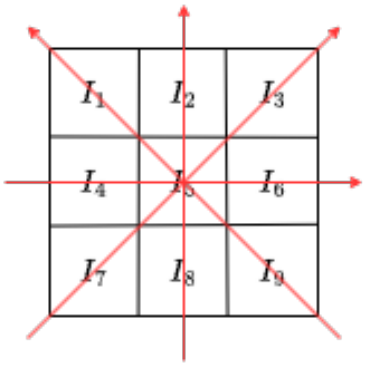


图 9: 像素邻域及 Cartesian 网格

分解到 x, y 方向

$$gradF'_x = (I_3 + 2I_6 + I_9) - (I_1 + 2I_4 + I_7)$$
$$gradF'_y = (I_1 + 2I_2 + I_3) - (I_7 + 2I_8 + I_9)$$

以上式子就是我们所见的 Sobel 算子的表达式了。

C 卷积的微分性质

当函数 $f(x)$ 和 $g(x)$ 可导时，卷积的定义如下：

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

那么有：

$$\frac{d}{dx}(f * g)(x) = \frac{df}{dx} * g(x) = f * \frac{dg}{dx}(x)$$

其中 $\frac{df}{dx}$ 和 $\frac{dg}{dx}$ 分别表示 $f(x)$ 和 $g(x)$ 的导数。

证明：

$$
\begin{aligned}
\frac{d}{dx}(f*g)(x) &= \frac{d}{dx}\left(\int_{-\infty}^{\infty} f(t)g(x-t)dt\right) \\
&= \int_{-\infty}^{\infty} \frac{d}{dx}\left(f(t)g(x-t)\right)dt \quad \text{(根据 Leibniz 积分法则)} \\
&= \int_{-\infty}^{\infty} f(t)\frac{d}{dx}\left(g(x-t)\right)dt \quad \text{(将导数移到 } f(t) \text{ 上)} \\
&= \int_{-\infty}^{\infty} f(t)\left(\frac{dg}{dx}(x-t)\right)dt \quad \text{(链式法则)} \\
&= \left(\int_{-\infty}^{\infty} f(t)\frac{dg}{dx}(x-t)dt\right) \quad \text{(移除 } dt\text{)} \\
&= (f*\frac{dg}{dx})(x)
\end{aligned}
$$