Task6 边缘检测

如果有人愿意看我写的文章，不需要顺序看，可以先看算法，再往前面基础知识看，因为我写的时候就是这样写的。

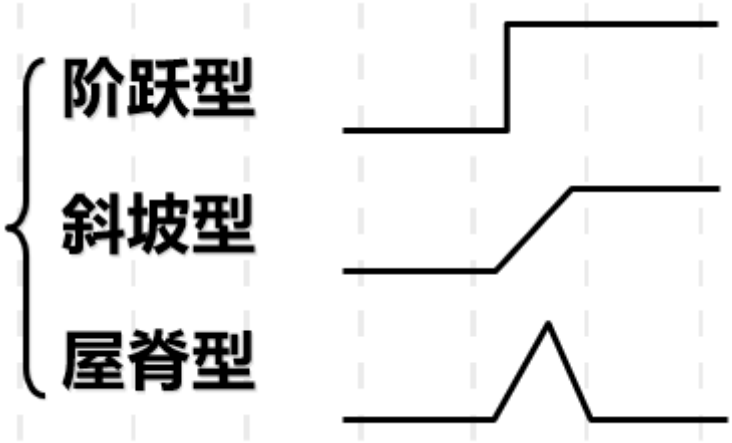
一、概念介绍

1、边缘像素：图像中灰度突变的像素。

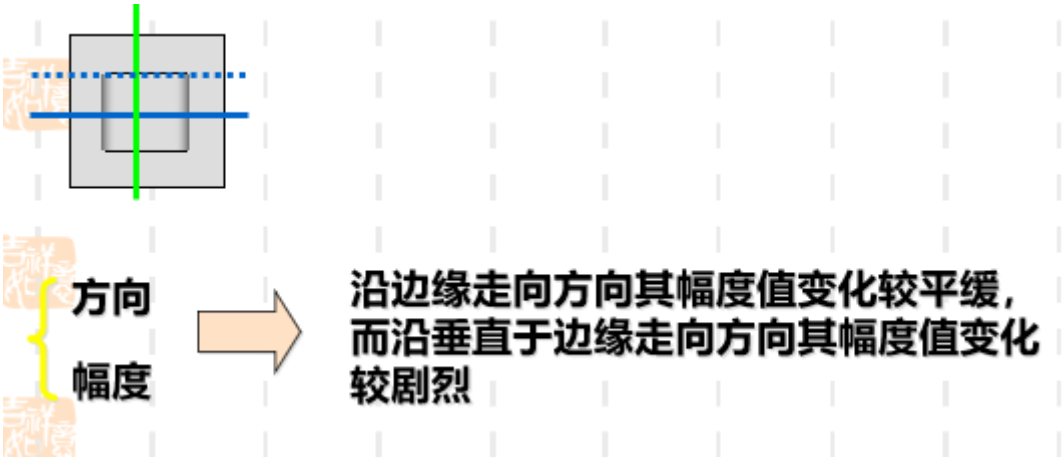
2、边缘：图像强度函数快速变化的地方。

3、边缘检测：**在灰度图像的情况下，基于图像像素的灰度值在空间的不连续性对图像作出的一种分割。**为了检测边缘，我们需要检测图像中的**不连续性**，可以使用**导数**来检测不连续性。

4、图像边缘的分类：



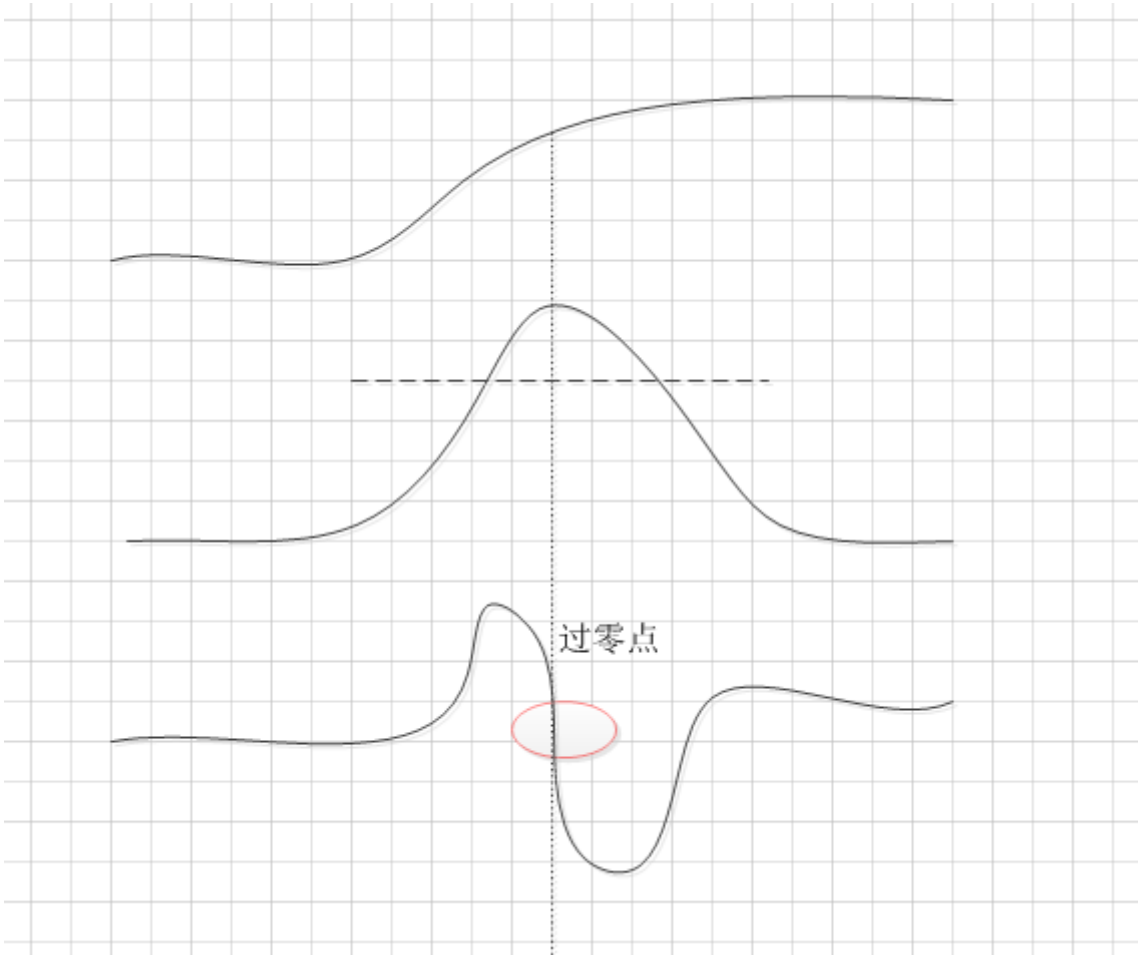
5、图像边缘的描述：



6、解释：该解释引用自博客：

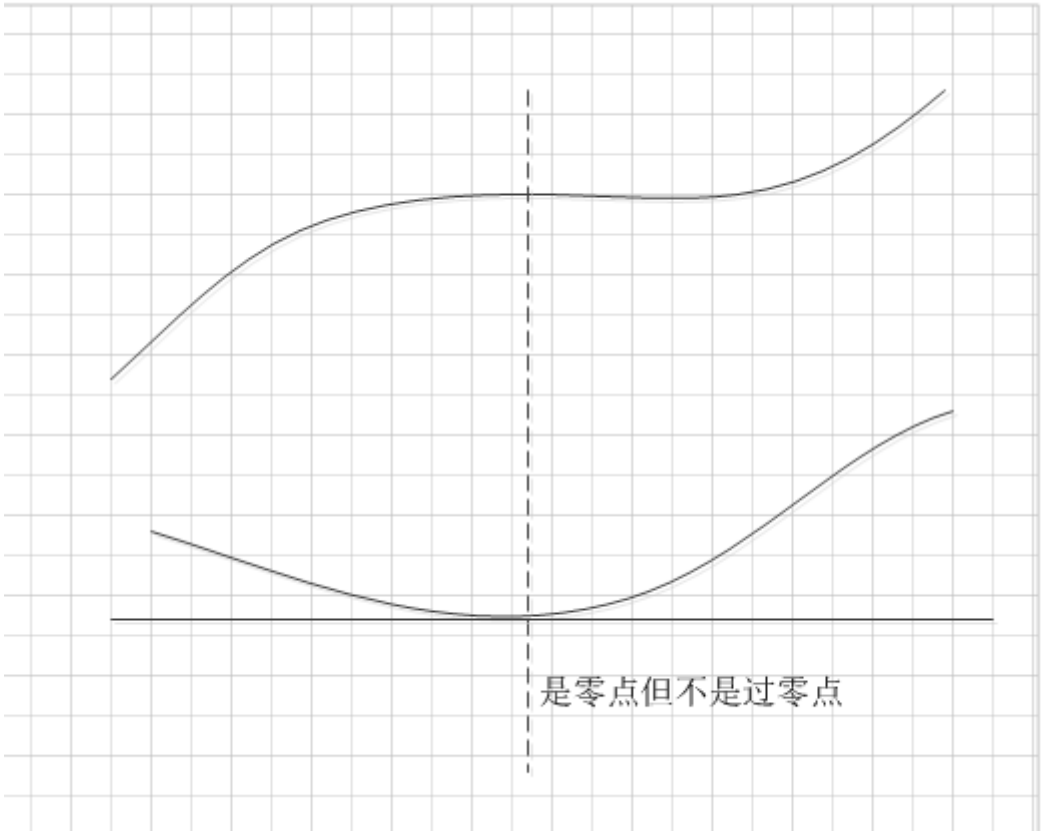
https://blog.csdn.net/zwq940103/article/details/89205820

如果图像灰度变化剧烈，进行一阶微分则会形成一个局部的极值，由数学上的知识，对图像进行二阶微分则会形成一个过零点，并且在零点两边产生一个波峰和波谷，我们要设定一个阈值，检测到这个过零点，如下图所示：

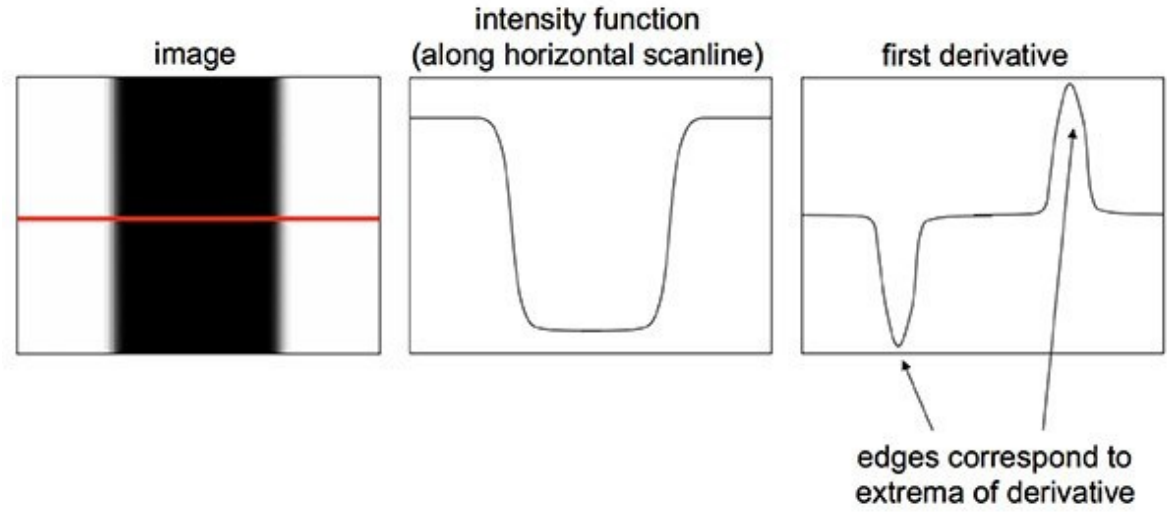


注意：

我们检测的必须是“过零点”，而不单单是零点，也就是要保证这个被选中的点一定要是局部极值点。比如下面这个例子，上面的曲线是图像空间，虚线处的点并不是图像的局部极值点，但求二阶导的时候确实是零点。再比如图像灰度的平坦区域，不管是一阶导还是二阶导都是0，它们显然不是我们要的点：



7、边缘检测的示例：

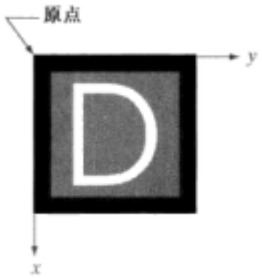


上图的第一幅图表示一张数字图片，我们对水平红线处进行求导，便可得到上图二中的关系，可以看到在边缘处有着较大的跳变。但是，导数也会受到噪声的影响，因此建议在求导数之前先对图像进行平滑处理（上图三）。这里只求了一阶导数。

但是，导数也会受到噪声的影响，因此建议在求导数之前先对图像进行平滑处理。 然后我们可以使用遮罩使用卷积来检测边缘。

二、数学基础

注意，图像坐标系与笛卡尔坐标系是不同的，下文中所有的坐标系都以下图的坐标系为准：



2.1 导数

我们按照如下方式来得到一维函数f(x)在点x处的导数的近似：将函数$f(x+\Delta x)$展开为关于x的泰勒级数，令$\Delta x=1$，且只保留该级数的线性项。结果得到数字差分：

$$\frac{\partial f}{\partial x}=f'(x)=f(x+1)-f(x)$$

将上式对x再微分，可以得到二阶导数表达式：

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2}&=\frac{\partial f'(x)}{\partial x}=f'(x+1)-f'(x)\\&=f(x+2)-f(x+1)-f(x+1)+f(x)\\&=f(x+2)-2f(x+1)+f(x)\end{aligned}$$

将x替换为x-1，可以得到：

$$\frac{\partial^2 f}{\partial x^2} = f''(x) = f(x+1) + f(x-1) - 2f(x)$$

2.2 图像梯度

为了在一幅图像f的(x,y)位置处寻找边缘的强度和方向，所选择的工具就是梯度，梯度用Δf表示，并用向量来定义：

$$\nabla f \equiv \text{grad}(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

该向量有一个重要的几何性质，它指出了f在位置(x,y)处的最大变化率方向。

向量Δf的大小表示为M(X,Y)，即

$$M(x,y) = mag(\Delta f) = \sqrt{g_x^2 + g_y^2}$$

它是梯度向量方向变化率的值。其中g_x和g_y和$M(x,y)$都是与原图像大小相同的图像，是x和y在f中的所有像素位置上变化时产生的。

在计算时，一般使用下面的方法代替计算平方和和平方根的复杂计算：

$$M(x,y) = |g_x| + |g_y|$$

梯度向量的方向由下列对于x轴度量的角度给出：

$$\alpha(x,y) = arctan[\frac{g_y}{g_x}]$$

如在梯度图像的情况那样，$\alpha(x,y)$也是与由g_y除以g_x的阵列创建的尺寸相同的图像。任意点(x,y)处于一个边缘的方向与该点处梯度向量的方向$\alpha(x,y)$正交。

2.3 梯度算子

要得到一幅图像的梯度，则要求在图像的每个像素位置处计算偏导数$\frac{\partial f}{\partial x}$和$\frac{\partial f}{\partial y}$。我们处理的是数字量，因此要求关于一点的邻域上的偏导数的数字近似。由2.1节，我们可以知道：

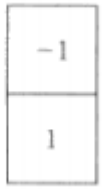
$$g_x = \frac{\partial f(x,\ y)}{\partial x} = f(x+1,\ y) - f(x,\ y)$$

$$g_y = \frac{\partial f(x,\ y)}{\partial y} = f(x,\ y+1) - f(x,\ y)$$

这种形式又称为差分算子。

这两个公式对所有x和y的有关值可用下图中的一维模板对f(x,y)的滤波来进行：

a b



是x方向上和y方向上的操作。

当对对角线方向的边缘感兴趣时，我们需要一个二维模板。罗伯特交叉梯度算子是最早尝试使用具有对焦优势的二维模板之一。（**用于计算梯度偏导数的滤波器模板，通常称为梯度算子、差分算子、边缘算子或者边缘检测子。**）

下文中几个算子都是在此图像基础上进行操作的：

如图所示的3x3区域：

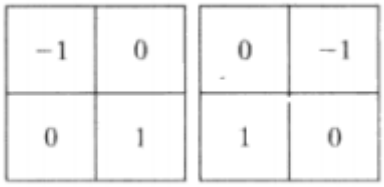
三、Robert算子

Robert算子以求对角像素之差为基础：

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5)$$

$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6)$$

这些导数可以使用下图所示的模板对图像进行滤波来实现：



Roberts

三、Prewitt算子

对于Robert算子，其对于用关于中心点对称的模板来计算边缘方向不是很有用。这种情况下最小模板大小为3x3，考虑了中心点对端数据的性质，并有关于边缘方向的更多信息。这就是Prewitt算子：

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

形式如图：

−1	−1	−1	−1	0	1
0	0	0	−1	0	1
1	1	1	−1	0	1

Prewitt

公式中，3x3区域的第三行和第一行之差近似为x方向的导数，第三列和第一列之差近似为y方向上的导数。

四、Sobel算子

在Prewitt算子上在中心系数上使用一个权值2，即得到了Sobel算子：

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

Sobel

相比Prewitt算子，Sobel算子能够较好得抑制噪声。

注意到，上面所有的算子中系数之和均为0，即表示恒定灰度区域的响应为0。

举例：

若图像G中一个3x3的窗口为A，要计算梯度的像素点为e，则和Sobel算子进行卷积之后，像素点e在x和y方向的梯度值分别为：

$$G'_x = G_x * A = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \text{sum}\left(\begin{bmatrix} -a & 0 & c \\ -2d & 0 & 2f \\ -g & 0 & i \end{bmatrix}\right)$$

$$G'_y = G_y * A = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \text{sum}\left(\begin{bmatrix} a & 2b & c \\ 0 & 0 & 0 \\ -g & -2h & -i \end{bmatrix}\right)$$

其中“ * ”为卷积符号，sum表示矩阵中所有元素相加求和。

上面提到的边缘检测方法未对图像特性和噪声内容采取预防措施。下面介绍一些高级的检测器。

五、Marr-Hildreth边缘检测器

5.1 高斯拉普拉斯算子（LoG）

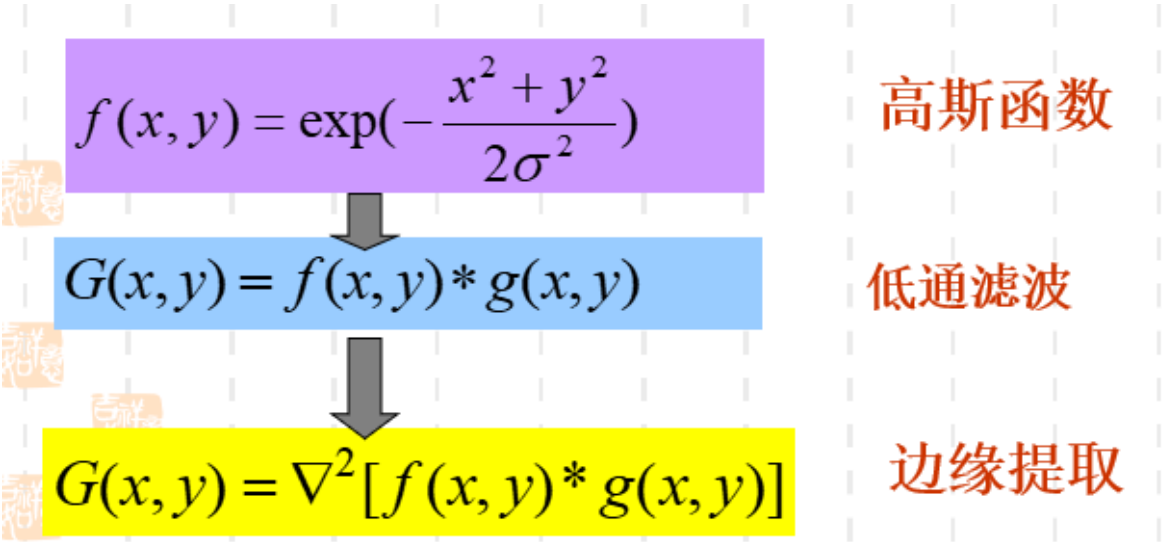
拉普拉斯算子的形式为：

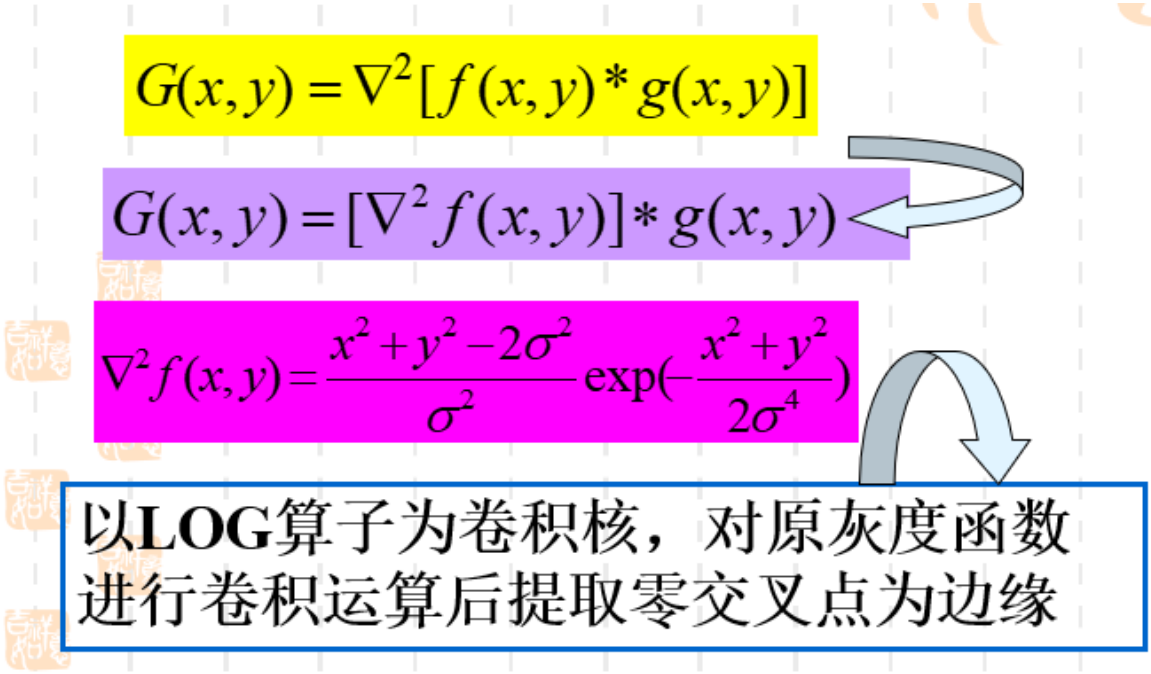
$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

标准差为σ的二维高斯函数为：

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

高斯拉普拉斯算子就是**首先用高斯函数先进行低通滤波，然后利用拉普拉斯算子进行高通滤波并提取零交叉点。**





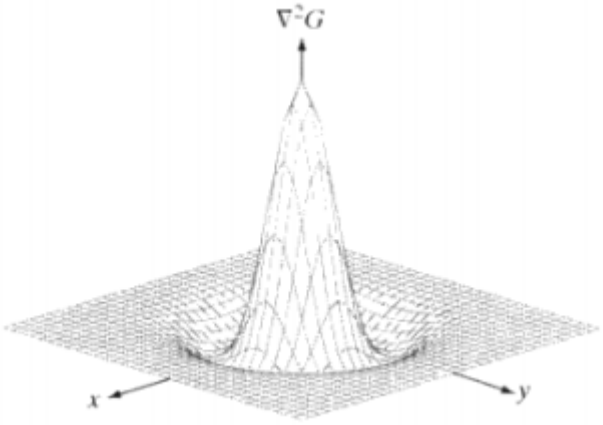
计算过程如上图所示。得到的最终的表达式为：

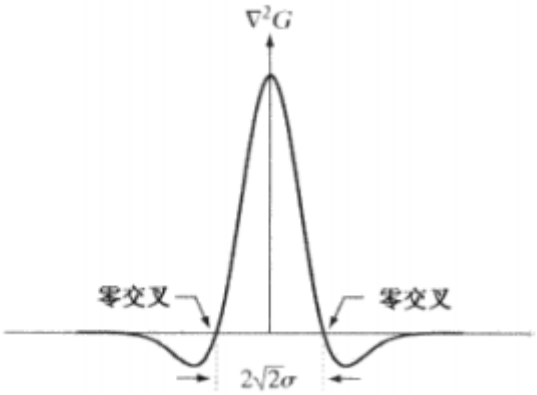
$$\nabla^2 G(x,\ y)=\frac{\partial^2 G(x,\ y)}{\partial x^2}+\frac{\partial^2 G(x,\ y)}{\partial y^2}=\frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}\right]+\frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}\right]$$

$$=\left[\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}+\left[\frac{y^2}{\sigma^4}-\frac{1}{\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\nabla^2 G(x,\ y)=\left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

LoG的负函数的三维图、图像和剖面图如图所示：





0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

注意LoG的零交叉出现在$x^2 + y^2 = \sigma^2$处，它定义了一个中心位于原点、半径为$\sqrt{2}\sigma$的圆。因为其形状，LoG算子又被称为**墨西哥草帽算子**。

图中还给出了一个近似的模板，但这个并不是唯一的。注意系数之和一定为0。

LoG的优点：

- 算子的高斯部分会模糊图像，从而在尺寸结构上将结构的灰度（包括噪声）降低到远小于σ的程度。高斯函数会在空间和频率两个域上平滑图像，从而在原图像中引入不存在的认为干扰的可能性很小。
- 对于一阶导数，是有方向的算子。但是拉普拉斯算子是二阶导数，且有各向同性（旋转不变）的优点，符合人的视觉系统，且对任何模板方向的灰度变化有相等的相应，从而避免了使用多个模板去计算图像中的任何点处的最强响应。

5.2 Marr-Hildreth边缘检测器

采用LoG滤波器和图像f(x,y)卷积组成：

$$g(x, y) = [\nabla^2 G(x, y)] \bigstar f(x, y)$$

然后寻找g(x,y)的零交叉来确定f(x,y)中边缘的位置。因为这些都是线性操作，因此上式也可写成：

$$g(x, y) = \nabla^2 [G(x, y) \bigstar f(x, y)]$$

上式的物理意义是先使用一个高斯滤波器去平滑图像，再计算改结果的拉普拉斯。

这就是上面彩图中显示的步骤。

关于高斯函数的σ和窗口大小的选取，可以参见这篇文章：https://blog.csdn.net/blogshinelee/article/details/82734769，以及数字图像处理第三版462页示例。

六、Canny检测器

Canny边缘检测是从不同视觉对象中提取有用的结构信息并大大减少要处理的数据量的一种技术，目前已广泛应用于各种计算机视觉系统。Canny发现，在**不同视觉系统上对边缘检测的要求较为类似**，因此，可以实现一种具有**广泛应用意义**的边缘检测技术。边缘检测的一般标准包括：

1. 以低的错误率检测边缘，也即意味着需要尽可能准确的捕获图像中尽可能多的边缘。
2. 检测到的边缘应精确定位在真实边缘的中心。
3. 图像中给定的边缘应只被标记一次，并且在可能的情况下，图像的噪声不应产生假的边缘。

为了满足这些要求，Canny使用了变分法。Canny检测器中的最优函数使用四个指数项的和来描述，它可以由高斯函数的一阶导数来近似。

Canny算法的步骤：

步骤	目的
1.高斯滤波	去噪声降低错误率
2.计算梯度幅值和方向	估计每一点处的边缘强度与方向
3.非极大值抑制（NMS）	对Sobel、Prewitt等算子的结果进一步细化
4应用双阈值（Double-Threshold）检测	确定真实的和可能的边缘。

为什么采用高斯滤波器呢？**平滑去噪和边缘检测是一对矛盾，应用高斯函数的一阶导数，在二者之间获得最佳的平衡。**

6.1 高斯滤波

令f(x,y)表示输入图像，G(x,y)表示高斯函数：

$$G(x,\ y)=\mathrm{e}^{\frac{x^2+y^2}{2\sigma^2}}$$

我们用G和f卷积形成一幅平滑后的图像$f_s(x,y)$：

$$f_s(x,\ y)=G(x,\ y)\ \bigstar\ f(x,\ y)$$

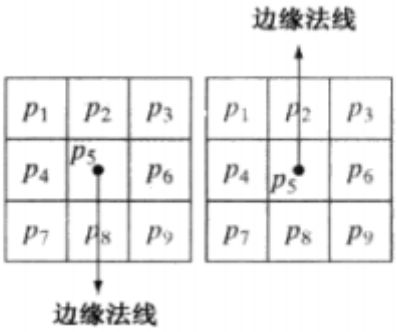
6.2 计算梯度幅值和方向

进行高斯滤波后，图像中的边缘可以指向各个方向，接下来使用四个算子来检测图像中的水平、垂直和对角边缘。边缘检测的算子（如Roberts，Prewitt，Sobel等）返回水平g_x和垂直g_y方向的一阶导数值，由此便可以确定像素点的梯度M和方向α。

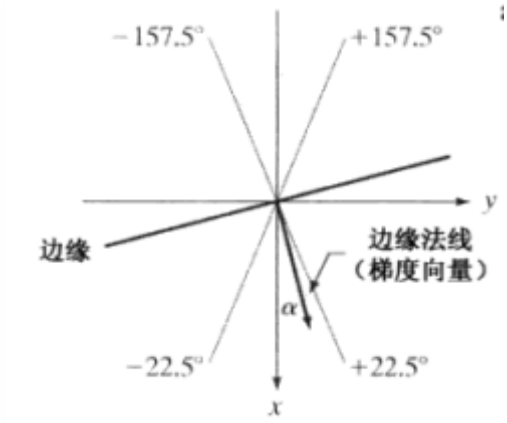
$$M(x,\ y)=\sqrt{g_x^2+g_y^2}$$

$$\alpha(x,\ y)=\arctan\left[\frac{g_y}{g_x}\right]$$

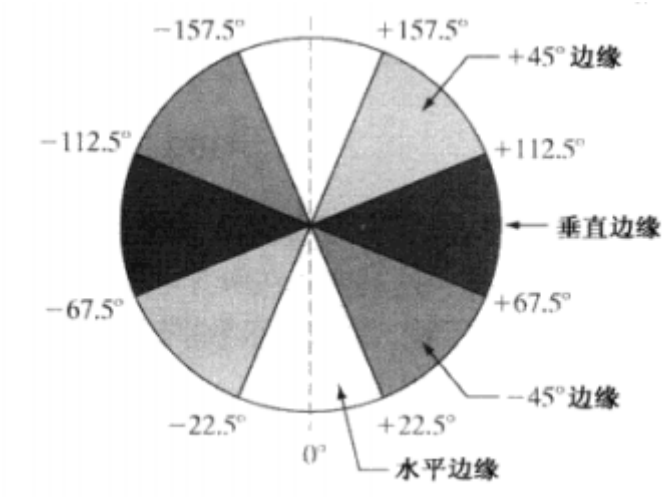
得到的边缘再局部最大值周围通常包含更宽的范围。例如，在一个3x3的区域内，对于一个通过该区域中心点的边缘，我们可以定义4个方向：水平、垂直、+45°和-45°。如图所示为一个水平边缘的两个可能方向：



因为我们必须把所有可能的边缘方向量化为四个方向，故需要定义一个方向范围。我们由**边缘法线的方向确定边缘方向。**如图所示的例子，如果边缘法线的方向是从-22.5°到22.5°，或者从-157.5°到157.5°，则我们称该边缘为水平边缘。



下图为按照此定义得到的各个边缘范围：



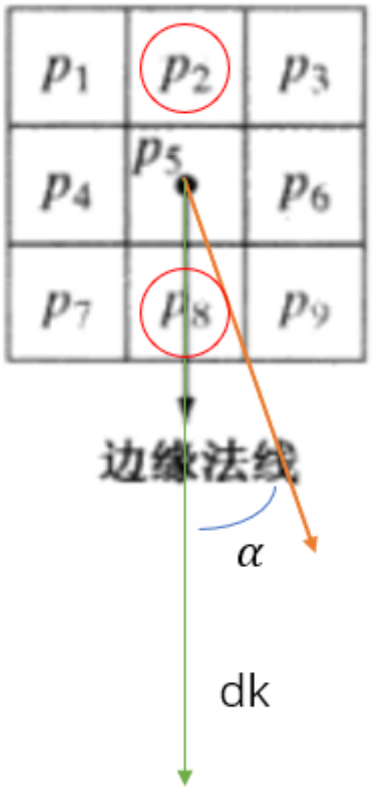
6.3 非极大值抑制

在每一点上，邻域中心与沿着其对应的梯度方向的两个像素相比，若中心像素为最大值，则保留，否则中心置0，这样可以抑制非极大值，保留局部梯度最大的点，以得到细化的边缘。

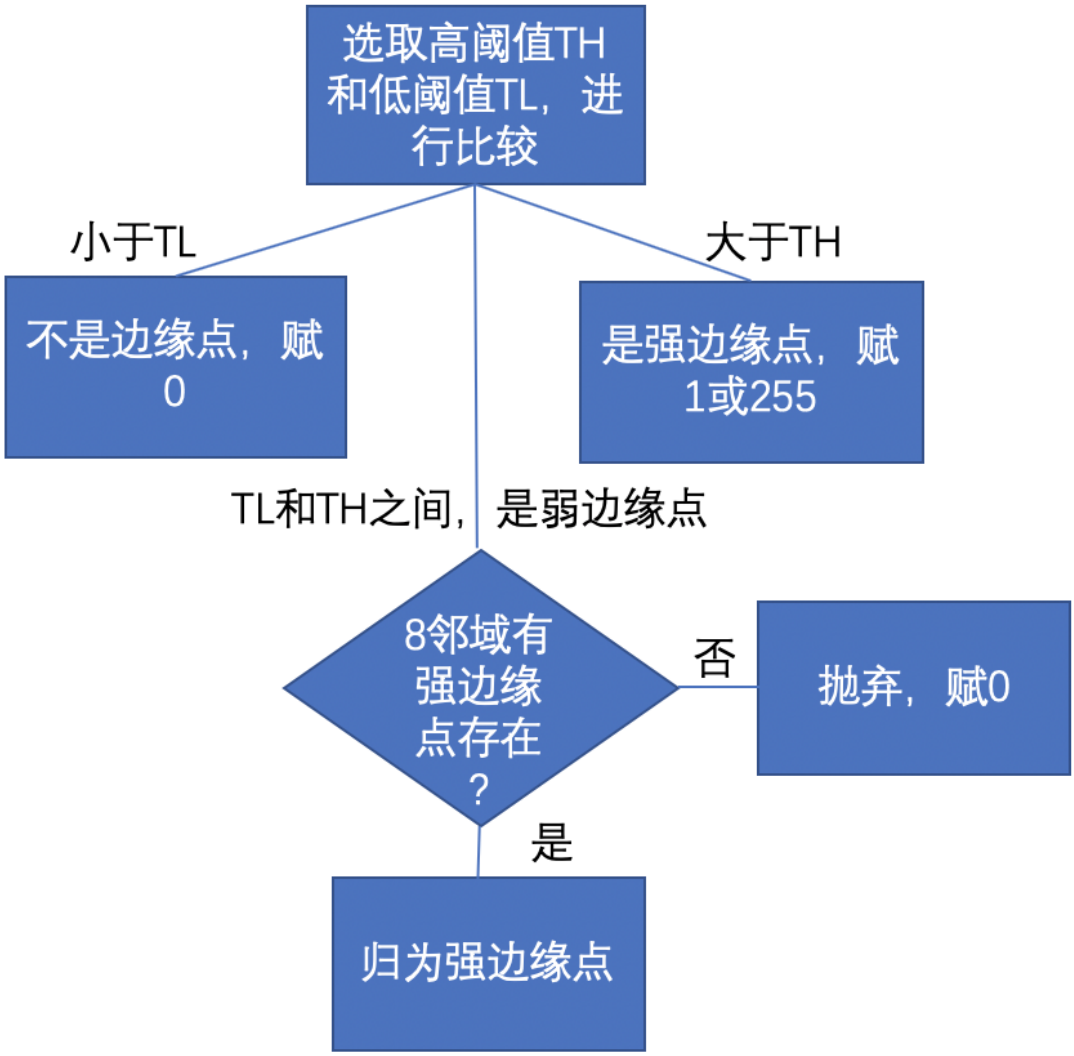
对图像进行梯度计算后，仅仅基于梯度值提取的边缘仍然很模糊。对边缘有且应当只有一个准确的响应。而非极大值抑制则可以帮助将局部最大值之外的所有梯度值抑制为0。非极大值抑制是一种边缘稀疏技术，非极大值抑制的作用在于“瘦”边。直观上地看，对第二步得到的图片，边缘由粗变细了。

令 d_1, d_2, d_3 和 d_4 表示刚才讨论的 3×3 区域的四个基本边缘方向：水平、−45°、垂直、+45°。对于在 $\alpha(x, y)$ 中以每一点 (x, y) 为中心的 3×3 区域，我们可给出如下非最大抑制方案：

1. 寻找最接近 $\alpha(x, y)$ 的方向 d_k。
2. 如果 $M(x, y)$ 的值至少小于沿 d_k 的两个邻居之一，则令 $g_N(x, y) = 0$（抑制）；否则，令 $g_N(x, y) = M(x, y)$，这里 $g_N(x, y)$ 是非最大抑制后的图像。例如，参考图 10.24(a)，令 (x, y) 在 p_5 处，并假设一个水平边缘通过 p_5，在步骤 2 中我们感兴趣的像素是 p_2 和 p_8。图像 $g_N(x, y)$ 仅包含细化后的边缘；它等于抑制了非最大边缘点的 $M(x, y)$。



6.4 双阈值算法检测和连接边缘



双阈值法非常简单，我们假设两类边缘：经过非极大值抑制之后的边缘点中，**梯度值超过TH的称为强边缘，梯度值小于TH大于TL的称为弱边缘，梯度小于TL的不是边缘。**

可以肯定的是，强边缘必然是边缘点，因此必须将T1设置的足够高，以要求像素点的梯度值足够大（变化足够剧烈），而弱边缘可能是边缘，也可能是噪声，如何判断呢？**当弱边缘的周围8邻域有强边缘点存在时，就将该弱边缘点变成强边缘点**，以此来实现对强边缘的补充。实际中人们发现T1:T2=2:1的比例效果比较好，其中T1可以人为指定，也可以设计算法来自适应的指定，比如定义梯度直方图的前30%的分界线为T1。检查8邻域的方法叫边缘滞后跟踪，连接边缘的办法还有区域生长法等等。

边缘的连接可以由下面步骤生成：

(a) 在 $g_{NH}(x,y)$ 中定位下一个未被访问的边缘像素 p。
(b) 在 $g_{NL}(x,y)$ 中将所有的弱像素标记为有效边缘像素，用 8 连通性的连接方法连接到 p。
(c) 如果 $g_{NH}(x,y)$ 中的所有非零像素已被访问，则跳到步骤(d)，否则返回步骤(a)。
(d) 将 $g_{NL}(x,y)$ 中未标记为有效边缘像素的所有像素置零。

在这一过程的末尾，将来自 $g_{NL}(x,y)$ 的所有非 0 像素附加到 $g_{NH}(x,y)$，用坎尼算子形成最终的输出图像。

其中

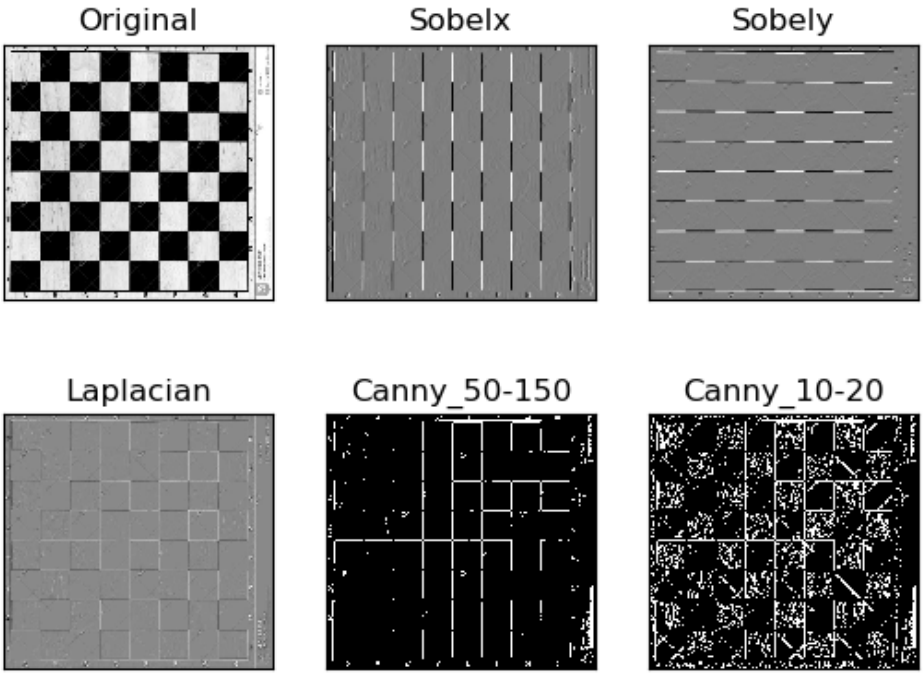
$$g_{NH}(x,y)=g_N(x,y)\geqslant T_H$$

$$g_{NL}(x,y)=g_N(x,y)\geqslant T_L$$

七、代码实现

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

def imgshow(name, source, number=0):
    if number != 0:
        plt.subplot(number)
    plt.imshow(source, cmap='gray')
    plt.title(str(name)), plt.xticks([]), plt.yticks([])

img = cv2.imread('E:/PythonProgram/opencv_study/fig_transaction/qipan.jpg', 0)

# 1、sobel算子
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5) # 64F代表每一个像素点元素占64位浮点数,通道数为1
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=5)

# 2、拉普拉斯算子
laplacian = cv2.Laplacian(img, cv2.CV_64F)

# 3、canny算子
gaussian = cv2.GaussianBlur(img, (3, 3), 0)
canny = cv2.Canny(gaussian, 50, 150)
canny2 = cv2.Canny(gaussian, 10, 20)

imgshow('Original', img, 231)
imgshow('Sobelx', sobelx, 232)
imgshow('Sobely', sobely, 233)
imgshow('Laplacian', laplacian, 234)
imgshow('Canny_50-150', canny, 235)
imgshow('Canny_10-20', canny2, 236)
plt.show()
```

得到结果:


Original
Sobelx
Sobely
Laplacian
Canny_50-150
Canny_10-20