

《图像处理与建模》结课报告

边缘检测算法实践

姓名：高博钰 学号：21838031 学院：地球科学学院/理学院

2021 年 5 月 24 日

Abstract

5 行地球科学涉及对海量、各级精度卫星遥感与合成影像进行批处理，其中边缘检测是重要的环节，其应用领域主要在大尺度沙丘脊线提取，提取结果用于近代风场和气候变化研究。出于对边缘检测的兴趣，本文通过编程实践了多种经典的基于一阶和二阶梯度的边缘检测算子，其中基于一阶梯度的检测算子包括：Roberts 算子、Prewitt 算子、Scharr 算子、Kirsch 算子、Robinson 算子、Sobel 算子和 Canny 算子；基于二阶梯度的包括：Laplacian 算子和 LoG 算子。通过结果对比，本文认为在不考虑时间成本的前提下，Canny 算子能获得最精细化的边缘结果，并能够通过调整超参数完成对各类图像的边缘检测。

Key words: 边缘检测，Roberts 算子，Prewitt 算子，Scharr 算子，Kirsch 算子，Robinson 算子，Sobel 算子，Canny 算子，Laplacian 算子，LoG 算子

1 引言

图像在形成过程中，由于亮度、纹理、颜色、光照等外界因素的不同，会导致图像中某些像素点周围发生灰度值的突变，从而形成“差别”较为明显的不同区域，这些区域的边界线就是图像的边缘，也是人眼视觉系统最先捕捉到的框架信息。传统的边缘检测算子多利用图像灰度函数的“导数”信息：一阶导数能反映图像中“灰度值突变”位置；二阶导数能反映图像中“灰度值突变剧烈”的位置。由于图像是离散化的二维空间，因此在图像处理中常用差分近似求导，并在编程中用“差分卷积核”实现差分的功能。此外，基于梯度的边缘检测算子与图像去噪关系密切！因为梯度对突变十分敏感，故很多算子的卷积核都内嵌有平滑滤波的功能，例如 Prewitt、Sobel、Canny 等算子中的卷积核都是可分离的，其构成可分为：1. 平滑卷积核；2. 差分卷积核。

2 基于一阶梯度的边缘检测算子

边缘可粗略认为是图像中那些沿某一特定方向发生灰度值局部突变的像素点，这种局部突变越强烈，出现边缘的可能性越高。在数学中，变化的剧烈程度可用函数的一阶导数进行量化。如

图1中的灰度变化剖面图所示：左图沿红线剖分，沿线上的灰度值变化可用一个一维函数 $f(x)$ 描述，其导数为：

$$f'(x) = \frac{df}{dx}(x) \quad (1)$$

由右图可见：灰度值突变的位置，在一阶导数图像中会有明显的记录：

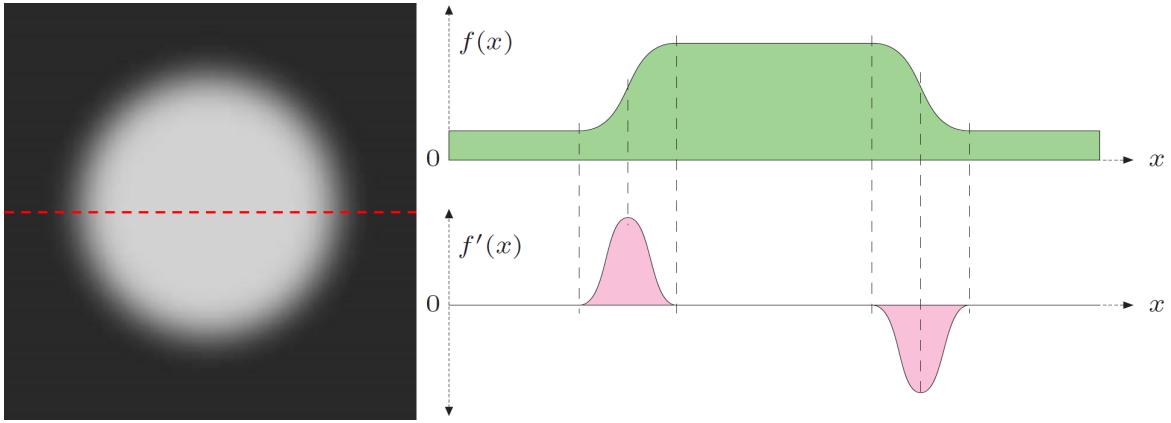


图 1. 左图为一张灰度图；右图为沿红色切线的一维灰度变化剖面图 $f(x)$ 和 $f'(x)$

现实中的图像本质都是离散的像素点，没有办法直接求导，即连续函数 $f(x)$ 在实际问题中是一个离散函数 $f(u)$ 。对于离散问题，常用差分近似代替求导（割线代替切线）。差分近似的表达式如下，其本质是离散数列的差分：

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{(u+1) - (u-1)} = \frac{f(u+1) - f(u-1)}{2} \quad (2)$$

图像差分在实际编程中用含义近似的卷积核替代，差分计算即转化为卷积计算。

更进一步，图像都是二维的，这意味着我们需要计算图像函数 $I(u, v)$ 沿 u 轴和 v 轴方向的偏导数，也就是图像 I 在坐标 (u, v) 处的梯度向量：

$$\nabla I(u, v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u, v) \\ \frac{\partial I}{\partial v}(u, v) \end{bmatrix} \quad (3)$$

每个像素点处的梯度向量的模值计算公式如下，它不会随着图像旋转而发生改变，这意味着它是一种暗含图像结构信息的变量：同为边缘的像素点的模值相近，边缘像素点与背景像素点的模值相差较大，故可通过设定阈值进行区分！后面实践的多种算法都会用到该变量：

$$|\nabla I|(u, v) = \sqrt{\left(\frac{\partial I}{\partial u}(u, v)\right)^2 + \left(\frac{\partial I}{\partial v}(u, v)\right)^2} \quad (4)$$

2.1 Roberts 边缘检测算子

Roberts 边缘检测^[1] 会用到下面两个卷积核：

$$\text{Roberts}_{135} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \text{Roberts}_{45} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \quad (5)$$

图像与上面两个 Roberts 卷积核进行卷积计算，实质上进行两个对角方向 (45° 和 135°) 的差分计算。与 Roberts_{45} 的卷积结果反应的是 45° 方向上的灰度值变化率；同理，与 Roberts_{135} 卷积

结果反应的是图像沿 135° 方向的灰度值变化率。最后，对这两个卷积结果进行合并，和得多最终的灰度图像边缘。假设有 n 个卷积核卷积后的结果，记为 $\text{conv}_1, \text{conv}_2, \dots, \text{conv}_n$ ，有下面 4 种方式将结果合并为最终的灰度图像边缘（本文均采用方式 2）：

1. 对应像素点绝对值的和： $\sum_{i=1}^n |\text{conv}_i|$
2. 对应像素点平方和再开方： $\sqrt{\sum_{i=1}^n \text{conv}_i^2}$
3. 对应像素点取绝对值的最大值： $\max \{|\text{conv}_1|, |\text{conv}_2|, \dots, |\text{conv}_n|\}$
4. 按照权重大小合并为一项： $\sum_{i=1}^n a_i |\text{conv}_i|$ ，其中权重 $a_i \geq 0$ ，且 $\sum_{i=1}^n a_i = 1$

Roberts 边缘检测较为简单，其仅使用很少的邻域像素信息（卷积核 2×2 尺寸太小，且数量不够）作为边缘强度的判断，因此可以发现该算法存在的最大问题：边缘不够连贯；也可认为该算法对细节处的边缘确定性不够。下图为图像测试实例：

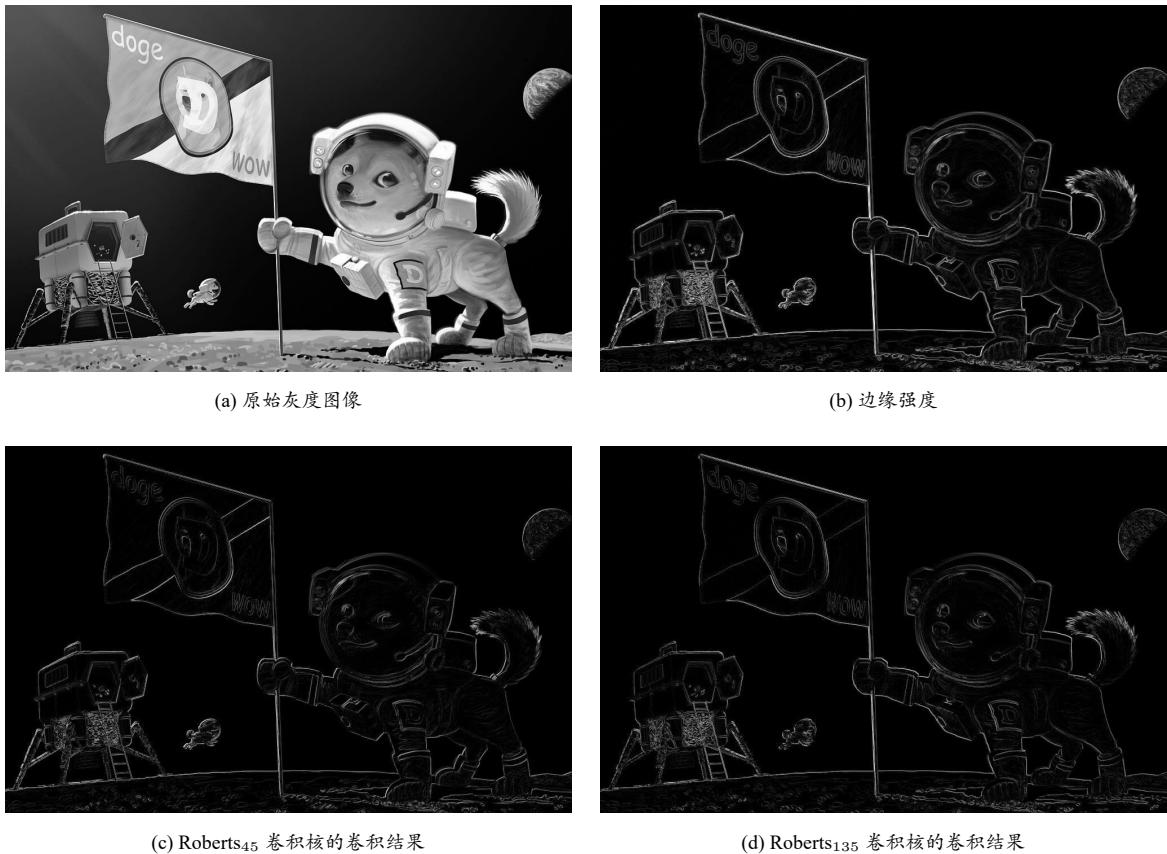


图 2. Roberts 边缘检测算子图像测试实例

2.2 Prewitt 和 Scharr 边缘检测算子

Roberts 边缘检测核心问题是卷积核尺寸太小，视野十分有限，从而导致检测到的边缘连贯性不够。为扩大卷积核的视野，标准的 Prewit 边缘检测^[2] 算子使用如下两个卷积核：

$$\text{prewitt}_x = \begin{pmatrix} 1 & 0 & -1 \\ 1 & \boxed{0} & -1 \\ 1 & 0 & -1 \end{pmatrix} \quad \text{prewitt}_y = \begin{pmatrix} 1 & 1 & 1 \\ 0 & \boxed{0} & 0 \\ -1 & -1 & -1 \end{pmatrix} \quad (6)$$

图像与 Prewitt_x 卷积可得到反映图像“垂直”方向上的边缘强度；与 Prewitt_y 卷积可得到反映图像“水平”方向上的边缘强度。另外可注意到，Prewitt 这两个卷积核都是可分离的，即：

$$\text{prewitt}_x = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 \end{pmatrix} \quad \text{prewitt}_y = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \quad (7)$$

其中， prewitt_x 中的 $(1\ 1\ 1)$ 和 prewitt_y 中的 $(1\ 1\ 1)^T$ ，在与图像做卷积运算时都有“均值平滑”的作用，即 Prewitt 算子相较于 Roberts 算子有一定的抗噪性。因此，Prewitt 算子实质由两部分组成：均值平滑算子 + 差分算子。由于卷积核的可分离性，在实际编程中可让图像先与均值平滑算子进行卷积运算，再与差分算子进行卷积运算。图像测试实例如下：

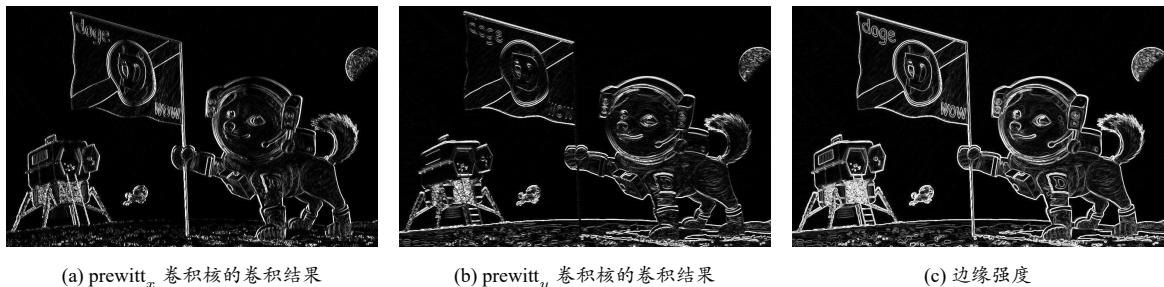


图 3. Prewitt 边缘检测算子 (x 和 y 向) 图像测试实例

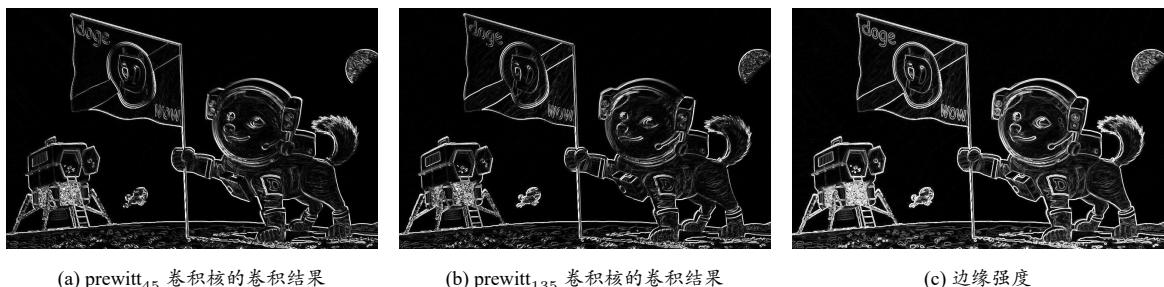


图 4. Prewitt 边缘检测算子 (45° 和 135° 向) 图像测试实例

从上面结果图可以看出 prewitt_x 与 prewitt_y 作用的明显区别。 prewitt_x 侧重“垂直 y 向”的边缘强度，忽略“水平 x 向”的边缘强度：图3(a)中“竖直的旗杆”被明显标注，但近似水平 x 向的“地表界面”却几乎看不到！ prewitt_y 的效果图正好相反。将 prewitt_x 有 prewitt_y 所发现的边缘信息整合后，可以看到非常不错的边缘提取效果（图3(c)）。

此外，Prewitt 也可像 Roberts 一样，将卷积核拓展到 45° 和 135° 方向，但这两种卷积核是不可分离的，即无法转换为两个小矩阵相乘的形式：

$$\text{prewitt}_{45} = \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix} \quad \text{prewitt}_{135} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{pmatrix} \quad (8)$$

效果如图4： prewitt_{45} 和 prewitt_{135} 都能较好的提取图像边缘，且二者只在细节处稍微不同（纹理走向不同）。由此可见，虽然卷积核尺寸（视野）仅从 2×2 扩展到 3×3 ，但边缘提取效果相较于 Roberts 算子大幅提升！

Scharr 算子相较于 Prewitt 算子只是卷积核内元素值的改变（元素总和仍为 0），卷积核都是可分离的。但是，Scharr 算子分离得到的是：非归一化的平滑算子 + 差分算子！两种算子在处于原始图像中包含“不同种噪声”时效果会有所区别，这种区别由平滑算子导致（差分算子无区别）。

$$\text{schar}_{x} = \begin{pmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{pmatrix} \quad \text{schar}_{y} = \begin{pmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{pmatrix} \quad (9)$$

对卷积核进行分离结果如下：

$$\text{schar}_{x} = \begin{pmatrix} 3 \\ 10 \\ 3 \end{pmatrix} \left(1 \quad 0 \quad -1 \right) \quad \text{schar}_{y} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \left(3 \quad 10 \quad 3 \right) \quad (10)$$

同 Prewitt，Scharr 卷积核通过旋转可得到 45° 和 135° 方向卷积核：

$$\text{schar}_{45} = \begin{pmatrix} 0 & 3 & 10 \\ -3 & 0 & 1 \\ -10 & -3 & 0 \end{pmatrix} \quad \text{schar}_{135} = \begin{pmatrix} 10 & 3 & 0 \\ 3 & 0 & -3 \\ 0 & -3 & -10 \end{pmatrix} \quad (11)$$

由于本报告的重点是通过编程实践各类算法，故在此没有测试对原始图像加不同种类噪声后，各类边缘检测算子的效果差异。图5和图6是 Scharr 算子实例测试结果：

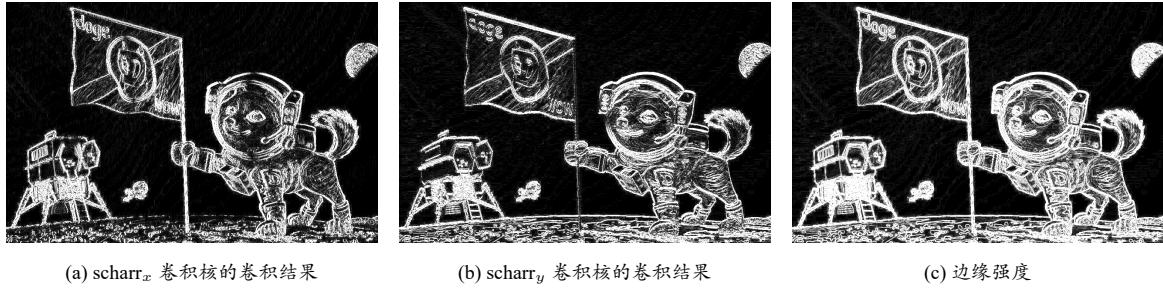


图 5. Scharr 边缘检测算子 (x 和 y 向) 图像测试实例

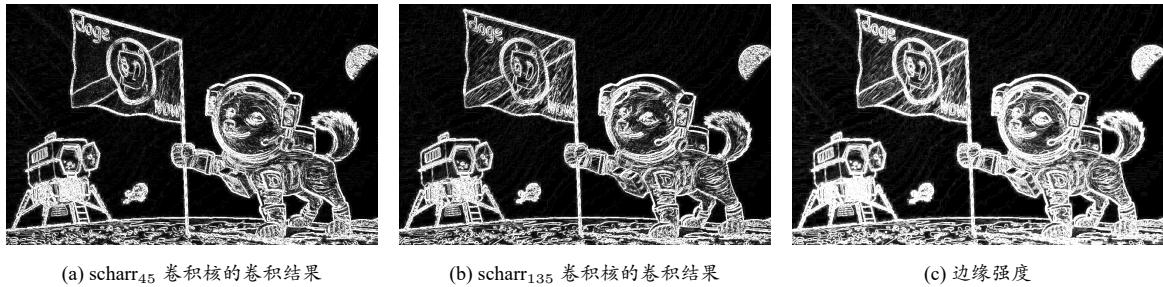


图 6. Scharr 边缘检测算子 (45° 和 135° 向) 图像测试实例

2.3 Kirsch 和 Robinson 边缘检测算子

上面的边缘检测算子都只有两个（ x 和 y 向），并通过卷积核旋转的方式获得另外两个方向的卷积核（ 45° 和 135° ）。Kirsch 和 Robinson 边缘检测算子在卷积核“旋转”上进一步拓展^[3]，从而

增多卷积核的数量！这两种算子都各有 8 个卷积核，各卷积核彼此间通过旋转可相互得到：

$$k_1 = \begin{pmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix} \quad k_3 = \begin{pmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{pmatrix} \quad k_5 = \begin{pmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{pmatrix} \quad k_7 = \begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{pmatrix}$$

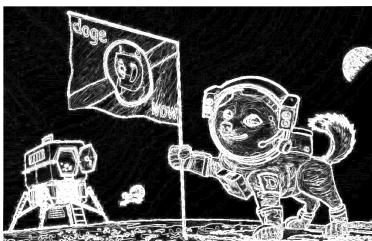
$$k_2 = \begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{pmatrix} \quad k_4 = \begin{pmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{pmatrix} \quad k_6 = \begin{pmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{pmatrix} \quad k_8 = \begin{pmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}$$

Robinson 算子的 8 个卷积核：

$$r_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{pmatrix} \quad r_2 = \begin{pmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{pmatrix} \quad r_3 = \begin{pmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{pmatrix} \quad r_4 = \begin{pmatrix} -1 & -1 & 1 \\ -1 & -2 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$r_5 = \begin{pmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad r_6 = \begin{pmatrix} 1 & -1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & 1 \end{pmatrix} \quad r_7 = \begin{pmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{pmatrix} \quad r_8 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & -1 \end{pmatrix}$$

下面只展示 Kirsch 算子 8 个卷积核的卷积结果和最终的边缘强度，Robinson 算子只需在 Kirsch 程序中修改卷积核内数值即可。不同卷积核关注不同方向的灰度值变化，这种不同只体现在细节处，大范围边缘提取几乎一样好：



(a) 边缘强度

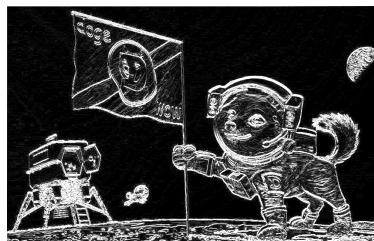
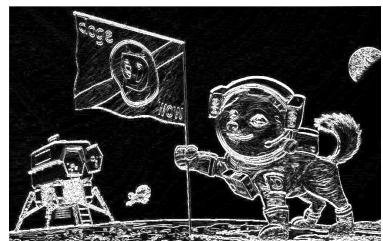
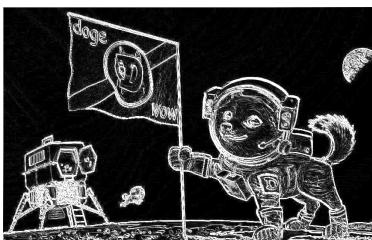
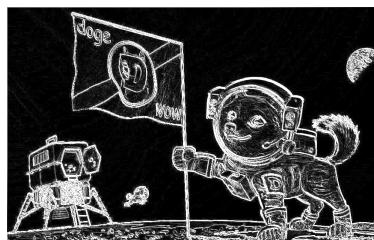
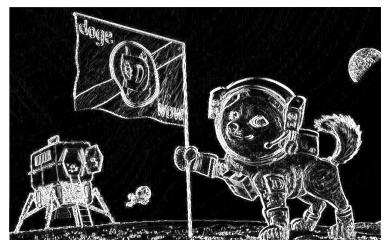
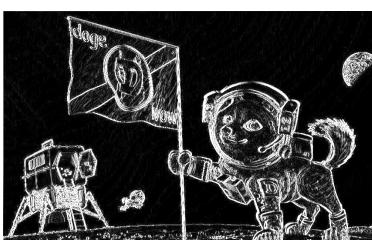
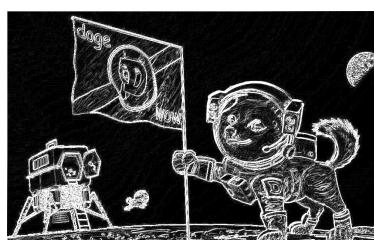
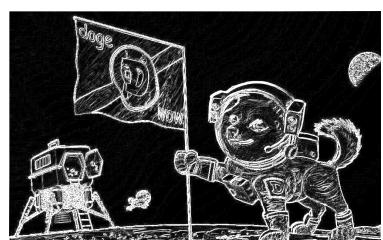
(b) kirsch_{k1} 卷积核的卷积结果(c) kirsch_{k2} 卷积核的卷积结果(d) kirsch_{k3} 卷积核的卷积结果(e) kirsch_{k4} 卷积核的卷积结果(f) kirsch_{k5} 卷积核的卷积结果(g) kirsch_{k6} 卷积核的卷积结果(h) kirsch_{k7} 卷积核的卷积结果(i) kirsch_{k8} 卷积核的卷积结果

图 7. Kirsch 边缘检测算子 (8 个卷积核) 图像测试实例

2.4 Sobel 边缘检测算子

前面所有边缘检测算子使用到的卷积核都是固定尺寸的，无法拓展到更大的卷积核！Sobel 算子^[4]在可分离卷积核的基础上，借鉴高斯卷积算子的二项式近似，将二项式展开式的系数作为非归一化的高斯平滑算子。不同幂次的二项式展开能获得不同阶的系数，使得 Sobel 算子中的卷积核理论上可拓展到任意高阶。基础的 3 阶 Sobel 算子使用如下两个可分离卷积核：

$$\begin{aligned} \text{sobel}_x &= \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \\ \text{sobel}_y &= \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \end{aligned} \quad (12)$$

sobel_x 和 sobel_y 中的非归一化平滑算子 $(1 \ 2 \ 1)^T$ 和 $(1 \ 2 \ 1)$ 本质上都是二项式展开系数。Sobel 算子工作原理：在一个坐标轴方向上进行非归一化的高斯平滑，在另一个坐标轴方向上进行差分处理。以一个 $n \times n$ 的 Sobel 算子为例（一般 n 都取奇数，方便卷积计算后图像大小一致）：非归一化的高斯平滑算子是 $n - 1$ 次幂/阶的二项式展开式的系数；差分算子是 $n - 2$ 次幂/阶的二项式展开式系数两侧补 0，然后后向差分得到的结果。对于如下的二项式通式：

$$(x + y)^n = \sum_{k=0}^n C_n^k x^k y^{n-k}, \quad n \geq 1 \quad (13)$$

展开式的系数计算通式为：

$$C_n^k = \frac{n!}{(k!) * (n - k)!}, \quad k = 0, 1, 2, 3, \dots, n \quad (14)$$

假设 Sobel 算子是 5×5 的窗口大小，即 $n = 5$ 。那么非归一化的高斯平滑算子应使用 $n - 1 = 4$ 次幂/阶的二项式展开式的系数，差分算子使用 $n - 2 = 3$ 次幂/阶的二项式展开式的系数，并做两侧补 0 和后向差分处理。二项式幂次 $n = 4$ 时，其系数为：[1 4 6 4 1]；二项式幂次 $n = 3$ 时，其系数为：[1 3 3 1]。差分算子的两侧补 0 和后向差分操作如下：

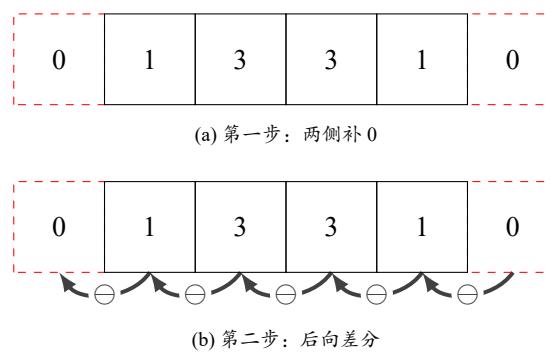


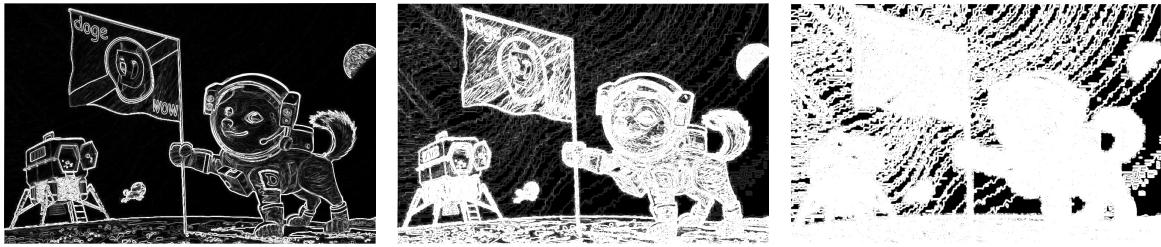
图 8. 差分算子的两侧补 0 和后向差分操作

最终获得的差分算子是: $[1 \ 2 \ 0 \ -2 \ -1]$, 元素和为 0。综上, 5 阶 Sobel 算子的最终形式为:

$$\text{sobelx}_{5 \times 5} = \begin{pmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 0 & -2 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 0 & -2 & -1 \\ 4 & 8 & 0 & -4 & -8 \\ 6 & 12 & 0 & -12 & -6 \\ 4 & 8 & 0 & -4 & -8 \\ 1 & 2 & 0 & -2 & -1 \end{pmatrix} \quad (15)$$

$$\text{sobely}_{5 \times 5} = \begin{pmatrix} 1 \\ 2 \\ 0 \\ -2 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 2 & 8 & 12 & 8 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -8 & -12 & -8 & -2 \\ -1 & -4 & -6 & -4 & -1 \end{pmatrix}$$

最后, 展示 3×3 , 5×5 和 7×7 的 Sobel 算子实测结果 (只展示最终边缘强度, 不展示 x 和 y 向), 可以发现规律: 随着卷积核尺寸的增大, 检测到的边缘连贯性不断增强, 但对细节处的小边缘不断弱化 (甚至直接合并成一条又粗又长的边缘线); 此外, 噪声也随着卷积核视野的变大而被放大! 由此可见, Sobel 算子虽理论上可拓展到任意高阶, 但检测效果**并非卷积核越大越好**。



(a) 3×3 Sobel 算子检测的边缘强度 (b) 5×5 Sobel 算子检测的边缘强度 (c) 7×7 Sobel 算子检测的边缘强度

图 9. 3×3 , 5×5 和 7×7 的 Sobel 算子实测结果

2.5 Canny 边缘检测算子

之前通过“卷积运算”的边缘检测算法不论卷积核大小和数量的多少, 都存在以下两个问题:

1. 只使用了每个像素点处的梯度模值信息, 没有使用梯度的方向信息;
2. 最后输出的边缘二值图, 都只是通过简单的阈值处理, 即: 原数值大于 255 的设定为边缘, 其他全部设置为背景。这种单阈值判断太过武断, 会损失很多细节处的边缘信息, 从而导致检测到的边缘连续性不够。

为解决以上两个问题, Canny 算子基于 Sobel 算子提出了一下两点改进^[5]:

1. 充分利用梯度方向信息进行非极大值的抑制;
2. 使用双阈值代替传统的单阈值判断: 一个高阈值, 一个低阈值; 大于高阈值的设定为边缘, 小于低阈值的设置为背景, 在两阈值中间的像素点再利用边缘的“连续性”进行判断。

(1) Canny 算子的第一步：获取边缘强度。图像矩阵 I 与水平和竖直方向卷积核 sobel_x 和 sobel_y 进行卷积计算，得到的结果记为 dx 和 dy 。根据 dx 和 dy 获得图像的幅度信息： $\text{magnitude} = \sqrt{dx^2 + dy^2}$ ，即图像的边缘强度。第一步到此为止，跟 Sobel 算子的第一步完全一样。下面用一个 5×5 的矩阵例子演示 Canny 算子的整个过程：

$$I = \begin{array}{|c|c|c|c|c|} \hline & 3 & 10 & 12 & 19 & 256 \\ \hline 240 & 239 & 8 & 7 & 10 & \\ \hline 255 & 180 & 78 & 9 & 1 & \\ \hline 170 & 200 & 197 & 168 & 50 & \\ \hline 2 & 10 & 180 & 140 & 140 & \\ \hline \end{array}$$

(a) 5×5 图像矩阵示例

$$\text{sobel}_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

$$\text{sobel}_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

(b) 3×3 Sobel 算子的两个卷积核

图 10. 原始 5×5 图像矩阵和 Sobel 的两个 3×3 卷积核

卷积计算，得到关于 x 和 y 方向的边缘强度结果 dx 和 dy ，并计算得到 magnitude：

$$dx = \begin{array}{|c|c|c|c|c|c|} \hline & 259 & -214 & -214 & 490 & -455 \\ \hline & 668 & -632 & -626 & 171 & -42 \\ \hline & 799 & -559 & -606 & -299 & -193 \\ \hline & 590 & 55 & -105 & -411 & -485 \\ \hline & 220 & 383 & 228 & -227 & -448 \\ \hline \end{array}$$

$$dy = \begin{array}{|c|c|c|c|c|c|} \hline & 719 & 726 & 262 & 32 & 27 \\ \hline & 674 & 658 & 292 & -209 & -520 \\ \hline & -179 & 41 & 500 & 551 & 241 \\ \hline & -676 & -491 & 165 & 503 & 409 \\ \hline & -540 & -767 & -762 & -583 & -268 \\ \hline \end{array}$$

$$m = \begin{array}{|c|c|c|c|c|c|} \hline & 940 & 720 & 292 & 783 & 1035 \\ \hline & 928 & 912 & 690 & 270 & 809 \\ \hline & 276 & 560 & 785 & 626 & 310 \\ \hline & 929 & 494 & 195 & 649 & 599 \\ \hline & 696 & 796 & 437 & 267 & 269 \\ \hline \end{array}$$

图 11. 卷积计算得到的 dx 和 dy ，并计算总边缘强度 magnitude

到此，Sobel 算子的处理方法是：直接对 magnitude 矩阵进行单阈值处理，大于 255 的截断为 255，其他均置为 0，然后将该矩阵保存为 8 位图即可。Canny 算子在阈值处理前，又进一步挖掘了图像中的“梯度方向”信息。

(2) Canny 算子第二步：获取梯度方向并进行非极大值抑制。利用第一步卷积计算得到的 dx 和 dy ，计算每一个像素点位置的梯度方向（角度值）： $\text{angle} = \text{actan}(\frac{dy}{dx})$ ， $\text{angle}(r, c) \in [0, 180] \cup [-180, 0]$ 。如图 12，注意图像坐标系 x 和 y 轴方向。

由于卷积运算采用了边缘补 0 的操作，使得 magnitude 的矩阵产生了边缘效应。因此，非极大值抑制不对 magnitude 矩阵的边缘一圈做处理，只根据 angle 矩阵处理 magnitude 的中间元素。记非极大值抑制处理后的结果为 nonMaxSup，那么其初始值为设置为图 12(b)。下面只需往该矩阵中剩下的位置填值即可，以 nonMaxSup(1,1) 为例：在该点放入坐标系，然后找到该点处 angle 矩阵中对应位置的角度值 $\text{angle}(1, 1) = 133^\circ$ ，再按照该角度值画出梯度方向所在的直线，最后在以 (1, 1) 为中心的 3×3 邻域内找梯度方向线大致跨过的区域，这些区域是重点关注的对象，如图 12 中的灰色区域都是每个中心点需要重点关注的邻域。

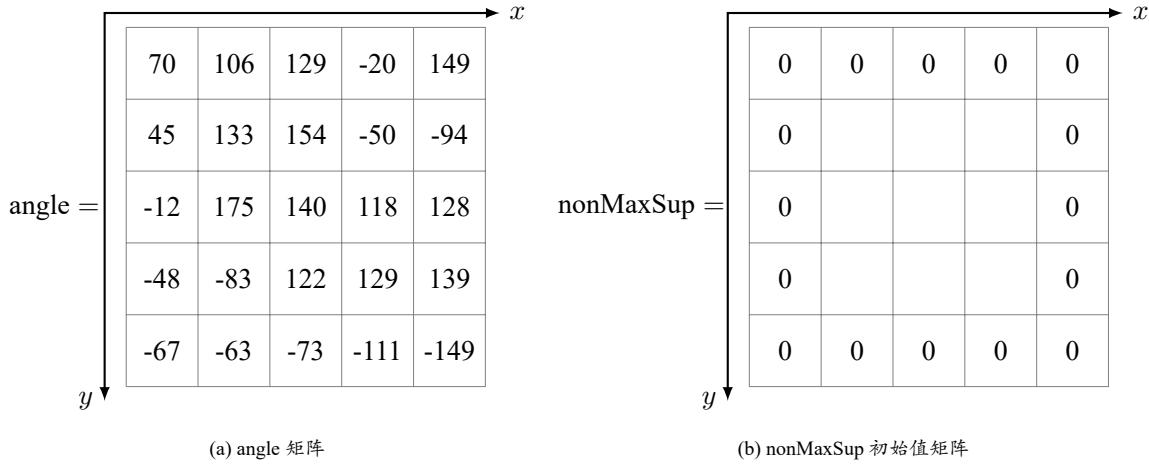
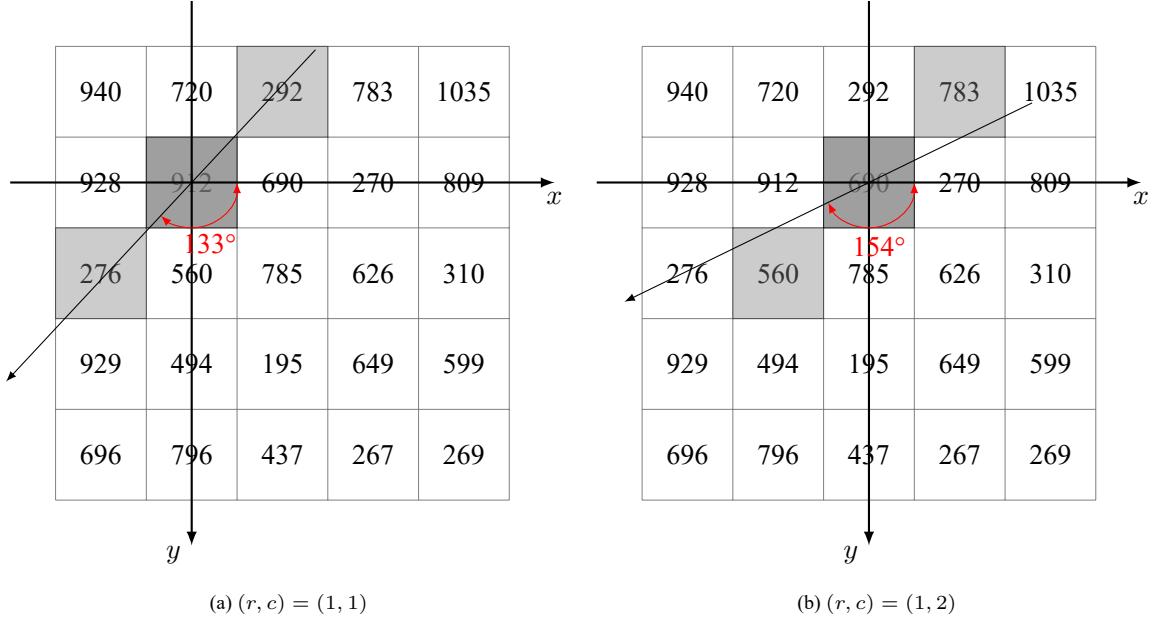


图 12. 梯度方向值矩阵与非极大值抑制的初始矩阵

接下来, 以图13(a)为例: (1, 1)位置的梯度线跨过(2, 0)和(0, 2), 即在 3×3 小区域内, 在三个像素点的有较大的可能属于同一类(边缘或背景), 非极大值抑制处理是将 $\text{magnitude}(1,1)$ 的值与另外两个进行对比, 如果它的中比梯度方向上邻域的值都大, 则可看作是局部极大值, 令 $\text{nonMaxSup}(1,1) = \text{magnitude}(1,1)$, 否则对该值进行抑制, 令 $\text{nonMaxSup}(1,1) = 0$ 。图13(a)中的(1, 1)位置: $912 > 292, 912 > 276$; 所以 $\text{nonMaxSup}(1,1) = 912$ 。同理, 图13(b)中的(2, 1)位置: $783 > 690 > 560$; 所以 $\text{nonMaxSup}(2,1) = 0$ 。

图 13. 根据梯度方向线所跨过的区域进行 3×3 局部非极大值抑制处理的示意图

总结非极大值抑制的过程: 如果 $\text{magnitude}(r, c)$ 在沿着其梯度方向线 $\text{angle}(r, c)$ 上的邻域内是最大的, 则保留其值, 否则置为0。可观察到, 上面对邻域的确定相对粗糙: 只选择梯度方向线能跨过的最有可能的两个邻域! 按照这种较为粗糙的划分思路, $\text{angle}(r, c)$ 可离散为以下4种区间, 并对应4种邻域横跨的情况:

1. $\text{angle}(r, c) \in [0, 22.5] \cup (-22.5, 0] \cup (157.5, 180] \cup [-180, 157.5]$

2. $\text{angle}(r, c) \in [22.5, 67.5] \cup [-157.5, -112.5]$
3. $\text{angle}(r, c) \in [67.5, 112.5] \cup [-112.5, -67.5]$
4. $\text{angle}(r, c) \in (112.5, 157.5] \cup [-67.5, -22.5]$

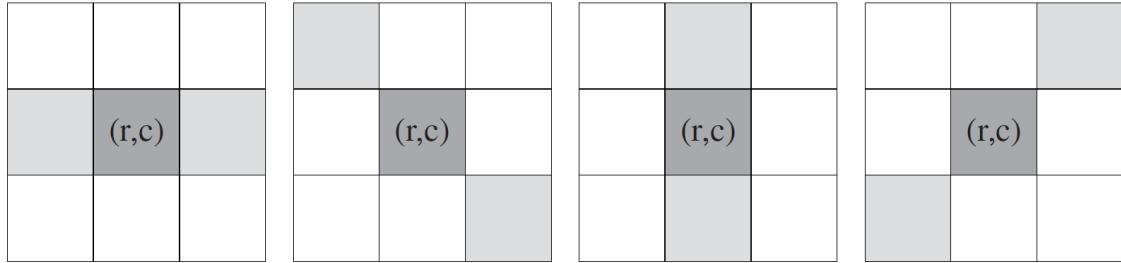


图 14. 梯度方向线跨过的 4 种可能的邻域情况

在 Canny.py 程序中，non_maximum_suppression_default 函数采用这种粗糙的邻域选择方式。图13(b) 中，很明显可以发现：梯度方向线其实跨过了 4 个邻域！如图15所示。

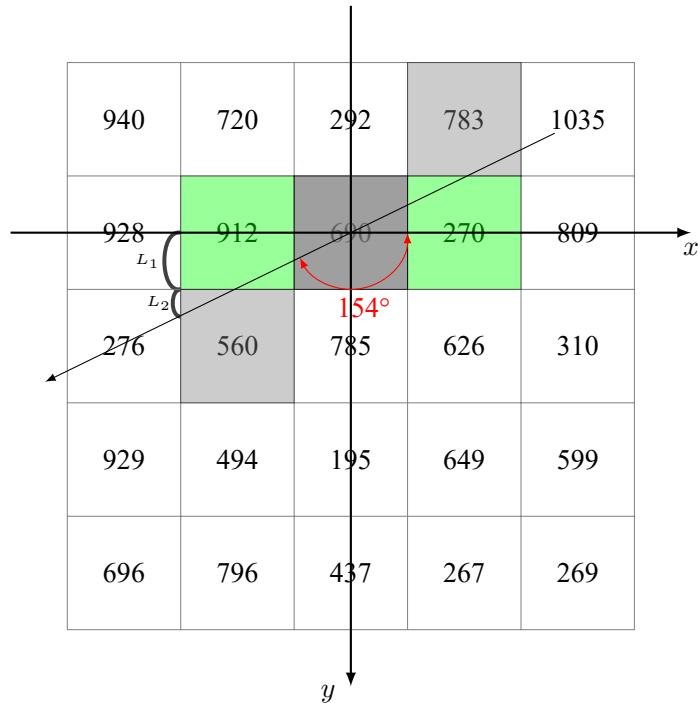


图 15. 梯度方向线跨过的 4 种可能的邻域情况

根据相似三角形易得：

$$L_1 : L_2 = \left| \frac{dy(2,1)}{dx(2,1)} \right| : \left(1 - \left| \frac{dy(2,1)}{dx(2,1)} \right| \right) \quad (16)$$

因此，我们可以用上式的比例关系，将 (2,1) 的左、左下邻域按比例插值成一项 $f(L_1, L_2)$ ：

$$f(L_1, L_2) = \left| \frac{dy(2,1)}{dx(2,1)} \right| * 560 + \left(1 - \left| \frac{dy(2,1)}{dx(2,1)} \right| \right) * 912 \quad (17)$$

同理，(2,1) 的右、右上邻域也可按照相同的比例插值成一项 $g(L_1, L_2)$ 。非极大值抑制过程：将 $\text{magnitude}(2,1)$ 的值与 $f(L_1, L_2)$ 和 $g(L_1, L_2)$ 进行对比即可。按照这种插值的方式，nonMaxSup(2,1)

依旧是 0 (计算复杂度增加)。

$$g(L_1, L_2) = \left| \frac{dy(2, 1)}{dx(2, 1)} \right| * 270 + \left(1 - \left| \frac{dy(2, 1)}{dx(2, 1)} \right| \right) * 783 \quad (18)$$

这种利用插值进行“邻域合并”的方式同样可以将 $\text{angle}(r, c)$ 离散成以下 4 种区间，并对应 4 种新的邻域横跨的情况：

1. $\text{angle}(r, c) \in (45, 90] \cup (-135, -90]$
2. $\text{angle}(r, c) \in (90, 135] \cup (-90, -45]$
3. $\text{angle}(r, c) \in [0, 45] \cup [-180, -135]$
4. $\text{angle}(r, c) \in (135, 180] \cup (-45, 0)$

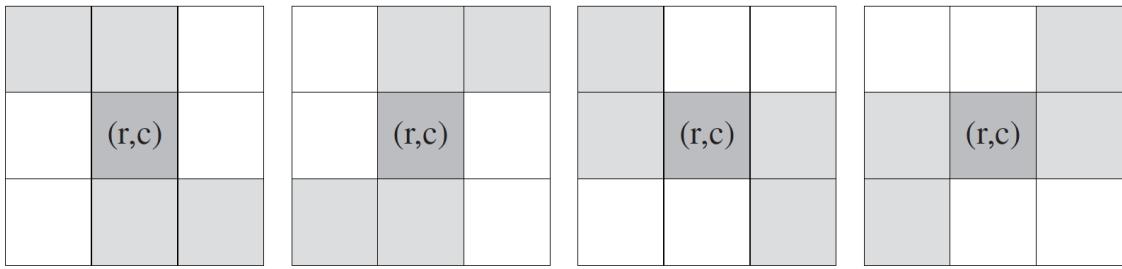


图 16. 新的 4 种可能的邻域横跨情况

根据图 16 的相对关系，可总结出以下插值计算公式。当 $\text{angle}(r, c) \in (45, 90] \cup (-135, -90]$ 时，需要计算左上 $(r - 1, c - 1)$ 和上 $(r - 1, c)$ 的插值、右下 $(r + 1, c + 1)$ 和下 $(r + 1, c)$ 的插值。此时， $|dy(r, c)| > dx(r, c)$ 。两个插值的计算公式分别为：

$$\begin{aligned} \left| \frac{dx(r, c)}{dy(r, c)} \right| * \text{magnitude}(r - 1, c - 1) + \left(1 - \left| \frac{dx(r, c)}{dy(r, c)} \right| \right) * \text{magnitude}(r - 1, c) \\ \left| \frac{dx(r, c)}{dy(r, c)} \right| * \text{magnitude}(r + 1, c + 1) + \left(1 - \left| \frac{dx(r, c)}{dy(r, c)} \right| \right) * \text{magnitude}(r + 1, c) \end{aligned} \quad (19)$$

当 $\text{angle}(r, c) \in (90, 135] \cup (-90, -45]$ 时，需要计算右上 $(r - 1, c + 1)$ 和上 $(r - 1, c)$ 的插值、左下 $(r + 1, c - 1)$ 和下 $(r + 1, c)$ 的插值。此时， $|dy(r, c)| > dx(r, c)$ 。两个插值计算公式分别为：

$$\begin{aligned} \left| \frac{dx(r, c)}{dy(r, c)} \right| * \text{magnitude}(r - 1, c + 1) + \left(1 - \left| \frac{dx(r, c)}{dy(r, c)} \right| \right) * \text{magnitude}(r - 1, c) \\ \left| \frac{dx(r, c)}{dy(r, c)} \right| * \text{magnitude}(r + 1, c - 1) + \left(1 - \left| \frac{dx(r, c)}{dy(r, c)} \right| \right) * \text{magnitude}(r + 1, c) \end{aligned} \quad (20)$$

当 $\text{angle}(r, c) \in [0, 45] \cup [-180, -135]$ 时，需要计算左上 $(r - 1, c + 1)$ 和上 $(r - 1, c)$ 的插值、右下 $(r + 1, c + 1)$ 和右 $(r, c + 1)$ 的插值。此时， $|dy(r, c)| < dx(r, c)$ 。两个插值计算公式分别为：

$$\begin{aligned} \left| \frac{dy(r, c)}{dx(r, c)} \right| * \text{magnitude}(r - 1, c - 1) + \left(1 - \left| \frac{dy(r, c)}{dx(r, c)} \right| \right) * \text{magnitude}(r, c - 1) \\ \left| \frac{dy(r, c)}{dx(r, c)} \right| * \text{magnitude}(r + 1, c + 1) + \left(1 - \left| \frac{dy(r, c)}{dx(r, c)} \right| \right) * \text{magnitude}(r, c + 1) \end{aligned} \quad (21)$$

当 $\text{angle}(r, c) \in (135, 180] \cup (-45, 0)$ 时，需要计算右上 $(r - 1, c + 1)$ 和右 $(r, c + 1)$ 的插值、左下 $(r + 1, c - 1)$ 和左 $(r, c - 1)$ 的插值。两个插值计算公式分别为：

$$\begin{aligned} \left| \frac{dy(r, c)}{dx(r, c)} \right| * \text{magnitude}(r - 1, c + 1) + \left(1 - \left| \frac{dy(r, c)}{dx(r, c)} \right| \right) * \text{magnitude}(r, c + 1) \\ \left| \frac{dy(r, c)}{dx(r, c)} \right| * \text{magnitude}(r + 1, c - 1) + \left(1 - \left| \frac{dy(r, c)}{dx(r, c)} \right| \right) * \text{magnitude}(r, c - 1) \end{aligned} \quad (22)$$

非极大值抑制过程相当于只保留了局部极大值，抑制非极大值，所以 Canny 算子中的这一步相当于进一步细化 Sobel 算子得到的边缘。这种效果反映在提取结果中就是：Canny 算子检测到的边缘要比其他算子结果都“细”！

(3) Canny 算子第三步：双阈值处理。双阈值处理的对象是上一步中获得的非极大值抑制处理后的矩阵。双阈值处理顾名思义，有两个阈值：1. 高阈值 (upperThresh); 2. 低阈值 (lowerThresh)。借助这两个阈值，按照如下规则进行边缘/背景的最后筛选：

1. 边缘强度大于高阈值的像素点直接定为边缘点；
2. 边缘强度小于低阈值的像素点直接定为背景点；
3. 边缘强度处于两个阈值中间时（待定），要按照以下要求进行适量地“边缘延长”：待定点必须能按某一路径（水平/竖直/斜线）与已确定的边缘点相连时，才有可能作为“边缘的延续”被接受为边缘点。组成这条路径上的所有点（不能有间断，即路径上不能有背景点）都必须比低阈值大，即都是边缘点或待定点。

注意：规则 3 看似复杂，但其实它的目的很明确！尽可能让边缘点周围的待定点成为“边缘的延续”，宁滥毋缺。以上就是 Canny 算子的实习步骤和技术细节，在 Canny.py 程序中，non_maximum_suppression_Inter 函数实现插值的邻域选择方式。此外，Canny.py 程序中有三个“超参数”可以人为指定：卷积核大小、高/低阈值。虽然都是超参数，但并非意味着这些参数取任意范围内的值都能有好结果。下图先展示：卷积核大小为 3×3 ，高低阈值分别为 180 和 40 的处理结果，并与 Sobel 算子的结果做对比：

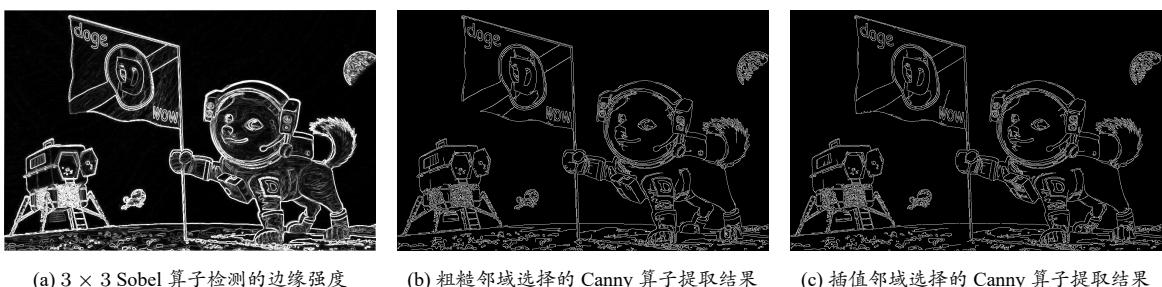


图 17. 3×3 粗糙/插值邻域选择的 Canny 算子边缘检测结果，并与同卷积核尺寸的 Sobel 算子提取结果做对比，可发现 Canny 算子提取到的边缘都特别“精细”，且边缘连续性较好。

卷积核尺寸大小对提取效果的影响同 Sobel 算子中的测试，高/低阈值对整体提取效果影响不大，且对于输入的一张图像，高/低阈值可选择的区间不大，故这里不再展示不同阈值设定下的边缘提取结果。

3 基于二阶梯度的边缘检测算子

前面所有算子都是基于一阶导数的差分近似进行边缘提取。一阶导数最大的问题是只能得到“灰度值突变”类型的边缘，无法得到“灰度值突变速度最快”类型的边缘。为获得后者，需要进阶到二阶梯度，即度量局部“曲率”的变化。通常，图像中关于后类的边缘，常出现在二阶导数的零点位置。

3.1 Laplacian 边缘检测算子

二维函数 $f(x, y)$ 的 Laplacian 变化如下，但由于图像是离散的二维空间，故仍需要用差分近似替代二阶梯度：

$$\begin{aligned}\nabla^2 f(x, y) &= \frac{\partial^2 f(x, y)}{\partial^2 x} + \frac{\partial^2 f(x, y)}{\partial^2 y} \\ &\approx \frac{\partial(f(x+1, y) - f(x, y))}{\partial x} + \frac{\partial(f(x, y+1) - f(x, y))}{\partial y} \\ &\approx f(x+1, y) - f(x, y) - (f(x, y) - f(x-1, y)) \\ &\quad + f(x, y+1) - f(x, y) - (f(x, y) - f(x, y-1)) \\ &\approx f(x+1, y) + f(x-1, y) + f(x, y-1) + f(x, y+1) - 4f(x, y)\end{aligned}\tag{23}$$

更进一步，用卷积核近似替代差分，可得如下两种拉普拉斯卷积核（二者只差一个负号，使用时只用其中一个即可）。图像与拉普拉斯卷积核进行卷积运算的实质是：计算任意像素点位置的值与其在“水平方向”和“垂直方向”方共 4 个邻域像素点平均值之间的差值。

由于 Laplacian 算子考察灰度值变化的二阶梯度，那么它对图像中的噪声特别敏感！单独使用拉普拉斯卷积核而不配备一个平滑滤波算子，很难得到好的提取效果，效果如图 18(a)（只关注“灰度值突变速度最快”类型的边缘，有可能忽略“灰度值突变”的边缘）。

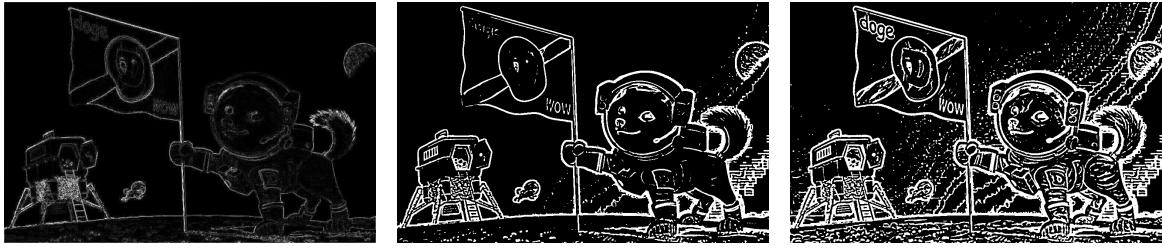


图 18. Laplacian 算子和 LoG 算子边缘提取结果

3.2 LoG 边缘检测算子

只用拉普拉斯卷积核处理图像会对噪声十分敏感，故通常在卷积计算前对图像进行高斯平滑滤波。LoG 算子^[?]（Laplacian-of-Gaussian）的思想就是将“高斯平滑滤波”和“卷积计算”合并在一起形成一个新的卷积核，用该卷积核处理图像可同时实现平滑滤波和边缘提取。这里要用到二维的高斯函数：

$$\text{gauss}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)\tag{24}$$

上式的二维梯度表达式如下。通常称 $\nabla^2(\text{gauss}(x, y, \sigma))$ 为高斯拉普拉斯（LoG）算子。LoG 边缘检测可分为以下两个步骤：

1. 构建窗口大小为 $H \times W$ 、标准差为 σ 的 LoG 卷积核（尺寸一般为奇数，且 $H = W$ ）；

$$\text{LoG}_{H \times W} = \left[\nabla^2 \text{gauss} \left(w - \frac{W-1}{2}, h - \frac{H-1}{2}, \sigma \right) \right]_{0 \leq h < H, 0 \leq w < W}\tag{25}$$

2. 图像与 LoG 卷积核进行卷积计算，计算结果用单阈值 255 划分边缘与背景值。

$$\begin{aligned}
\nabla^2(\text{gauss}(x, y, \sigma)) &= \frac{\nabla^2(\text{gauss}(x, y, \sigma))}{\partial^2 x} + \frac{\nabla^2(\text{gauss}(x, y, \sigma))}{\partial^2 y} \\
&= \frac{1}{2\pi\sigma^2} \frac{\partial \left(-\frac{x}{\sigma^2} \exp \left(-\frac{x^2+y^2}{2\sigma^2} \right) \right)}{\partial x} + \frac{1}{2\pi\sigma^2} \frac{\partial \left(-\frac{y}{\sigma^2} \exp \left(-\frac{x^2+y^2}{2\sigma^2} \right) \right)}{\partial y} \\
&= \frac{1}{2\pi\sigma^4} \left(\frac{x^2}{\sigma^2} - 1 \right) \exp \left(-\frac{x^2+y^2}{2\sigma^2} \right) + \frac{1}{2\pi\sigma^4} \left(\frac{y^2}{\sigma^2} - 1 \right) \exp \left(-\frac{x^2+y^2}{2\sigma^2} \right) \\
&= \frac{1}{2\pi\sigma^4} \left(\frac{x^2+y^2}{\sigma^2} - 2 \right) \exp \left(-\frac{x^2+y^2}{2\sigma^2} \right)
\end{aligned} \tag{26}$$

LoG 边缘检测算子的测试结果如图18(b-c) 所示。注意到：LoG 算子需要使用较大尺寸的卷积核！对于本例中的图像，LoG 卷积核不能小于 9×9 ，否则很多细节边缘根本检测不到。但与此同时，卷积核尺寸过大将放大噪声：如图18(c) 中“天空”中的“平行条纹”边缘，这明显是细节噪声被放大导致的。

4 结论

本文编程实践了 9 种经典的边缘检测算法，并详细阐述了各算子的实现过程。可以总结出以下规律：1. 算法使用的卷积核数量越多越好：卷积核是差分的编程实现形式，它实质上代表图像沿各个方法的灰度值（梯度）变化情况。考察的方向越多，越有可能发现新的边缘，在最终的信息汇总时能获得更准确、细节的边缘强度信息。2. 卷积核尺寸不能太大：不论是基于一阶还是二阶梯度，卷积核的视野都不能太大（如 Sobel 和 LoG 的实践例子）！因此当卷积核视野太大时，很多并不相连的边缘有可能被算法“延长”成同一条边缘！甚至有些并非边缘的噪声，也会被“延长”成边缘。

综上，在不考虑时间成本的条件下，Canny 算子能提取到最“精细”的边缘效果，并且由于它基于 Sobel 算子，使得其卷积核尺寸可以根据图像情况自由调整，即 Canny 算子兼顾实用性与灵活性。但是，这并不意味着其他算子不好！通过本次结课报告的实践，我对算子提取到的“边缘”产生了新的认识：不同算子对于明显的大框架、大边缘都能获得很好的提取效果，差别只在于细节。但细节处的“边缘”也分种类，也因人而异！Canny 算法虽好，但有些它提取到细节边缘我认为可有可无，因为在人眼看来那些细节只能认为“神似”边缘而已，并非是人一眼就可区分的边缘（肉眼一开始是不认可它是边缘，只是程序结果在说服大脑而已）。因此，我认为对一张高质量（非科学类图像）图像的边缘提取已经完全实现了，没有必要进一步提高算法在“模糊细节”处的边缘提取。

另外，本文原计划将编程实践的各算子应用到真实遥感影像的边缘检测，但失败了。原因很简单：真实遥感影像中的目标边缘与背景相似度太高，可理解为边缘太过“模糊”。因此，本报告只使用一张无噪声干扰的灰度图像作为算法测试的样本。

参考文献

- [1] L.G.Roberts. Machine Perception of Three-Dimensional Solids, In: Optical and Electro-Optical Information Processing[M]. MT Press: Cambridge, MA, pp.159-197, 1965.

- [2] Prewitt.J. Object Enhancement and Extraction. In: Lipkin, B. and Rosenfeld, A., Eds., Picture Processing and Psychopictorics[M]. Academic Press: New York, pp.75-102, 1970.
- [3] R.A.Kirsch. Computer Determination of the Constituent Structure of Biological Image[J]. Computers and Biomedical Research, vol.4, no.3, pp.315-328, Sep. 1970.
- [4] I.E.Sobel. Camera Models and Machine Perception[D]. PhD Thesis, Stanford University, pp.33-36, 1970.
- [5] J.Canny. A Computational Approach to Edge Detection[C]. IEEE Trans.PAMI, vol.8, no.6, pp.679-698, Nov. 1984.
- [6] D.C.Marr and E.Hildreth. Theory of edge detection[J]. Proc.R.Soc.Lond, Series B. Biological Sciences. vol.207, no.1167, pp.187-217, Nov. 1980.