资料来源：www.gwylab.com

【传统 CV 算法】图像边缘检测算法

目 录

资料来源：www.gwylab.com

第一章 概述

1. 边缘检测概述

1.1 认识边缘

边缘的定义

边缘是不同区域的分界线，是周围（局部）灰度值有显著变化的像素点的集合，有幅值与方向两个属性。这个不是绝对的定义，主要记住边缘是局部特征，以及周围灰度值显著变化产生边缘。

轮廓和边缘的关系

一般认为轮廓是对物体的完整边界的描述，边缘点一个个连接起来构成轮廓。边缘可以是一段边缘，而轮廓一般是完整的。人眼视觉特性，看物体时一般是先获取物体的轮廓信息，再获取物体中的细节信息，比如看到几个人站在那，我们一眼看过去马上能知道的是每个人的高矮胖瘦，然后才获取脸和衣着等信息。

边缘的类型

简单分为 4 种类型，阶跃型、屋脊型、斜坡型、脉冲型，其中阶跃型和斜坡型是类似的，只是变化的快慢不同，同样，屋脊型和脉冲型也是如此。在边缘检测中更多关注的是阶跃和屋脊型边缘。见图 1，（a）和（b）可认为是阶跃或斜坡型，（c）脉冲型，（d）屋脊型，阶跃与屋脊的不同在于阶跃上升或下降到某个值后持续下去，而屋脊则是先上升后下降。

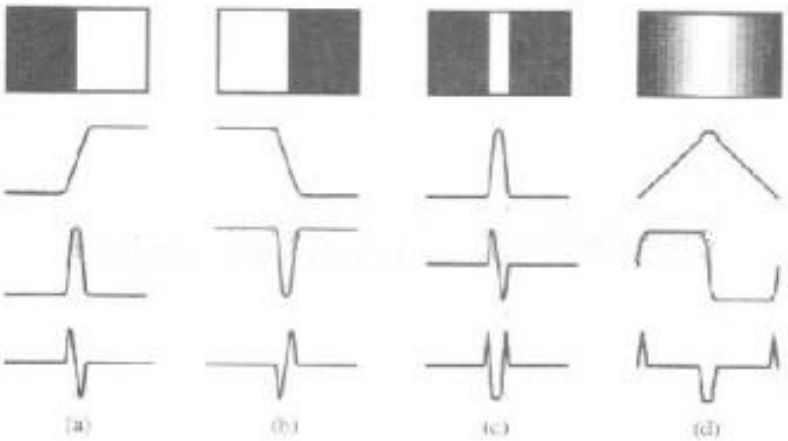
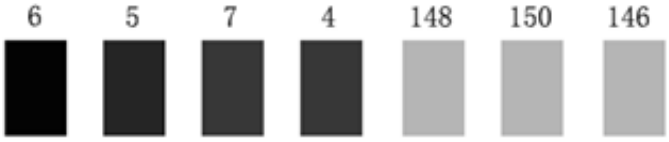


图 1

1.2 边缘检测的概念

边缘检测是图像处理与计算机视觉中极为重要的一种分析图像的方法，它的目的是找到图像中亮度变化剧烈的像素点构成的集合，表现出来往往是**轮廓**。如果图像中边缘能够精确的测量和定位，那么，就意味着实际的物体能够被定位和测量，包括物体的面积、物体的直径、物体的形状等就能被测量。在对现实世界的图像采集中，有下面 4 种情况会表现在图像中时形成一个边缘。

1. 深度的不连续（物体处在不同的物平面上）；
2. 表面方向的不连续（如正方体的不同的两个面）；
3. 物体材料不同（这样会导致光的反射系数不同）；
4. 场景中光照不同（如被树萌投向的地面）；



例如上面的图像是图像中水平方向 7 个像素点的灰度值显示效果，我们很容易地判断在第 4 和第 5 个像素之间有一个边缘，因为它俩之间发生了强烈的灰度跳变。在实际的边缘检测中，边缘远没有上图这样简单明显，我们需要取对应的阈值来区分出它们。

1.3 边缘检测的基本方法

一般图像边缘检测方法主要有如下四个步骤：

Step1. 图像滤波

传统边缘检测算法主要是基于图像强度的一阶和二阶导数，但导数的计算对噪声很敏感，因此必须使用滤波器来改善与噪声有关的边缘检测器的性能。需要指出的是，大多数滤波器在降低噪声的同时也造成了了边缘强度的损失，因此，在增强边缘和降低噪声之间需要一个折衷的选择。

Step2. 图像增强

增强边缘的基础是确定图像各点邻域强度的变化值。增强算法可以将邻域(或局部)强度值有显著变化的点突显出来。边缘增强一般是通过计算梯度的幅值来完成的。

Step3. 图像检测

在图像中有许多点的梯度幅值比较大，而这些点在特定的应用领域中并不都是边缘，所以应该用某种方法来确定哪些点是边缘点。最简单的边缘检测判断依据是梯度幅值。

Step4. 图像定位

如果某一应用场合要求确定边缘位置，则边缘的位置可在子像素分辨率上来估计，边缘的方位也可以被估计出来。

1.4 边缘检测算子的概念

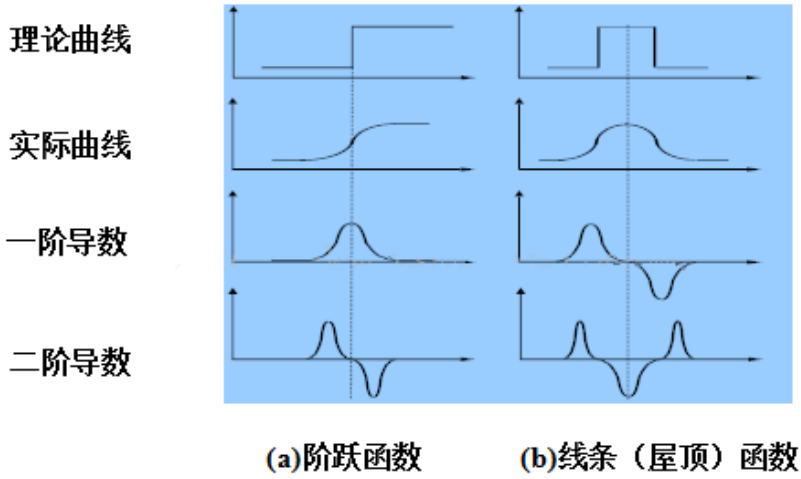


图 2

在数学中，函数的变化率由导数来刻画，图像我们看成二维函数，其上面的像素值变化，当然也可以用导数来刻画，当然图像是离散的，那我们换成像素的差分来实现。对于阶跃型边缘，图 2 中显示其一阶导数具有极大值，极大值点对应二阶导数的过零点，也就是，准确的边缘的位置是对应于一阶导数的极大值点，或者二阶导数的过零点（注意不仅仅是二阶导数为 0 值的位置，二值正负值过渡的零点）。故边缘检测算子的类型当然就存在一阶和二阶微分算子。

1.5 常见的边缘检测算子

近 20 多年来提出了许多边缘检测算子，在这里我们仅讨论集中常见的边缘检测算子。

原图	Roberts边缘	Prewitt边缘
		
Sobel边缘	Log边缘	Canny边缘
		

常见的一阶微分边缘算子包括 Roberts，Prewitt，Sobel，Kirsch 以及 Nevitia，常见的二阶微分边缘算子包括 Laplace 算子，LOG 算子，DOG 算子和 Canny 算子等。其中 Canny 算子是最为常用的一种，也是当前被认为最优秀的边缘检测算子。

此外本文还会介绍一种边缘检测方法 SUSAN，它没有用到图像像素的梯度（导数）。以及本文会概述一些新兴的边缘检测方法，如小波分析，模糊算法以及人工神经网络等。

2. 用梯度算子实现边缘检测的原理

2.1 认识梯度算子

边缘点

对应于一阶微分幅度的最大值点以及二阶微分的零点。

梯度的定义

一个曲面沿着给定方向的倾斜程度，在单变量函数中，梯度只是导数，在线性函数中，是线的斜率——有方向的向量。

梯度算子

梯度属于一阶微分算子，对应一阶导数。若图像含有较小的噪声并且图像边缘的灰度值过渡较为明显，梯度算子可以得到较好的边缘检测结果。

前篇介绍的 Roberts，Sobel 等算子都属于梯度算子。

2.2 梯度的衡量

对于连续函数 f(x,y)，我们计算出了它在(x,y)处的梯度，并且用一个矢量(沿 x 方向和沿 y 方向的两个分量)来表示，如下：

$$G(x,y)=\begin{bmatrix}G_x\\G_y\end{bmatrix}=\begin{bmatrix}\frac{\partial f}{\partial x}\\\frac{\partial f}{\partial y}\end{bmatrix}$$

现在我们需要衡量梯度的幅值，可以用到以下三种范数：

$$|G(x,y)|=\sqrt{G_x^2+G_y^2}\quad ,\quad 2\text{ 范数梯度}$$

$$|G(x,y)|=|G_x|+|G_y|\quad ,\quad 1\text{ 范数梯度}$$

$$|G(x,y)|\approx\max(|G_x|,|G_y|)\quad ,\quad \infty\text{范数梯度}$$

要注意的是，由于使用 2 范数梯度要对图像中的每个像素点进行平方及开方运算，计算复杂度高，在实际应用中，通常取绝对值或最大值来近似代替该运算以实现简化，与平方及开方运算相比，取绝对值或最大值进行的边缘检测的准确度和边缘的精度差异都很小。

2.3 使用梯度算子实现边缘检测

原理

基于梯度算子的边缘检测大多数是基于方向导数求卷积的方法

实现过程

以 3×3 的卷积模板为例。

Z_1	Z_2	Z_3
Z_4	Z_5	Z_6
Z_7	Z_8	Z_9

原始图像中的 3×3 子区域

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

3×3 卷积模板

设定好卷积模板后，将模板在图像中移动，并将图像中的每个像素点与此模板进行卷积，得到每个像素点的响应 R，用 R 来表征每个像素点的邻域灰度值变化率，即灰度梯度值，从而可将灰度图像经过与模板卷积后转化为梯度图像。模板系数W_i（i=1,2,3,……9）相加的总和必须为零，以确保在灰度级不变的区域中模板的响应为零。Z 表示像素的灰度值

$$R = W_1Z_1 + W_2Z_2 + \cdots + W_9Z_9$$

然后我们设定一个阈值，如果卷积的结果 R 大于这个阈值，那么该像素点为边缘点，输出白色；如果 R 小于这个阈值，那么该像素不为边缘点，输出黑色。于是最终我们就能输出一幅黑白的梯度图像，实现边缘的检测。

第二章 一阶微分边缘算子

1. 一阶微分边缘算子基本思想

一阶微分边缘算子也称为梯度边缘算子，它是利用图像在边缘处的阶跃性，即图像梯度在边缘取得极大值的特性进行边缘检测。梯度是一个矢量，它具有方向 θ 和模|ΔI|：

$$\Delta I = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix}$$

$$|\Delta I| = \sqrt{(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2} = \sqrt{I_x^2 + I_y^2}$$

$$\theta = arctan(I_y / I_x)$$

梯度的方向提供了边缘的趋势信息，因为梯度方向始终是垂直于边缘方向，梯度的模值大小提供了边缘的强度信息。

在实际使用中，通常利用有限差分进行梯度近似。对于上面的公式，我们有如下的近似：

$$\frac{\partial I}{\partial x} = \lim_{h \to 0} \frac{I(x + \Delta x, y) - I(x, y)}{\Delta x} \approx I(x+1, y) - I(x, y), (\Delta x = 1)$$

$$\frac{\partial I}{\partial y} = \lim_{h \to 0} \frac{I(x, y + \Delta xy) - I(x, y)}{\Delta y} \approx I(x, y+1) - I(x, y), (\Delta y = 1)$$

2. Roberts 算子

2.1 Roberts 算法思想

1963 年，Roberts 提出了这种寻找边缘的算子。Roberts 边缘算子是一个 2x2 的模板，采用的是对角方向相邻的两个像素之差。从图像处理的实际效果来看，边缘定位较准，对噪声敏感。

由 Roberts 提出的算子是一种利用局部差分算子寻找边缘的算子，边缘的锐利程度由图像灰度的梯度决定。梯度是一个向量，∇f 指出灰度变化的最快的方向和数量。

因此最简单的边缘检测算子是用图像的垂直和水平差分来逼近梯度算子：

$$\nabla f = \left(f(x,y) - f(x-1,y), f(x,y) - f(x,y-1) \right)$$

对每一个像素计算出以上式子的向量，求出它的绝对值，然后与阈值进行比较，利用这种思想就得到了 Roberts 交叉算子：

$$g(i,j) = \left| f(i,j) - f(i+1,j+1) \right| + \left| f(i,j+1) - f(i+1,j) \right|$$

2.2 Roberts 算法步骤

Roberts 算法过程非常简单。

选用 1 范数梯度计算梯度幅度：$|G(x,y)| = |G_x| + |G_y|$。

卷积模板为：

Roberts交叉算子模板			
G_x		G_y	
1	0	0	1
0	-1	-1	0

则模板运算结果：

$$G_x = 1*f(x,y)+0*f(x+1,y)+0*f(x,y+1)+(-1)*f(x+1,y+1)$$
$$= f(x,y)-f(x+1,y+1)$$

$$G_y = 0*f(x,y)+1*f(x+1,y)+(-1)*f(x,y+1)+0*f(x+1,y+1)$$
$$= f(x+1,y)-f(x,y+1)$$

$$G(x,y) = |G_x|+|G_y| = |f(x,y)-f(x+1,y+1)|+|f(x+1,y)-f(x,y+1)|$$

如果 G(x,y)大于某一阈值，则我们认为(x,y)点为边缘点。

2.3 Roberts 算子的推导

$$\frac{\partial f}{\partial x} \approx f(x+1,y)-f(x,y) \approx f(x+1,y+1)-f(x,y+1)$$
$$\frac{\partial f}{\partial y} \approx f(x,y+1)-f(x,y) \approx f(x+1,y+1)-f(x+1,y)$$
$$\|\nabla f(x,y)\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2+\left(\frac{\partial f}{\partial y}\right)^2} \approx \left|\frac{\partial f}{\partial x}\right|+\left|\frac{\partial f}{\partial y}\right|$$
$$\approx \frac{\partial f}{\partial x}+\frac{\partial f}{\partial y} \approx f(x+1,y)-f(x,y)+f(x+1,y+1)-f(x+1,y) = f(x+1,y+1)-f(x,y)$$
$$\approx \frac{\partial f}{\partial x}-\frac{\partial f}{\partial y} \approx f(x+1,y)-f(x,y)-\left(f(x,y+1)-f(x,y)\right) = f(x+1,y)-f(x,y+1)$$
$$\approx \frac{\partial f}{\partial y}-\frac{\partial f}{\partial x} \approx f(x,y+1)-f(x,y)-\left(f(x+1,y)-f(x,y)\right) = f(x,y+1)-f(x+1,y)$$
$$\approx -\frac{\partial f}{\partial x}-\frac{\partial f}{\partial y} \approx -\left(f(x+1,y)-f(x,y)\right)-\left(f(x+1,y+1)-f(x+1,y)\right) = f(x,y)-f(x+1,y+1)$$

2.4 Roberts 算法优缺点

Roberts 算子利用局部差分算子寻找边缘，边缘定位精度高，但容易丢失一部分边缘，同时由于图像没有经过平滑处理，因此不具备抑制噪声能力。该算子对具有陡峭边缘且含有噪声少的图像处理效果较好。另外，由于该检测算子模板没有中心点，所以它在实际中很少使用。

3. Prewitt 算子

3.1 Prewitt 算法思想

Prewitt 算子是 J.M.S.Prewitt 于 1970 年提出的检测算子。不同于 Roberts 算子采用 2*2 大小的模板，Prewittt 算法采用了 3*3 大小的卷积模板。2*2 大小的模板在概念上很简单，但是他们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，其最小模板大小为 3*3。3*3 模板考虑了中心点对段数据的性质，并携带有关于边缘方向的更多信息。

在算子的定义上，Prewitt 希望通过使用水平+垂直的两个有向算子去逼近两个偏导数 G_x，G_y,这样在灰度值变化很大的区域上卷积结果也同样达到极大值。

1	1	1
0	0	0
-1	-1	-1

水平方向

1	0	-1
1	0	-1
1	0	-1

垂直方向

3.2 Prewitt 算法步骤

Prewitt 边缘检测算子使用两个有向算子(水平+垂直)，每一个逼近一个偏导数，这是一种类似计算偏微分估计值的方法，x，y 两个方向的近似检测算子为：、

$$\begin{aligned} p_x &= \{f(i+1,j-1)+f(i+1,j)+f(i+1,j+1)\} \\ &\quad -\{f(i-1,j-1)+f(i-1,j)+f(i-1,j+1)\} \\ p_y &= \{f(i-1,j+1)+f(i,j+1)+f(i+1,j+1)\} \\ &\quad -\{f(i-1,j-1)+f(i,j-1)+f(i+1,j-1)\} \end{aligned}$$

得出卷积模板为：

$$G_x = \begin{vmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{vmatrix} \qquad G_y = \begin{vmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{vmatrix}$$

记图像 M，梯度幅值 T，比较

$$T = (M \otimes G_x)^2 + (M \otimes G_y)^2 > threshold$$

如果最终 T 大于阈值 threshold，那么该点为边缘点。

3.3 Prewitt 算法优缺点

Prewitt 算子对灰度渐变和噪声较多的图像处理效果较好，但不能完全排除检测结果中出现的虚假边缘。

4. Sobel 算子

4.1 Sobel 算法思想

Sobel 最初是 1968 年在一次博士生课题讨论会 1968 上提出("A 3x3 Isotropic Gradient Operator for Image Processing")，后来正式出版发表是在 1973 年的一本专著（"Pattern Classification and Scene Analysis"）的脚注里作为注释出现和公开的。Sobel 算子和 Prewitt 算子都是加权平均，但是 Sobel 算子认为，邻域的像素对当前像素产生的影响不是等价的，所以距离不同的像素具有不同的权值，对算子结果产生的影响也不同。一般来说，距离越远，产生的影响越小。

将 Prewitt 边缘检测算子模板的中心系数增加一个权值 2，不但可以突出中心像素点，而且可以得到平滑的效果，这就成为索贝尔算子。

4.2 Sobel 算法步骤

Sobel 算子一种将方向差分运算与局部平均相结合的方法。该算子是在以 f(x,y)为中心的 3x3 邻域上计算 x 和 y 方向的偏导数，即

$$\begin{aligned} p_x &= \{f(i+1,j-1)+2f(i+1,j)+f(i+1,j+1)\} \\ &\quad -\{f(i-1,j-1)+2f(i-1,j)+f(i-1,j+1)\} \\ p_y &= \{f(i-1,j+1)+2f(i,j+1)+f(i+1,j+1)\} \\ &\quad -\{f(i-1,j-1)+2f(i,j-1)+f(i+1,j-1)\} \end{aligned}$$

得出卷积模板为：

$$G_x = \begin{vmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{vmatrix} \qquad G_y = \begin{vmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{vmatrix}$$

记图像 M，梯度幅值 T，比较

$$T = (M \otimes G_x)^2 + (M \otimes G_y)^2 > threshold$$

如果最终 T 大于阈值 threshold，那么该点为边缘点。

4.3 Sobel 算法优缺点

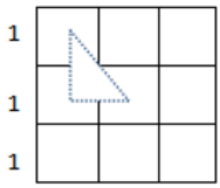
Sebel 有明显的两个优点：一是高频的像素点少，低频的像素点多，使像素的灰度平均值下降，且由于噪声一般为高频信号，所以它具有抑制噪声的能力；二是检测到的边缘比较宽，至少具有两个像素的边缘宽度，从而使其得到了广泛应用。但是不能完全排除检测结果中出现的虚假边缘。

4.4 *Sobel 的变种——Istropic Sobel

Sobel 算子还有一种变种 Istropic Sobel，是各向同性 Sobel 算子，其模板为

-1	$-\sqrt{2}$	-1
0	0	0
1	$\sqrt{2}$	1

-1	0	1
$-\sqrt{2}$	0	$\sqrt{2}$
-1	0	1



Sobel 各向同性算子的权值比普通 Sobel 算子的权值更准确。为什么？模板的权值是离中心位置越远则权值（看绝对值）影响越小，如上图，把模板看成是 9 个小正方形，小正方形边长为 1，则虚线三角形的斜边长为$\sqrt{2}$，下直角边长为 1，则如果（0,0）位置权值绝对值大小为 1，则按照距离关系，位置（1,0）处的权值绝对值大小应该为$\sqrt{2}$才是准确的

5. Kirsch 算子

5.1 Kirsch 算法思想

Kirsch 算子是 R.Kirsch 提出来的一种边缘检测算法。与前述的算法不同之处在于，Kirsch 考虑到 3*3 的卷积模板事实上涵盖着 8 种方向（左上，正上，……，右下），于是 Kirsch 采用 8 个 3*3 的模板对图像进行卷积，这 8 个模板代表 8 个方向，并取最大值作为图像的边缘输出。

5.2 Kirsch 算法步骤

它采用下述 8 个模板对图像上的每一个像素点进行卷积求导数：

$$K_N=\begin{bmatrix}5&5&5\\-3&0&3\\-3&-3&-3\end{bmatrix},K_{NE}=\begin{bmatrix}-3&5&5\\-3&0&5\\-3&-3&-3\end{bmatrix}$$

$$K_E=\begin{bmatrix}-3&-3&5\\-3&0&5\\-3&-3&5\end{bmatrix},K_{SE}=\begin{bmatrix}-3&-3&-3\\-3&0&5\\-3&5&5\end{bmatrix}$$

$$K_S=\begin{bmatrix}-3&-3&-3\\-3&0&-3\\5&5&5\end{bmatrix},K_{SW}=\begin{bmatrix}-3&-3&-3\\5&0&-3\\5&5&-3\end{bmatrix}$$

$$K_W=\begin{bmatrix}5&-3&-3\\5&0&-3\\5&-3&-3\end{bmatrix},K_{NW}=\begin{bmatrix}5&5&-3\\5&0&-3\\-3&-3&-3\end{bmatrix}$$

最终选取 8 次卷积结果的最大值作为图像的边缘输出。

5.3 Kirsch 算法计算优化

假设图像中一点 A 及其周围 3×3 区域的灰度如下图所示，设q_i（i=0,1,……,7）为图像经过 Kirsch 算子第（i+1）个模板处理后得到的 A 点的灰度值。

$$\begin{matrix}p_0&p_1&p_2\\p_7&p_A&p_3\\p_6&p_5&p_4\end{matrix}$$

处理后在 A 点的灰度值为：

$$q_A = \max\{q_i\} \qquad (i=0,1,\cdots\cdots,7)$$

通过矩阵变换发现经过 Kirsch 算子得到的像素值直接的关系，事实上需要直接由邻域像素点计算得到的只有p_0，，因此可以大大减少计算量。

$$Q = 1/8(q_0,q_1,\cdots,q_7)^T = (r_0,r_1,\cdots,r_7)^T$$

$$r_0=0.625(p_0+p_1+p_2)-0.375(p_3+p_4+p_5+p_6+p_7)$$

$$r_1=r_0+p_7-p_2 \qquad r_2=r_1+p_6-p_1 \qquad r_3=r_2+p_5-p_0$$

$$r_4=r_3+p_4-p_7 \qquad r_5=r_4+p_3-p_6 \qquad r_6=r_5+p_2-p_5$$

$$r_7=r_6+p_1-p_4$$

$$q_A = max\{q_i\} = 8max\{r_i\} \qquad (i = 0,1,\cdots,7)$$

5.4 Kirsch 算法优缺点

Kirsch 算子属于模板匹配算子，采用八个模板来处理图像的检测图像的边缘，运算量比较大，但是它在保持细节和抗噪声方面都有较好的效果。

5.5 *Robinson 算子

Robinson 与 Kirsch 算子非常类似，也是 8 个模板，只是模板的内容有差异：

$$R_N=\begin{bmatrix}1&2&1\\0&0&0\\-1&-2&1\end{bmatrix},R_{NE}=\begin{bmatrix}0&1&2\\-1&0&1\\-2&-1&0\end{bmatrix}$$

$$R_E=\begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix},R_{SE}=\begin{bmatrix}-2&-1&0\\-1&0&1\\0&1&2\end{bmatrix}$$

$$R_S=\begin{bmatrix}-1&-2&-1\\0&0&0\\1&2&1\end{bmatrix},R_{SW}=\begin{bmatrix}0&-1&-2\\1&0&-1\\2&1&0\end{bmatrix}$$

$$R_W=\begin{bmatrix}1&0&-1\\2&0&-2\\1&0&-1\end{bmatrix},R_{NW}=\begin{bmatrix}2&1&0\\1&0&-1\\0&-1&-2\end{bmatrix}$$

6. *Nevitia 算子

Nevitia 算子采用的是 5×5 的卷积模板，它与 Kirsch 类似，也是属于方向算子，总共能确定 4*（5-1）=12 个方向，因此包含有 12 个 5×5 的模板。

其中前 6 个模板如下，后 6 个模板可以由对称性得到。

-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100

-100	32	100	100	100
-100	-78	92	100	100
-100	-100	0	100	100
-100	-100	-92	78	100
-100	-100	-100	-32	100

100	100	100	100	100
-32	78	100	100	100
-100	-92	0	92	100
-100	-100	-100	-78	32
-100	-100	-100	-100	-100

100	100	100	100	100
100	100	100	100	100
0	0	0	0	0
-100	-100	-100	-100	-100
-100	-100	-100	-100	-100

100	100	100	100	100
100	100	100	78	32
100	92	0	-92	-100
-32	-78	-100	-100	-100
-100	-100	-100	-100	-100

100	100	100	32	-100
100	100	92	-78	-100
100	100	0	-100	-100
100	78	-92	-100	-100
100	-32	-100	-100	-100

5×5 Nevitia 算子的前 6 个模板

最终选取 12 次卷积结果的最大值作为图像的边缘输出。

第三章 二阶微分边缘算子

1. 二阶微分边缘算子基本思想

学过微积分我们都知道，边缘即是图像的一阶导数局部最大值的地方，那么也意味着该点的二阶导数为零。二阶微分边缘检测算子就是利用图像在边缘处的阶跃性导致图像二阶微分在边缘处出现零值这一特性进行边缘检测的。

对于图像的二阶微分可以用拉普拉斯算子来表示：

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

我们在像素点(i,j)的 3×3 的邻域内，可以有如下的近似：

$$\frac{\partial^2 I}{\partial x^2} = I(i,j+1) - 2I(i,j) + I(i,j-1)$$

$$\frac{\partial^2 I}{\partial y^2} = I(i+1,j) - 2I(i,j) + I(i-1,j)$$

$$\nabla^2 I = -4I(i,j) + I(i,j+1) + I(i,j-1) + I(i+1,j) + I(i-1,j)$$

对应的二阶微分卷积核为

$$\boldsymbol{m} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

所以二阶微分检测边缘的方法就分两步：1）用上面的 Laplace 核与图像进行卷积；2）对卷积后的图像，取得那些卷积结果为 0 的点。

2. Laplace 算子

2.1 拉普拉斯表达式

Laplace （拉普拉斯）算子是最简单的各向同性微分算子，一个二维图像函数的拉普拉斯变换是各向同性的二阶导数。下式为 Laplace 算子的表达式：

$$\nabla^2 f(x,y)=\frac{\partial^2 f(x,y)}{\partial x^2}+\frac{\partial^2 f(x,y)}{\partial y^2}$$

我们知道梯度的表达式为$\nabla=\frac{\partial}{\partial x}\vec{i}+\frac{\partial}{\partial y}\vec{j}$，于是上式的推导过程就是：

$$\nabla^2 \triangleq \nabla\cdot\nabla=(\frac{\partial}{\partial x}\vec{i}+\frac{\partial}{\partial y}\vec{j})\cdot(\frac{\partial}{\partial x}\vec{i}+\frac{\partial}{\partial y}\vec{j})=\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}$$

2.2 图像中的 Laplace 算子

考虑到图像是离散的二维矩阵，用差分近似微分可以得到：

$$\frac{\partial^2 f}{\partial x^2}=\frac{\partial G_x}{\partial x}=\frac{\partial[f(i,j)-f(i,j-1)]}{\partial x}=\frac{\partial f(i,j)}{\partial x}-\frac{\partial f(i,j-1)}{\partial x}$$
$$=[f(i,j+1)-f(i,j)]-[f(i,j)-f(i,j-1)]$$
$$=f(i,j+1)-2f(i,j)+f(i,j-1)$$

同理，可得：

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

于是有：

$$\nabla^2 f=\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial x^2}=f(i+1,j)+f(i-1,j)+f(i,j+1)+f(i,j-1)-4f(i,j)$$

用模板来表示为：

$$\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}$$

还有一种常用的卷积模板为：

$$\begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix}$$

有时我们为了在邻域中心位置取到更大的权值，还使用如下卷积模板：

$$\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

2.3 Laplace 算法过程

（1）遍历图像（除去边缘，防止越界），对每个像素做 Laplancian 模板卷积运算，注意是只做其中的一种模板运算，并不是两个。

（2）复制到目标图像，结束。

2.4 Laplace 算子的旋转不变性证明

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

$$= \frac{\partial}{\partial x'}\left(\frac{\partial f}{\partial x'}\right) + \frac{\partial}{\partial y'}\left(\frac{\partial f}{\partial y'}\right)$$

$$= \frac{\partial}{\partial x'}\left(\frac{\partial f}{\partial x}\frac{\partial x}{\partial x'} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial x'}\right) + \frac{\partial}{\partial y'}\left(\frac{\partial f}{\partial x}\frac{\partial x}{\partial y'} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial y'}\right)$$

$$= \frac{\partial}{\partial x'}\left(\frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta\right) + \frac{\partial}{\partial y'}\left(-\sin\theta\frac{\partial f}{\partial x} + \cos\frac{\partial f}{\partial y}\right)$$

$$= \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta\right)\frac{\partial x}{\partial x'} + \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta\right)\frac{\partial y}{\partial x'}$$
$$+ \frac{\partial}{\partial x}\left(-\sin\theta\frac{\partial f}{\partial x} + \cos\frac{\partial f}{\partial y}\right)\frac{\partial x}{\partial y'} + \frac{\partial}{\partial y}\left(-\sin\theta\frac{\partial f}{\partial x} + \cos\frac{\partial f}{\partial y}\right)\frac{\partial y}{\partial y'}$$

$$= \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta\right)\cos\theta + \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta\right)\sin\theta$$
$$+ \frac{\partial}{\partial x}\left(-\sin\theta\frac{\partial f}{\partial x} + \cos\frac{\partial f}{\partial y}\right)(-\sin\theta) + \frac{\partial}{\partial y}\left(-\sin\theta\frac{\partial f}{\partial x} + \cos\theta\frac{\partial f}{\partial y}\right)\cos\theta$$

$$= \frac{\partial}{\partial x}\frac{\partial f}{\partial x} + \frac{\partial}{\partial y}\frac{\partial f}{\partial y}$$

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

2.4 Laplace 算子优缺点

Laplace 算子具有旋转不变性，适用于仅注重边缘点所处的具体位置，而对边缘点附近的实际灰度差没有要求的情况。不过在边缘检测中并不常用拉普拉斯算子，而主要是用在判断像素是在边缘亮的一面还是暗的一面，主要有以下三点原因：

(1) 二阶导数算子与一阶导数算子相比，去除噪声的能力更弱；

(2) 该算子对边缘的方向检测不到；

(3) 该算子的幅值会产生双边元。

为了使图像的抗噪能力提高，人们提出了下节将要介绍的改进的 LOG(Laplacian of Gaussian) 算子。

3. LOG 算子

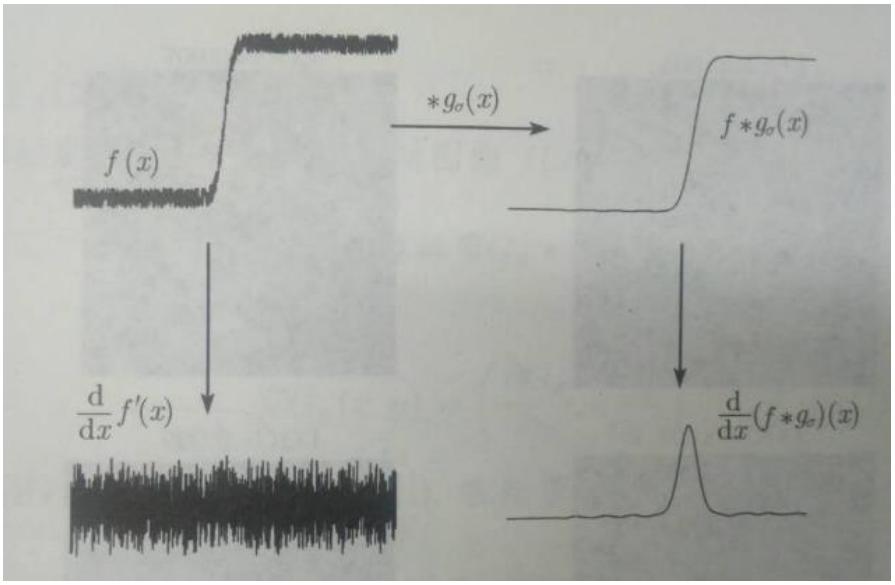
3.1 LoG 算子概述

1980 年，Marr 和 Hildreth 提出将 Laplace 算子与高斯低通滤波相结合，提出了 LOG（Laplace and Guassian）算子，又称为马尔（Marr）算子。

该算子是先运用高斯滤波器平滑图像达到去除噪声的目的，然后用 Laplace 算子对图像边缘进行检测。这样既达到了降低噪声的效果，同时也使边缘平滑，并得到了延展。为了防止得到不必要的边缘，边缘点应该选取比某阈值高的一阶导数零交叉点。 LOG 算子已经成为目前对阶跃边缘用二阶导数过零点来检测的最好的算子。

3.2 LoG 解决的问题

前篇当中说到直接使用 Laplace 算法，去噪的能力是非常弱的，最终提取出的边缘信息极易被噪音干扰，如下图所示：



由图可以看出，LoG 解决的最重要的问题是在提取边缘信息之前给原始图像增添一层高斯滤波器，使得原始图像的梯度异常点大大减少，最终能够更有效地提取出边缘信息。

3.3 LoG 算子的计算过程

在图像处理中，常用的二维高斯函数为

$$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}e^{-(x^2+y^2)/2\sigma^2}$$

拉普拉斯算子为

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

对二维高斯函数应用拉普拉斯算子得

$$\nabla^2 G = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2} = \frac{-2\sigma^2 + x^2 + y^2}{2\pi\sigma^6} e^{-(x^2+y^2)/2\sigma^2}$$

LoG（Laplacian of Gaussian）算子定义为

$$LoG = \sigma^2 \nabla^2 G$$

注意到 LoG 算子，该函数轴对称，且函数的平均值为 0（在定义域内），所以用它与图像进行卷积计算并不会对完整的图像动态区域有所改变，而是会使图像变模糊（因为它相当平滑），且模糊程度与σ成正比。当σ较大时，高斯滤波起到了很好的平滑效果，较大程度地抑制了噪声，但同时使一些边缘细节丢失，降低了图像边缘的定位精度；当σ比较小时，有很强的图像边缘定位精度，反而信噪比较低。因此，在应用 LOG 算子时，如何选取适当的σ值很重要，要根据边缘的定位精度的需求以及噪声情况而定

3.4 LoG 的卷积模板

LOG 算子的卷积模板通常采用 5×5 的矩阵，如：

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \text{和} \begin{bmatrix} -2 & -4 & -4 & -4 & -2 \\ -4 & 0 & 8 & 0 & -4 \\ -4 & 8 & 24 & 8 & -4 \\ -4 & 0 & 8 & 0 & -4 \\ -2 & -4 & -4 & -4 & -2 \end{bmatrix}$$

这个模板的系数是如何计算出来的呢？见如下的代码：

```
LoG( ,size):= cx ← (size-1)/2
              cy ← (size-1)/2
              for x∈0.. size-1
                for y∈0.. size-1
                  nx ← x-cx
                  ny ← y-cy
                  template_{y,x} ← (1/2)·((nx^2+ny^2)/2 - 2)·e^(-((nx^2+ny^2)/(2·2^2)))
              template ← normalize(template)
              template
```

其中左边这个模板是在高斯标准差为 0.5 时的逼近，右边的模板的高斯标准差为 1。Normalize 是一个正则函数，它确保模板系数的总和为 1. 以便在均匀亮度区域不会检测到边缘。最后由于整数比浮点数更易于计算，模板的系数都会近似为整数。

3.5 LoG 算法过程

（1）遍历图像（除去边缘，防止越界），对每个像素做 Gauss -Laplancian 模板卷积运算。

（2）复制到目标图像，结束。

3.6 DoG 与 LoG

由于数学上的关系，我们可以简化 LOG 的计算——这便是 DOG 算子。

DoG（Difference of Gaussian）算子定义为

$$DoG = G(x, y, \sigma_1) - G(x, y, \sigma_2)$$

下面我们来证明为何 DoG 能近似替代 LoG。

对二维高斯函数关于 σ 求一阶偏导数得

$$\frac{\partial G}{\partial \sigma} = \frac{-2\sigma^2 + x^2 + y^2}{2\pi\sigma^5} e^{-(x^2+y^2)/2\sigma^2}$$

不难发现

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G$$

在 DoG 算子中，令 σ1=kσ2=kσ，则

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

进一步地

$$\frac{\partial G}{\partial \sigma} = \lim_{\Delta\sigma \to 0} \frac{G(x, y, \sigma + \Delta\sigma) - G(x, y, \sigma)}{(\sigma + \Delta\sigma) - \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

因此

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

即

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G$$

这表明 DoG 算子可以近似于 LoG 算子，证毕。

于是，使用 DoG 算子，让我们只需对图像进行两次高斯平滑再将结果相减就可以近似得到 LOG 作用于图像的效果了。

值得注意的是，当我们用 DOG 算子代替 LOG 算子与图像卷积的时候：

$$DoG(x, y, \sigma_1, \sigma_2) * I(x, y) = G(x, y, \sigma_1) * I(x, y) - G(x, y, \sigma_2) * I(x, y)$$

近似的 LOG 算子值的选取：

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_2^2} ln[\frac{\sigma_1^2}{\sigma_2^2}]$$

当使用这个值时，可以保证 LoG 和 DoG 的过零点相同，只是幅度大小不同。

3.7 LoG 算子优缺点

值得肯定的是，LoG 这种方法寻找二阶导数是很稳定的。高斯平滑有效地抑制了距离当前像素 3σ 范围内的所有像素的影响，这样 Laplace 算子就构成了一种反映图像变化的有效而稳定的度量。

但是，这种传统的二阶导数过零点技术也有缺点。第一，对形状作了过分的平滑。第二，它有产生环形边缘的倾向。

4. Canny 算子

4.1 Canny 算子概述

Canny 边缘检测算法是 1986 年有 John F. Canny 开发出来一种基于图像梯度计算的边缘检测算法，同时 Canny 本人对计算图像边缘提取学科的发展也是做出了很多的贡献。尽管至今已经许多年过去，但是该算法仍然是图像边缘检测方法经典算法之一。

Canny 根据以前的边缘检测算子以及应用，归纳了如下三条准则：

(1) 信噪比准则：避免真实的边缘丢失，避免把非边缘点错判为边缘点；

(2) 定位精度准则：得到的边缘要尽量与真实边缘接近；

(3) 单一边缘响应准则：单一边缘需要具有独一无二的响应，要避免出现多个响应，并最大抑制虚假响应。

以上三条准则是由 Canny 首次明确提出并对这个问题进行完全解决的，虽然在他之前有人提出过类似的要求。更为重要的是， Canny 同时给出了它们的数学表达式（现以一维为例），这就转化成为一个泛函优化的问题

4.2 Canny 算子检测步骤

经典的 Canny 边缘检测算法通常都是从高斯模糊开始，到基于双阈值实现边缘连接结束。但是在实际工程应用中，考虑到输入图像都是彩色图像，最终边缘连接之后的图像要二值化输出显示，所以完整的 Canny 边缘检测算法实现步骤如下：

1. 彩色图像转换为灰度图像

2. 对图像进行高斯模糊

3. 计算图像梯度，根据梯度计算图像边缘幅值与角度

4. 非极大值抑制（边缘细化）

5. 双阈值检测

6. 通过抑制孤立的弱边缘完成边缘检测

7. 二值化图像输出结果

4.3 Canny 算子各步骤详解

1. 彩色图像转换为灰度图像

根据彩色图像 RGB 转灰度公式：gray = R * 0.299 + G * 0.587 + B * 0.114。

2. 对图像进行高斯模糊

为了尽可能减少噪声对边缘检测结果的影响，所以必须滤除噪声以防止由噪声引起的错误检测。为了平滑图像，使用高斯滤波器与图像进行卷积，该步骤将平滑图像，以减少边缘检测器上明显的噪声影响。大小为(2k+1)x(2k+1)的高斯滤波器核的生成方程式由下式给出：

$$H_{ij}=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{\left(i-(k+1)\right)^2+\left(j-(k+1)\right)^2}{2\sigma^2}\right);1\le i,j\le(2k+1) \quad (3\text{-}1)$$

下面是一个 sigma = 1.4，尺寸为 3x3 的高斯卷积核的例子（需要注意归一化）：

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

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：

$$e=H*A=\begin{bmatrix}h_{11} & h_{12} & h_{13}\\ h_{21} & h_{22} & h_{23}\\ h_{31} & h_{32} & h_{33}\end{bmatrix}*\begin{bmatrix}a & b & c\\ d & e & f\\ g & h & i\end{bmatrix}=sum\left(\begin{bmatrix}a\times h_{11} & b\times h_{12} & c\times h_{13}\\ d\times h_{21} & e\times h_{22} & f\times h_{23}\\ g\times h_{31} & h\times h_{32} & i\times h_{33}\end{bmatrix}\right)$$

其中*为卷积符号，sum 表示矩阵中所有元素相加求和。

重要的是需要理解，高斯卷积核大小的选择将影响 Canny 检测器的性能。尺寸越大，检测器对噪声的敏感度越低，但是边缘检测的定位误差也将略有增加。一般 5x5 是一个比较不错的 trade off。

3. 计算图像梯度，根据梯度计算图像边缘幅值与角度

图像中的边缘可以指向各个方向，因此 Canny 算法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 Gx 和垂直 Gy 方向的一阶导数值，由此便可以确定像素点的梯度 G 和方向 theta 。

$$G=\sqrt{G_x^2+G_y^2}$$

$$\theta=arc\tan\left(G_y/G_x\right)$$

其中 G 为梯度强度， theta 表示梯度方向，arctan 为反正切函数。

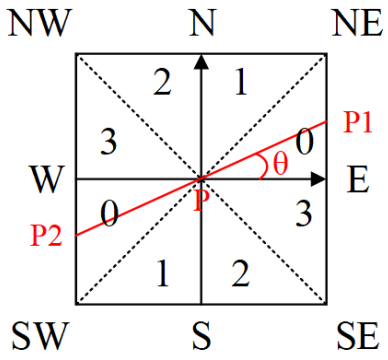
4. 非极大值抑制（边缘细化）

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。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。对于标准 3，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，对梯度图像中每个像素进行非极大值抑制的算法是：

1) 将当前像素的梯度强度与沿正负梯度方向上的两个像素进行比较。

2) 如果当前像素的梯度强度与另外两个像素相比最大，则该像素点保留为边缘点，否则该像素点将被抑制。

通常为了更加精确的计算，在跨越梯度方向的两个相邻像素之间使用线性插值来得到要比较的像素梯度，现举例如下：



如上图所示，将梯度分为 8 个方向，分别为 E、NE、N、NW、W、SW、S、SE，其中 0 代表 0°~45°,1 代表 45°~90°，2 代表-90°~-45°，3 代表-45°~0°。像素点 P 的梯度方向为 theta，则像素点 P1 和 P2 的梯度线性插值为：

$$\tan(\theta) = G_y / G_x$$

$$G_{p1} = (1 - \tan(\theta)) \times E + \tan(\theta) \times NE$$

$$G_{p2} = (1 - \tan(\theta)) \times W + \tan(\theta) \times SW$$

因此非极大值抑制的伪代码描写如下：

$$\begin{aligned} &if \;\; G_p \geq G_{p1} \;\; and \;\; G_p \geq G_{p2} \\ &\quad G_p \; may \; be \; an \; edge \\ &else \\ &\quad G_p \;\; should \;\; be \;\; \sup pressed \end{aligned}$$

需要注意的是，如何标志方向并不重要，重要的是梯度方向的计算要和梯度算子的选取保持一致。

5．双阈值检测

在施加非极大值抑制之后，剩余的像素可以更准确地表示图像中的实际边缘。然而，仍然存在由于噪声和颜色变化引起的一些边缘像素。为了解决这些杂散响应，必须用弱梯度值过滤边缘像素，并保留具有高梯度值的边缘像素，可以通过选择高低阈值来实现。如果边缘像素的梯度值高于高阈值，则将其标记为强边缘像素；如果边缘像素的梯度值小于高阈值并且大于低阈值，则将其标记为弱边缘像素；如果边缘像素的梯度值小于低阈值，则会被抑制。阈值的选择取决于给定输入图像的内容。

双阈值检测的伪代码描写如下：

$$
\begin{aligned}
&if\ \ G_p \geq HighThreshold \\
&\quad G_p\ is\ an\ strong\ edge \\
&else\ \ if\ \ G_p \geq LowThreshold \\
&\quad G_p\ is\ an\ weak\ edge \\
&else \\
&\quad G_p\ \ should\ \ be\ \ \sup pressed
\end{aligned}
$$

6．通过抑制孤立的弱边缘完成边缘检测

到目前为止，被划分为强边缘的像素点已经被确定为边缘，因为它们是从图像中的真实边缘中提取出来的。然而，对于弱边缘像素，将会有一些争论，因为这些像素可以从真实边缘提取也可以是因噪声或颜色变化引起的。为了获得准确的结果，应该抑制由后者引起的弱边缘。通常，由真实边缘引起的弱边缘像素将连接到强边缘像素，而噪声响应未连接。为了跟踪边缘连接，通过查看弱边缘像素及其 8 个邻域像素，只要其中一个为强边缘像素，则该弱边缘点就可以保留为真实的边缘。

抑制孤立边缘点的伪代码描述如下：

$$
\begin{aligned}
&if\ \ G_p == LowThreshold\ \ and\ \ G_p\ \ connected\ to\ a\ strong\ edge\ pixel \\
&\quad G_p\ is\ an\ strong\ edge \\
&else \\
&\quad G_p\ \ should\ \ be\ \ \sup pressed
\end{aligned}
$$

7． 二值化图像输出结果

将结果二值化输出，即将图像上的像素点的灰度值设置为 0 或 255，也就是将整个图像呈现出明显的黑白效果的过程。

4.4 Canny 算子优缺点

Canny 算子在二维空间的检测效果、定位性能及抗噪性能方面都更要优于 LOG 算子。 Canny 算子的不足之处是对于无噪声的图像会使图像边缘变模糊。为使检测效果更好，我们在运用该算子时一般选取稍大的滤波尺度，但是这样做易使图像的某些边缘细节特征丢失掉。

5. 各边缘检测算子对比

算子	优缺点比较
Roberts	对具有陡峭的低噪声的图像处理效果较好，但利用Roberts算子提取边缘的结果是边缘比较粗，因此边缘定位不是很准确。
Sobel	对灰度渐变和噪声较多的图像处理效果比较好，Sobel算子对边缘定位比较准确。
Kirsch	对灰度渐变和噪声较多的图像处理效果较好。
Prewitt	对灰度渐变和噪声较多的图像处理效果较好。
Laplacian	对图像中的阶跃性边缘点定位准确，对噪声非常敏感，丢失一部分边缘的方向信息，造成一些不连续的检测边缘。
LoG	LoG算子经常出现双边缘像素边界，而且该检测方法对噪声比较敏感，所以很少用LoG算子检测边缘，而是用来判断边缘像素是位于图像的明区还是暗区。
Canny	此方法不容易受噪声的干扰，能够检测到真正的弱边缘。在edge函数中，最有效的边缘检测方法是Canny方法。该方法的优点在于使用两种不同的阈值分别检测强边缘和弱边缘，并且仅当弱边缘与强边缘相连时，才将弱边缘包含在输出图像中。因此，这种方法不容易被噪声“填充”，跟容易检测出真正的弱边缘。

第四章 SUSAN 边缘及角点检测方法

1 SUSAN 检测方法概述

SUSAN 检测方法是一种基于窗口模板的检测方法，主要是通过在图像每个像素点位置处建立一个窗口，这个窗口是圆形的，这里为了得到各方向同性的响应，窗口内可以是常数权值或高斯权值，一般情况下，窗口半径为 3.4 个像素（即窗口内总有 37 个像素）。

这样的窗口模板被放置在每个像素的位置，确定点之间强度相似程度可以由下面的图来描述，这里的 x 轴是指像素点之间强度差别，y 轴指的相似程度，为 1 就指完全相似。

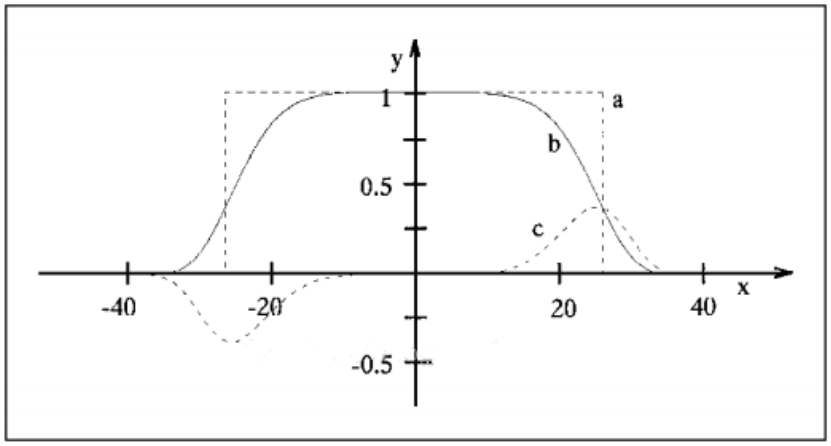


Figure 5. a) The original similarity function (*y* axis, no units) versus pixel brightness difference (*x* axis, in greylevels). For this example the pixel brightness difference "threshold" is set at ±27 greylevels. b) The more stable function now used. c) The boundary detector *B* (see later text).

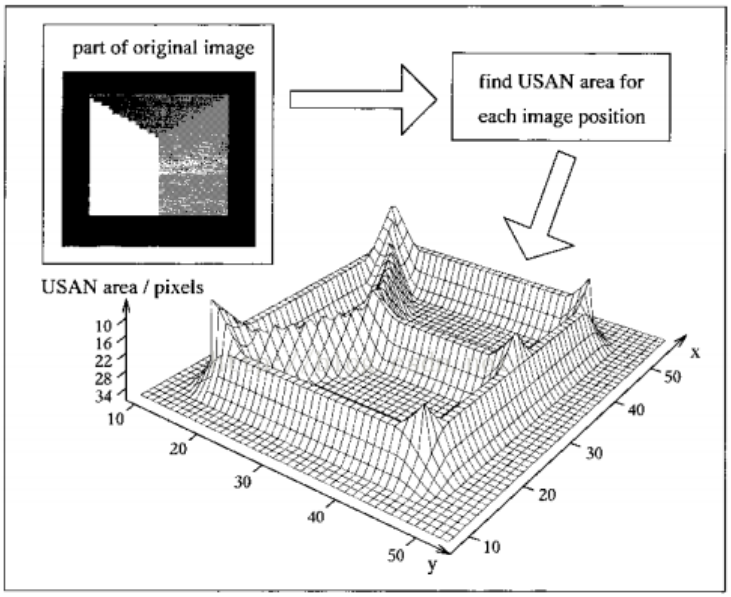


Figure 3. A three dimensional plot of USAN area given a small part of a test image, showing edge and corner enhancement.

考虑为了运算简单，一般情况下使用 a 线（上图），最后统计所以同中心点相似的点数（即相似程度为 1 的点）相似点所在区域被称为 USAN（Univalue Segment Assimilating Nucleus），而特征边缘或角点就是在这个点数值局部最小地方。图中的分界点（区分是否相似的像素差别）实际上反映了图像特征的最小对比程度，及被排除噪声的最大数目。

SUSAN 边缘及角点检测没有用到图像像素的梯度（导数），因为这个原因，所以即使在噪声存在情况下，也能有好的表现。因为只要噪声足够的少，没有包含到所有相似像素的点，那么噪声就可以被排除。在计算时，单个值的集合统一消除了噪声的影响，所以 SUSAN 对于噪声具有较好的鲁棒性。

SUSAN 是通过比较像素点邻域同其中心相似程度，所以也具有了光强变化不变性（因为像素点差别不会变化），及旋转不变性（旋转不会改变局部像素间的相似程度），及一定程度的尺度不变性（角点的角度在尺度放大并不会变化，这里的一定程度是指在尺度放大时，局部的曲度会慢慢平滑）。另外 SUSAN 使用的参数也非常的少，所以对于计算量及储存量要求低。

SUSAN 方法被提出来，主要是为针对于边缘检测和角点检测的，然后其对于噪声较高的鲁棒性，也会用于在噪声消除中，被用于去选择最佳局部平滑邻域（相似程度的点最多的地方）。本文重点描述 SUSAN 方法用于边缘检测和角点检测的思路，并简要介绍下 SUSAN 方法是如何应用于噪声消除领域的。

2 SUSAN 边缘检测

SUSAN 边缘检测总共包括 5 个步骤：边缘响应的计算；边缘方向的计算；非极大值抑制；子像素精度和检测位置。具体的检测过程如下。

1.边缘响应的计算

首先考虑到图像以每点像素为中心的圆形模板（半径为 3.4 个像素，模板内共有 37 个像素），对于其内的每个邻域点都作如下相似度衡量的计算（标准是上图的 a 线），这里的 r 是领域像素距离中心的长度，而 ro 是中心位置，t 指相似度分界值（其决定了特征同背景的最小区别程度，及最大可被排除的噪声数）。

$$c(\vec{r},\vec{r_0}) = \begin{cases} 1 & \text{if } |I(\vec{r}) - I(\vec{r_0})| \le t \\ 0 & \text{if } |I(\vec{r}) - I(\vec{r_0})| > t, \end{cases}$$

当然我们也可以用平滑的线来从代替这种直接的分割方式（如下图 b 线），这样可以获得更稳定而敏感的结果，虽然计算复杂但是可以通过查找表来获得较快的速度。公式如下：

$$c(\vec{r},\vec{r_0}) = e^{-(\frac{I(\vec{r})-I(\vec{r_0})}{t})^6}.$$

并计算总共的相似程度：

$$n(\vec{r_0}) = \sum_{\vec{r}} c(\vec{r},\vec{r_0})$$

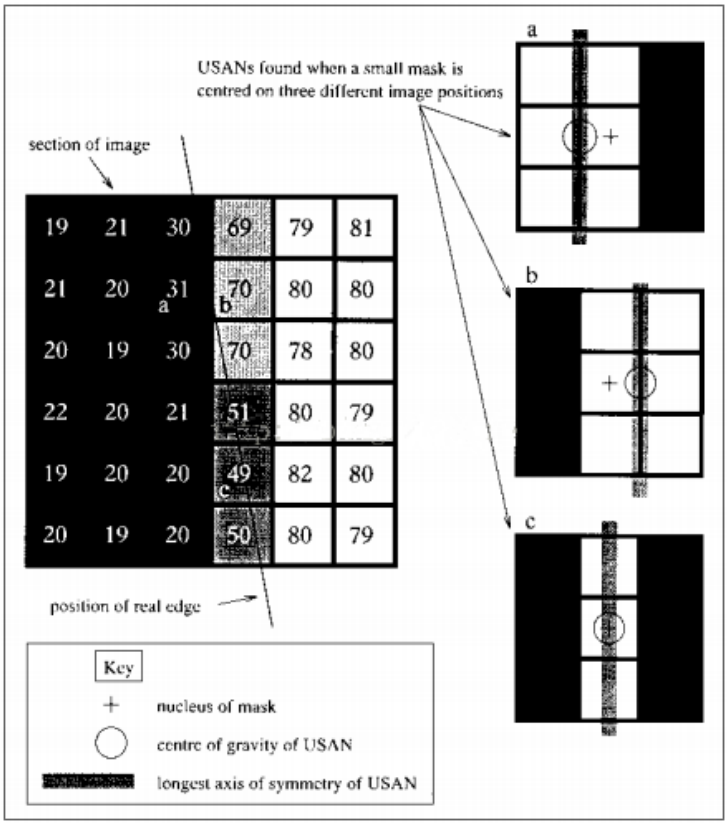
接下来，将 n 同一个固定的阈值 g 比较（一般设置为最大同中心相似点数 nmax 的 0.75 倍左右），初始的边缘响应可以用下面等式计算：

$$R(\vec{r}_0) = \begin{cases} g - n(\vec{r}_0) & \text{if } n(\vec{r}_0) < g \\ 0 & \text{otherwise,} \end{cases}$$

这里 g 是为了排除噪声的影响，而当 n 小于这个阈值时，才能被考虑其的边缘是否是边缘，n 越小，其边缘响应就越大。如果一个阶跃边缘被考虑，如果这个点在边缘上，那么其 USAN 的值应当是小于或等于 0.5 倍的最大值，而对于曲线边缘来说，这个值应该会更少（因为曲度会更小，所以在窗口内，曲线内的区域会更小），所以 g 的加入不会影响原来的边缘检测的结果的（即所得的边缘响应 R 是不会为 0 的）。

2.边缘方向的计算

为什么要计算边缘方向呢？首先，非极大值抑制需要找到边缘方向（这个之后再解释），另外确定子像素级精度也需要找到边缘方向（这个好理解），最后，一些应用可以会用到边缘方向（也包括位置及长度），一个像素若其边缘长度不会 0，那么便有边缘方向，边缘一般有两种情况：



一种是标准的边缘点（如同中的 a 和 b，其刚好位于边缘的一边或另一边），而 c 是另一种情况，其刚好位于左右两边的过渡带间，而过渡带的强度恰好为左右像素强度的一半。这里还引入了一个概念，USAN 的引力中心（USAN 区域即图像模板内的白色区域，就是同中心相似的区域），其可以由下式计算：

$$\vec{\bar{r}}(\vec{r}_0) = \frac{\sum_{\vec{r}} \vec{r}\, c(\vec{r}, \vec{r}_0)}{\sum_{\vec{r}} c(\vec{r}, \vec{r}_0)}$$

其中 a 和 b 情况可以被视一种像素间边缘，引力中心同模板中心的向量恰好垂直于局部边缘方向，我们可以通过这种方式来找到其边缘方向。

而上图 c 的情况就完全不一样了，其引力中心同模板中心恰好在一点上（这被视为一种像素内边缘），此时边缘的方向就要找到 USAN 区域内最长的对称轴方向，这个可以通过如下公式来获得：

$$\overline{(x-x_0)^2}(\vec{r_0}) = \sum_{\vec{r}}(x-x_0)^2\, c(\vec{r},\vec{r_0})$$

$$\overline{(y-y_0)^2}(\vec{r_0}) = \sum_{\vec{r}}(y-y_0)^2\, c(\vec{r},\vec{r_0})$$

$$\overline{(x-x_0)(y-y_0)}(\vec{r_0}) = \sum_{\vec{r}}(x-x_0)(y-y_0)c(\vec{r},\vec{r_0})$$

前两项的比值可以用于确定边缘的方向（度数），而后一项的符号用于确定一个倾斜边缘其梯度的符号（往那边倾斜）。

接下来的问题就是如何自动决定每点是属于以上哪种的边缘，首先如果 USAN 区域（像素数）比模板直径（像素数）要小，那么这应该是像素内边缘（如 c）。如果要大的话，然后又找到 USAN 的引力中心，那么这应该是像素间边缘情况（如 a 和 b）。如果引力中心同模板中心的距离小于 1 个像素的话，此时用于像素内边缘情况（如 c）来计算更为精确。

3.非极大值抑制

CANNY 算子中已有介绍，此处不再赘述。

4.子像素精度

对于每一个边缘点，首先找到边缘方向，然后在边缘垂直方向上减少边缘宽度，之后剩下的边缘点做 3 点二次曲线拟合，而曲线的转折点（一般情况下同最初边缘点的距离应当少于半个像素）被视为边缘的精确位置。

5.检测位置并不依赖于窗口大小

即窗口模板增大，不会改变检测的边缘的位置，即 SUSAN 具有尺度不变性，另外其对于视点改变后的图像检测重复率也非常高，因为视点变化不会影响其角点是否存在。

3 SUSAN 角点检测

角点检测方法同其边缘检测方法非常类似，其使用相同的方法来计算每个像素点的同圆形窗口内的相似程度 n，然后 n 也是同一个权值 g 来比较，不过这里的 g 只是控制噪声存在数目，而需要考虑边缘，所以 g 的值为 nmax 的一半就可以了。

在特征检测方法里，我们一般需要至少一个阈值来区分特征同非特征，而这个阈值的选择将极大影响了最终特征提取的结果。一般情况下，这里有两种类型的阈值：一个衡量质量，一个衡量数量。SUSAN 方法里的 t 和 g 恰好分别符合这两种情况，g 控制角点检测的质量，如果 g 越小，那么我们得到的角点就更尖锐，而 t 则控制角点数目，如果 t 越小，那么我们检测的角点数目就越小，然后 t 却不会影响最终角点质量。（这段是题外话了）

在计算角响应度后，而在非极大值抑制之前，我们可以排除一些在边缘或噪声位置的误检测角点，比如一条倾斜率比较小的直线，因为边缘线至少要一个像素宽度，所以会在直线上形成不连续断裂，这些断裂的位置可能会形成被误检的角点（如上图的 c）。

1.排除误检的角点

首先找到 USAN 的引力中心，然后计算引力中心同模板中心的距离，当 USAN 恰好能指示一个正确角点时，其引力中心同模板中心将不会靠得太近，而如果是条细线的话，引力中心同模板中心将会很近（如上图 a 和 b 所示）。另外的操作就是要加强 USAN 的邻近连续性，在真实图像里，非常容易出现小的噪声点，而这些噪声点可能分布于 USAN 内。所以对于模板内所有的像素，如果其是位于引力中心同模板中心所连的直线上，那么都应该视为 USAN 的一部分。

2.非极大值抑制

SUSAN 角点检测方法同基于导数检测方法一个非常大的优势在于，其不会在靠近中心相邻的区域不会与中心区域的角响应很难区分，所以局部的非极大值抑制中需要简单地选择局部的最大值就可以了。

附：SUSAN 论文下载地址：
http://www.gwylab.com/download/SUSAN_1997.pdf

4 SUSAN 噪声滤波方法

SUSAN 噪声滤波方法主要是通过仅平滑那些同中心像素相似的区域（即 USAN），而由此保留图像的结构信息。而 USAN 的平滑主要就是找到其中所以像素的平均值，而不会对其相邻的不相关区域进行操作。

不过在区分相似度的公式上，却没有用到原来的公式，而是如下：

$$c(\vec{r},\vec{r}_0) = e^{-(\frac{I(\vec{r})-I(\vec{r}_0)}{t})^2}$$

滤波器的最终公式为：

$$J(x,y) = \left(\sum_{i\neq 0, j\neq 0} I(x+i, y+j) * e^{-\frac{r^2}{2\sigma^2} - \frac{(I(x+i,y+j)-I(x,y))^2}{t^2}} \right) \Big/ \sum_{i\neq 0, j\neq 0} e^{-\frac{r^2}{2\sigma^2} - \frac{(I(x+i,y+j)-I(x,y))^2}{t^2}}$$

第五章 *新兴的边缘检测算法

1 小波分析

1986 年，小波分析在 Y.Meyer ， S.Mallat 与 I.Daubechies 等的研究工作基础上迅速地发展起来，成为一门新兴的学科，并在信号处理中应用相当广泛。小波变换的理论基础是傅里叶变换，它具有多尺度特征，如当尺度较大时，可以很好的滤除噪声，但不能很精确的检测到边缘；当尺度较小时，能够检测到很好的边缘信息，但图像的去噪效果较差。因此可以把用多种尺度来检测边缘而得到的结果相结合，充分利用各种尺度的优点，来得到更精确的检测结果。

目前，有很多不同的小波边缘检测算法，主要差别在于采用的小波变换函数不同，常用的小波函数包括： Morlet 小波、 Daubechies 小波、 Harr 小波、Mexican Hat 小波、Hermite 小波、 Mallet 小波、基于高斯函数的小波及基于 B 样条的小波等。

2 模糊算法

在上世纪 80 年代中期， Pal 与 King 等人率先提出了一种模糊算法用来检测图像的边缘，可以很好地把目标物体从背景中提取出来。基于模糊算法的图像边缘检测步骤如下：

(1) 把隶属度函数 G 作用在图像上，得到的映像为“模糊的隶属度矩阵”；

(2) 反复对隶属度矩阵实行非线性变换，从而使真实的边缘更加明显，而伪边缘变弱；

(3) 计算隶属度矩阵的G^{-1}（逆）矩阵；

(4)采用“ max ”和“ min ”算子提取边缘。该算法存在的缺点是计算复杂，且一部分低灰度值的边缘信息有所损失。

3 人工神经网络

利用人工神经网络 [36-38] 来提取边缘在近年来已经发展为一个新的研究方向，其本质是视边缘检测为模式识别问题，以利用它需要输入的知识少及适合并行实现等的优势。其思想与传统方法存在很大的不同，是先把输入图像映射到某种神经元网络，再把原始边缘图这样的先验知识输入进去，然后进行训练，一直到学习过程收敛或者用户满意方可停止。用神经网络方法来构造函数模型主要是依据训练。在神经网络的各种模型中，前馈神经网络应用最为广泛，而最常用来训练前馈网络的是 BP 算法。用 BP 网络检测边缘的算法存在一些缺点，如较慢的收敛速度，较差的数值稳定性，难以调整参数，对于实际应用的需求很难满足。目前， Hopfield 神经网络算法和模糊神经网络算法在很多方面都有所应用。神经网络具有下面 5 种特性：

(1) 自组织性；

(2) 自学习性；

(3) 联想存储功能；

(4) 告诉寻求优化解的能力；

(5) 自适应性等。

以上神经网络的特性决定了它用来检测图像边缘的可用性。

第六章 附录

1.致谢及引用

声明：本资料全部整理自互联网，仅用作学习。现注明转载出处如下：

致谢：

第一章·概述

1. **边缘检测概述**

网址：https://www.cnblogs.com/ronny/p/4001910.html
网址：https://blog.csdn.net/KYJL888/article/details/78253053
网址：https://blog.csdn.net/tigerda/article/details/61192943

2. **用梯度算子实现边缘检测的原理**

网址：
https://wenku.baidu.com/view/858615b250e2524de4187e04.html?from=search

第二章 一阶微分边缘算子

网址：
https://wenku.baidu.com/view/858615b250e2524de4187e04.html?from=search
网址：http://blog.sina.com.cn/s/blog_82a927880102vd9p.html
网址：https://blog.csdn.net/xiaojiegege123456/article/details/7714863
网址：https://blog.csdn.net/swj110119/article/details/51777422

第三章 二阶微分边缘算子

网址：https://blog.csdn.net/u014485485/article/details/78364573
网址：https://blog.csdn.net/gnehcuoz/article/details/52793654
网址：
https://www.gwylab.com/download/基于数学形态学的图像边缘检测方法.pdf
网址：https://wenku.baidu.com/view/0d4bc87d964bcf84b9d57b6e.html
网址：https://blog.csdn.net/jia20003/article/details/41173767
网址：https://www.cnblogs.com/techyan1990/p/7291771.html

第四章 SUSAN 边缘及角点检测方法

网址：https://blog.csdn.net/tostq/article/details/49305615

第五章 新兴的边缘检测方法

网址：
https://www.gwylab.com/download/基于数学形态学的图像边缘检测方法.pdf