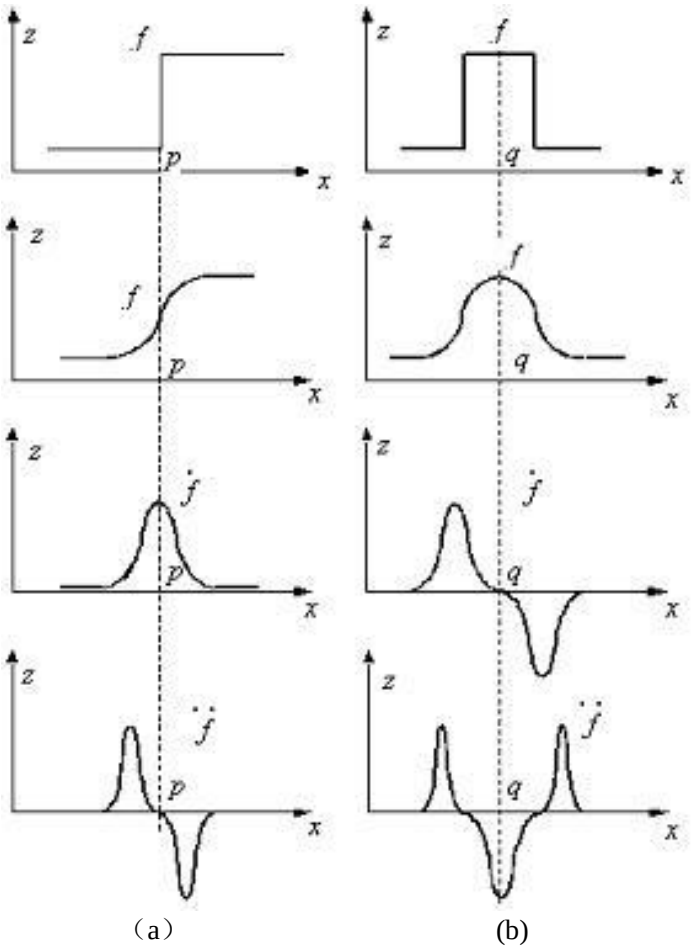
第六章 边缘检测

边缘(edge)是指图像局部强度变化最显著的部分．边缘主要存在于目标与目标、目标与背景、区域与区域(包括不同色彩)之间，是图像分割、纹理特征和形状特征等图像分析的重要基础．图像分析和理解的第一步常常是边缘检测(edge detection)．由于边缘检测十分重要，因此成为机器视觉研究领域最活跃的课题之一．本章主要讨论边缘检测和定位的基本概念，并使用几种常用的边缘检测器来说明边缘检测的基本问题．

图像中的边缘通常与图像强度或图像强度的一阶导数的不连续性有关．图像强度的不连续可分为：(1) 阶跃不连续，即图像强度在不连续处的两边的像素灰度值有着显著的差异；(2) 线条不连续，即图像强度突然从一个值变化到另一个值，保持一个较小的行程后又返回到原来的值．在实际中，阶跃和线条边缘图像是很少见的，由于大多数传感元件具有低频特性，使得阶跃边缘变成斜坡型边缘，线条边缘变成屋顶形边缘，其中的强度变化不是瞬间的，而是跨越一定的距离，这些边缘如图 6．1 所示．



（a） (b)

图 6．1 两种常见的边缘，(a) 阶跃函数，(b) 线条函数．
其中第一排为理想信号，第二排对应实际信号

对一个边缘来说，有可能同时具有阶跃和线条边缘特性．例如在一个表面上，由一个平面变化到法线方向不同的另一个平面就会产生阶跃边缘；如果这一表面具有镜面反射特性且两平面形成的棱角比较圆滑，则当棱角圆滑表面的法线经过镜面反射角时，由于镜面反射分

量，在棱角圆滑表面上会产生明亮光条，这样的边缘看起来象在阶跃边缘上叠加了一个线条边缘．由于边缘可能与场景中物体的重要特征对应，所以它是很重要的图像特征。比如，一个物体的轮廓通常产生阶跃边缘，因为物体的图像强度不同于背景的图像强度．

在讨论边缘算子之前，首先给出一些术语的定义：

边缘点：图像中具有坐标$[i, j]$且处在强度显著变化的位置上的点．

边缘段：对应于边缘点坐标$[i, j]$及其方位θ，边缘的方位可能是梯度角．

边缘检测器：从图像中抽取边缘(边缘点和边缘段)集合的算法．

轮廓：边缘列表，或是一条表示边缘列表的拟合曲线．

边缘连接：从无序边缘表形成有序边缘表的过程．习惯上边缘的表示采用顺时针方向来排序．

边缘跟踪：一个用来确定轮廓的图像（指滤波后的图像）搜索过程．

边缘点的坐标可以是边缘位置像素点的行、列整数标号，也可以在子像素分辨率水平上表示．边缘坐标可以在原始图像坐标系上表示，但大多数情况下是在边缘检测滤波器的输出图像的坐标系上表示，因为滤波过程可能导致图像坐标平移或缩放．边缘段可以用像素点尺寸大小的小线段定义，或用具有方位属性的一个点定义．请注意，在实际中，边缘点和边缘段都被称为边缘．

由边缘检测器生成的边缘集可以分成两个子集：真边缘集和假边缘集．真边缘集对应场景中的边缘，假边缘集不是场景中的边缘．还有一个边缘子集，即场景中漏检的边缘集．假边缘集称之为假阳性（false Positive），而漏掉的边缘集则称之为假阴性（false Negative）．

边缘连接和边缘跟踪之间的区别在于：边缘连接是把边缘检测器产生的无序边缘集作为输入，输出一个有序边缘集；边缘跟踪则是将一幅图像作为输入，输出一个有序边缘集．另外，边缘检测使用局部信息来决定边缘，而边缘跟踪使用整个图像信息来决定一个像素点是不是边缘．

6.1　梯度

边缘检测是检测图像局部显著变化的最基本运算．在一维情况下，阶跃边缘同图像的一阶导数局部峰值有关．梯度是函数变化的一种度量，而一幅图像可以看作是图像强度连续函数的取样点阵列．因此，同一维情况类似，图像灰度值的显著变化可用梯度的离散逼近函数来检测．梯度是一阶导数的二维等效式，定义为向量

$$G(x,y)=\begin{bmatrix}G_x\\G_y\end{bmatrix}=\begin{bmatrix}\frac{\partial f}{\partial x}\\\frac{\partial f}{\partial y}\end{bmatrix} \tag{6．1}$$

有两个重要的性质与梯度有关：(1) 向量$G(x,y)$的方向就是函数$f(x,y)$增大时的最大变化率方向；(2) 梯度的幅值由下式给出：

$$|G(x,y)|=\sqrt{G_x^2+G_y^2} \tag{6．2}$$

在实际应用中，通常用绝对值来近似梯度幅值：

$$|G(x,y)|=|G_x|+|G_y| \tag{6．3}$$

或

$$|G(x,y)|\approx\max(|G_x|,|G_y|) \tag{6．4}$$

由向量分析可知，梯度的方向定义为

$$a(x,y)=\arctan(G_y/G_x) \tag{6．5}$$

其中α角是相对x轴的角度．

注意梯度的幅值实际上与边缘的方向无关，这样的算子称为各向同性算子(isotropic operators)．

对于数字图像，方程 6．1 的导数可用差分来近似．最简单的梯度近似表达式为

$$G_x = f[i, j+1] - f[i, j]$$
$$G_y = f[i, j] - f[i+1, j] \qquad (6.6)$$

请注意 j 对应于 x 轴方向，而 i 对应于负 y 轴方向．这些计算可用下面的简单卷积模板来完成

$$G_x = \boxed{-1 \mid 1} \qquad G_y = \begin{array}{|c|}\hline 1 \\ \hline -1 \\ \hline \end{array} \qquad (6.7)$$

在计算梯度时，计算空间同一位置 x 和 y 处的真实偏导数是至关重要的．然而采用上面公式计算的梯度近似值 G_x 和 G_y 并不位于同一位置，G_x 实际上是内插点 $[i, j+1/2]$ 处的梯度近似值，G_y 是内插点 $[i+1/2, j]$ 处的梯度近似值．由于这个缘故，人们常常使用 2×2 一阶差分模板（而不用 1×2 或 2×1 模板）来求 x 和 y 的偏导数：

$$G_x = \begin{array}{|c|c|}\hline -1 & 1 \\ \hline -1 & 1 \\ \hline \end{array} \qquad G_y = \begin{array}{|c|c|}\hline 1 & 1 \\ \hline -1 & -1 \\ \hline \end{array} \qquad (6.8)$$

用上式计算 x 和 y 方向梯度的位置是相同的，这一点位于内插点 $[i+1/2, j+1/2]$ 处，即在 2×2 邻域的所有四个像素点之间．不过这种计算可能会导致一些混淆，所以，通常用 3×3 邻域计算梯度值．这一方法将在下一节讨论．

6．2 边缘检测算法

边缘检测算法有如下四个步骤：

滤波：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，大多数滤波器在降低噪声的同时也导致了边缘强度的损失，因此，增强边缘和降低噪声之间需要折衷．

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幅值来完成的．

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据是梯度幅值阈值判据．

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．

在边缘检测算法中，前三个步骤用得十分普遍。这是因为大多数场合下，仅仅需要边缘检测器指出边缘出现在图像某一像素点的附近，而没有必要指出边缘的精确位置或方向．边缘检测误差通常是指边缘误分类误差，即把假边缘判别成边缘而保留，而把真边缘判别成假边缘而去掉．边缘估计误差是用概率统计模型来描述边缘的位置和方向误差的．我们将边缘检测误差和边缘估计误差区分开，是因为它们的计算方法完全不同，其误差模型也完全不同．

最近的二十年里发展了许多边缘检测器，这里仅讨论常用的几种边缘检测器．

6．2．1 Roberts 算子

Roberts 交叉算子为梯度幅值计算提供了一种简单的近似方法：

$$G[i, j] = |f[i, j] - f[i+1, j+1]| + |f[i+1, j] - f[i, j+1]| \qquad (6.9)$$

用卷积模板，上式变成：

$$G[i, j] = |G_x| + |G_y| \qquad (6.10)$$

其中G_x和G_y由下面的模板计算：

$$G_x = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array} \qquad G_y = \begin{array}{|c|c|}\hline 0 & -1 \\ \hline 1 & 0 \\ \hline \end{array} \tag{6．11}$$

同前面的2×2梯度算子一样，差分值将在内插点$[i+1/2, j+1/2]$处计算．Roberts 算子是该点连续梯度的近似值，而不是所预期的点$[i, j]$处的近似值．Roberts 边缘检测器的试验结果见图 6.3.

6．2．2 Sobel 算子

正如前面所讲，采用3×3邻域可以避免在像素之间内插点上计算梯度．考虑一下图 6．2 中所示的点$[i, j]$周围点的排列．Sobel 算子也是一种梯度幅值，

$$M = \sqrt{s_x^2 + s_y^2} \tag{6．12}$$

其中的偏导数用下式计算：

$$\begin{aligned} s_x &= (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6) \\ s_y &= (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4) \end{aligned} \tag{6．13}$$

其中常数$c = 2$

和其他的梯度算子一样，s_x和s_y可用卷积模板来实现：

$$s_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad s_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} \tag{6．14}$$

请注意这一算子把重点放在接近于模板中心的像素点．图 6.3 和图 6.4 表明了这一算子的作用．Sobel 算子是边缘检测器中最常用的算子之一．

a_0	a_1	a_2
a_7	$[i, j]$	a_3
a_6	a_5	a_4

图 6．2 用于说明 Sobel 算子和 Prewitt 算子的邻域像素点标记

6．2．3 Prewitt 算子

Prewitt 算子与 Sobel 算子的方程完全一样，只是常量 c=1．所以

$$s_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad s_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \tag{6．15}$$

请注意，与 Sobel 算子不同，这一算子没有把重点放在接近模板中心的像素点．这种边缘检测器的性能也示意在图 6.3 和图 6.4 中．

6．2．4 各种算法比较

现在来比较一下上面讨论过的边缘检测器．我们将按照本节开头所提出的滤波、增强和

检测这三个步骤，来比较各种方法．第四步定位将不讨论．首先给出在忽略滤波步骤情况下Roberts、Sobel 和 Prewitt 边缘检测方法实验结果，如图 6.3 所示．对滤波后的图象进行边缘检测的结果见图 6.4，其中滤波器为前一章介绍的7×7高斯滤波器，梯度幅值的计算见方程 6.3．比较图 6.3 和图 6.4 可以发现，由于噪声影响，一些假边缘也被检测出来了．

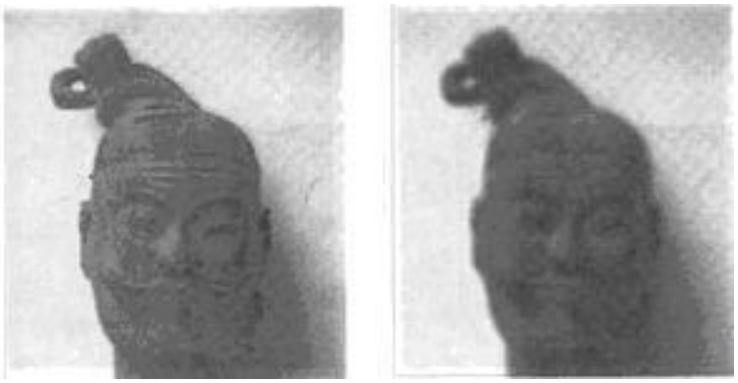


图 6.2 用于边缘检测的测试图像
(a)原始图像　　(b)7x7 高斯滤波的图像

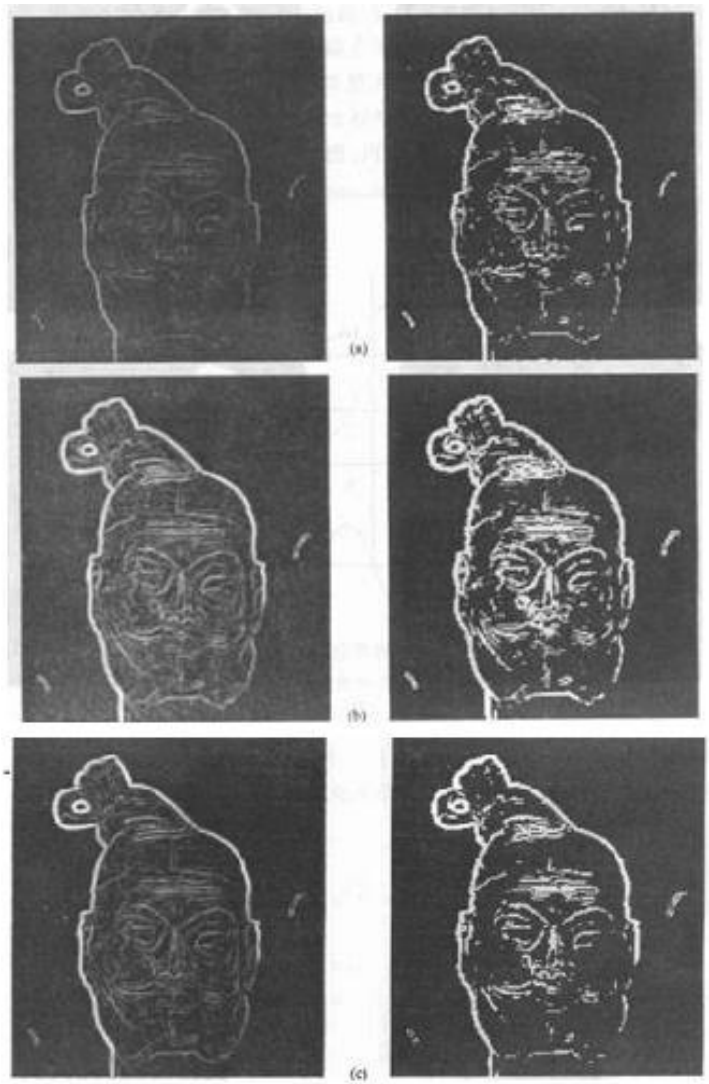


图 6．3　各种边缘检测器对未经滤波的图像（a）进行边缘检测的比较．
(c) Roberts 交叉算子．（d）Sobel 算子．(e) Prewitt 算子．

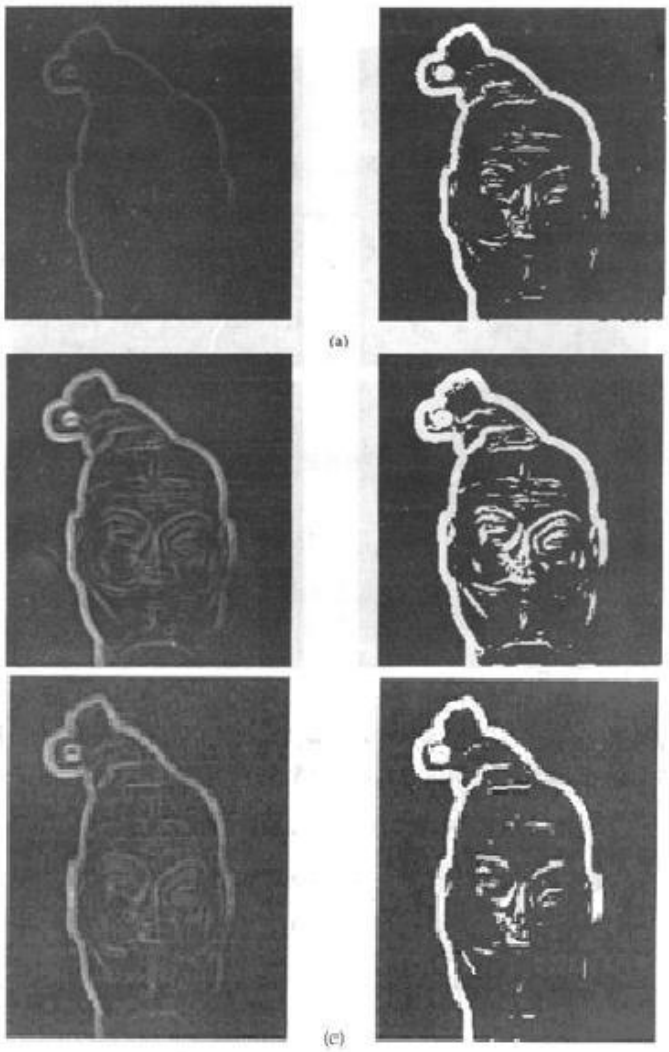


图 6．4　各种边缘检测器对滤波后的图像（图 6.3 (b)）进行边缘检测的结果．
(a) Roberts 交叉算子．（b）Sobel 算子．(c) Prewitt 算子

6．3 二阶微分算子

前面讨论了计算一阶导数的边缘检测器，如果所求的一阶导数高于某一阈值，则确定该点为边缘点．这样做会导致检测的边缘点太多(注意一下图 6．3 和图 6．4 中阈值化后的粗线)．一种更好的方法就是求梯度局部最大值对应的点，并认定它们是边缘点，如图 6．5 所示．在图 6.5 中，若用阈值来进行边缘检测，则在 a 和 b 之间的所有点都被记为边缘点．但通过去除一阶导数中的非局部最大值，可以检测出更精确的边缘．一阶导数的局部最大值对应着二阶导数的零交点这意味着在边缘点处有一阶导数的峰值，同样地，有二阶导数的零交叉点．这样，通过找图像强度的二阶导数的零交叉点就能找到边缘点．

在二维空间，对应二阶导数有两种算子：拉普拉斯算子和二阶方向导数．

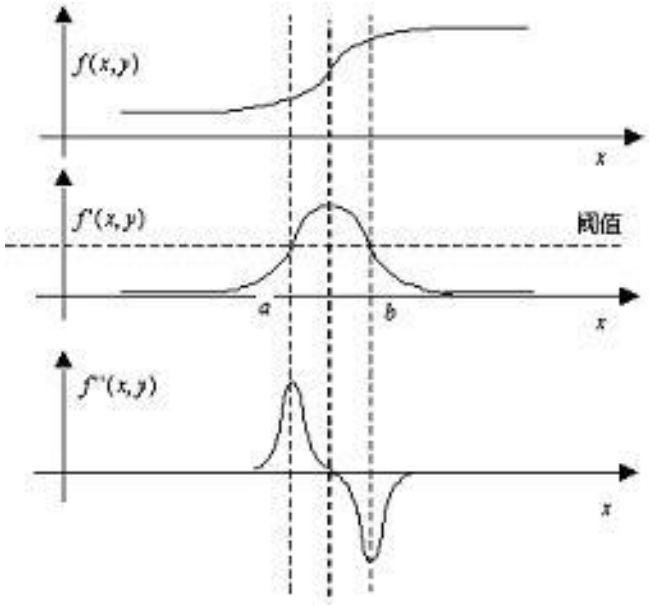


图 6．5 用阈值进行边缘检测和用二阶导数的零交点进行边缘检测示意图.

6．3．1　拉普拉斯算子

平滑过的阶跃边缘二阶导数是一个在边缘点处过零的函数(见图 6．5)．拉普拉斯算子是二阶导数的二维等效式．函数 $f(x,y)$ 的拉普拉斯算子公式为

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{6．16}$$

使用差分方程对 x 和 y 方向上的二阶偏导数近似如下：

$$\begin{aligned} \frac{\partial^2 f}{\partial x^2} &= \frac{\partial G_x}{\partial x} \\ &= \frac{\partial (f[i, j+1] - f[i, j])}{\partial x} \\ &= \frac{\partial f[i, j+1]}{\partial x} - \frac{\partial f[i, j]}{\partial x} \\ &= (f[i, j+2] - 2f[i, j+1]) + f[i, j] \end{aligned} \tag{6．17}$$

这一近似式是以点 $[i, j+1]$ 为中心的．用 $j-1$ 替换 j，得到

$$\frac{\partial^2 f}{\partial x^2} = (f[i, j+1] - 2f[i, j]) + f[i, j-1] \tag{6．18}$$

它是以点 $[i, j]$ 为中心的二阶偏导数的理想近似式，类似地，

$$\frac{\partial^2 f}{\partial y^2} = (f[i+1, j] - 2f[i, j]) + f[i-1, j] \tag{6．19}$$

把这两个式子合并为一个算子，就成为下面能用来近似拉普拉斯算子的模板：

$$\nabla^2 \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{6．20}$$

有时希望邻域中心点具有更大的权值，比如下面的模板就是一种基于这种思想的近似拉普拉斯算子：

$$\nabla^2 \approx \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} \qquad (6.21)$$

当拉普拉斯算子输出出现过零点时就表明有边缘存在，其中忽略无意义的过零点(均匀零区)．原则上，过零点的位置精度可以通过线性内插方法精确到子像素分辨率，不过由于噪声，结果可能不会很精确．

考虑图 6．6 中所给的例子．图中表明了对一幅具有简单阶跃边缘的图像进行拉普拉斯运算的结果．输出图像中的一行是：

0	0	0	6	-6	0	0	0

在本例中，对应于原始图像边缘的零交叉点位于两个中心像素点之间．因此，边缘可以用其左边的像素或右边的像素来标记，但整幅图像的标记必须一致．在多数情况下，零交叉点很少恰好在两像素点中间，因此边缘的实际位置要通过内插值来确定．

2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8

一幅包含垂直阶跃边缘的图像

0	0	0	6	-6	0	0	0
0	0	0	6	-6	0	0	0
0	0	0	6	-6	0	0	0
0	0	0	6	-6	0	0	0

图 6．6 垂直方向的阶跃边缘拉普拉斯响应

现在考虑一下图 6．7 所示的例子．该图给出了拉普拉斯算法对斜坡边缘的响应，其中的一行输出是

0	0	0	3	0	-3	0	0

零交叉点直接对应着图像中的一个像素点．再者，这是一种理想情况，边缘的实际位置仍要通过内插方法来确定．

2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8

一幅包含垂直斜坡边缘的图像

0	0	0	3	0	-3	0	0
0	0	0	3	0	-3	0	0
0	0	0	3	0	-3	0	0
0	0	0	3	0	-3	0	0

图 6．7　垂直方向的斜坡边缘拉普拉斯响应

6．3．2 二阶方向导数

已知图像曲面 $f(x,y)$，其 θ（与 y 轴夹角）方向的方向导数在 (x,y) 点的值为

$$\frac{\partial f}{\partial \theta}=\frac{\partial f(x,y)}{\partial x}\sin\theta+\frac{\partial f(x,y)}{\partial y}\cos\theta \tag{6．22}$$

二阶方向导数为，该算子由下式来实现：

$$\frac{\partial^2 f}{\partial \theta^2}=\frac{\partial^2 f(x,y)}{\partial x^2}\sin^2\theta+2\frac{\partial^2 f(x,y)}{\partial x\partial y}\sin\theta\cos\theta+\frac{\partial^2 f(x,y)}{\partial y^2}\cos^2\theta \tag{6．23}$$

根据式(6.5)，在梯度方向上的二阶导数为

$$\frac{\partial^2 f}{\partial \theta^2}=\frac{\dfrac{\partial^2 f}{\partial x^2}\left(\dfrac{\partial f}{\partial x}\right)^2+\dfrac{\partial^2 f}{\partial x\partial y}\dfrac{\partial f}{\partial x}\dfrac{\partial f}{\partial y}+\dfrac{\partial^2 f}{\partial y^2}\left(\dfrac{\partial f}{\partial y}\right)^2}{\left(\dfrac{\partial f}{\partial x}\right)^2+\left(\dfrac{\partial f}{\partial y}\right)^2} \tag{6．24}$$

拉普拉斯的二阶方向导数算子在机器视觉中并不常用，因为任何包含有二阶导数的算子比只包含有一阶导数的算子更易受噪声的影响. 甚至一阶导数很小的局部峰值也能导致二阶导数过零点. 为了避免噪声的影响，必须采用特别有效的滤波方法. 在下一节，我们将讨论高斯滤波与二阶导数相结合的边缘检测方法.

6．4　LoG 算法

正如上面所提到的，利用图像强度二阶导数的零交叉点来求边缘点的算法对噪声十分敏感，所以，希望在边缘增强前滤除噪声. 为此，Marr 和 Hildreth[146]将高斯滤波和拉普拉斯边缘检测结合在一起，形成 LoG（Laplacian of Gaussian, LoG）算法，也称之为拉普拉斯高斯算法. LoG 边缘检测器的基本特征是：

1. 平滑滤波器是高斯滤波器.
2. 增强步骤采用二阶导数(二维拉普拉斯函数).
3. 边缘检测判据是二阶导数零交叉点并对应一阶导数的较大峰值.
4. 使用线性内插方法在子像素分辨率水平上估计边缘的位置.

这种方法的特点是图像首先与高斯滤波器进行卷积(高斯滤波器在 6．6 节中将详细讨论)，这一步既平滑了图像又降低了噪声，孤立的噪声点和较小的结构组织将被滤除. 由于平滑会导致边缘的延展，因此边缘检测器只考虑那些具有局部梯度最大值的点为边缘点. 这一点可以用二阶导数的零交叉点来实现. 拉普拉斯函数用作二维二阶导数的近似，是因为它是一种无方向算子. 为了避免检测出非显著边缘，应选择一阶导数大于某一阈值的零交叉点作为边缘点.

LoG 算子的输出 $h(x,y)$ 是通过卷积运算得到的：

$$h(x,y)=\nabla^2[g(x,y)*f(x,y)] \tag{6．25}$$

根据卷积求导法有

$$h(x,y)=[\nabla^2 g(x,y)]*f(x,y) \tag{6．26}$$

其中：

$$\nabla^2 g(x,y)=\left(\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right)e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (6.\ 27)$$

称之为墨西哥草帽算子(见图 6．8)．这样，下面两种方法在数学上是等价的：

1． 求图像与高斯滤波器卷积，再求卷积的拉普拉斯变换．

2． 求高斯滤波器的拉普拉斯变换，再求与图像的卷积．

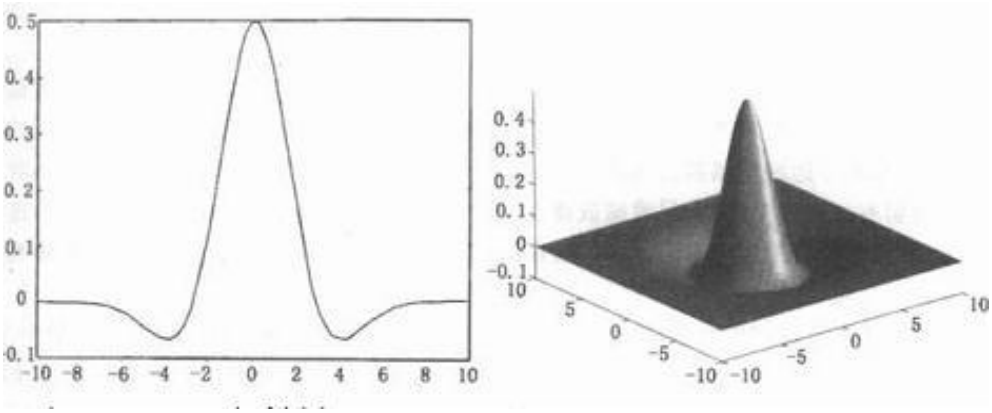


图 6．8 一维和二维高斯函数的拉普拉斯变换图的翻转图，其中 σ=2．

如果采用第一种方法，就可能用到 5．4 介绍的高斯平滑滤波器．直接实现 LoG 算法的典型模板见图 6．9．图 6．10 给出了应用 LoG 算子和零交叉点检测的结果．有关讨论实现 LoG 算法的有效方法，请参见文献[Huertas 1986]．

滤波(通常是平滑)、增强、检测这三个边缘检测步骤对使用 LoG 边缘检测仍然成立，其中平滑是用高斯滤波器来完成的；增强是将边缘转换成零交叉点来实现的；边缘检测则是通过检测零交叉点来进行的．

可以看到，零交叉点的斜率依赖于图像强度在穿过边缘时的变化对比度．剩下的问题是把那些由不同尺度算子检测到的边缘组合起来．在上述方法中，边缘是在特定的分辨下得到的．为了从图像中得到真正的边缘，有必要把那些通过不同尺度算子得到的信息组合起来．

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

图 6．9 5×5 拉普拉斯高斯模板

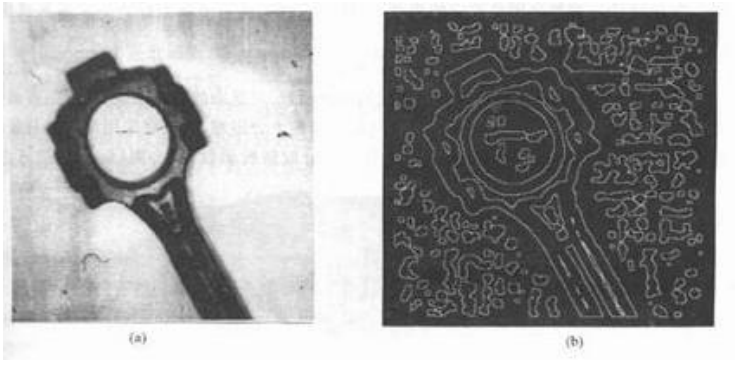


图 6．10 拉普拉斯高斯边缘检测结果．

这里介绍一下尺度空间概念. 高斯平滑运算导致图像中边缘和其它尖锐不连续部分的模糊，其中模糊量取决于σ的值. σ值越大，噪声滤波效果越好，但同时也丢失了重要的边缘信息，影响了边缘检测器的性能. 如果用小尺度的滤波器，又有可能平滑不完全而留有太多的的噪声. 大尺度滤波器在平滑相互邻近的两个边缘时，可能会将它们连在一起，这样只能检测出一个边缘. 因此，在不知道物体尺度和位置的情况下，很难准确确定滤波器的尺度.

使用多尺度滤波模板并在滤波器的不同尺度上分析边缘特性的方法仍在研究中. 这些方法的基本思想是，通过使用大尺度滤波模板产生鲁棒边缘和小尺度滤波模板产生精确定位边缘的特性，来检测出图像的最价边缘.

6．5 图像逼近

一幅图像是一个连续函数的采样阵列．有关图像的大多数思想首先在连续域内进行讨论，然后使用离散逼近法来计算所需要的性质. 如果我们能从采样图像中估计连续函数，那么我们就能从估计的函数中求得图像性质，并且可以在子像素分辨率上计算边缘的位置.

图 6．11 为一幅数字图像对应的离散三维图形表示，其中的纵坐标表示像素灰度值．设连续图像函数为：

$$z = f(x, y) \tag{6．28}$$

现在的任务是从数字图像的灰度值重构连续函数. 对复杂的图像来说，连续强度函数可能包含 x 和 y 的超阶幂方，从而使得重构原始函数十分困难，因此，我们将采用简单的分段函数来建立图像的模型．这样，任务就变成了重构每一个分段解析函数，或小面（Facets）。换言之，就是试图找到在每一像素点的邻域内最能逼近该邻域强度值的简单函数，如图 6. 12 所示．这种逼近叫小面模型（Facet model）[Haralick 1984]．图 6．13 给出了采用5×5邻域的小面模型坐标系统.

连续图像强度函数可以在每一个像素点上得到局部逼近. 对一幅$n \times m$的图像，你能得到$n \cdot m$个逼近函数，每一个函数仅对图像中一个特定像素有效．使用这些函数（而不是像素值）来确定边缘位置.

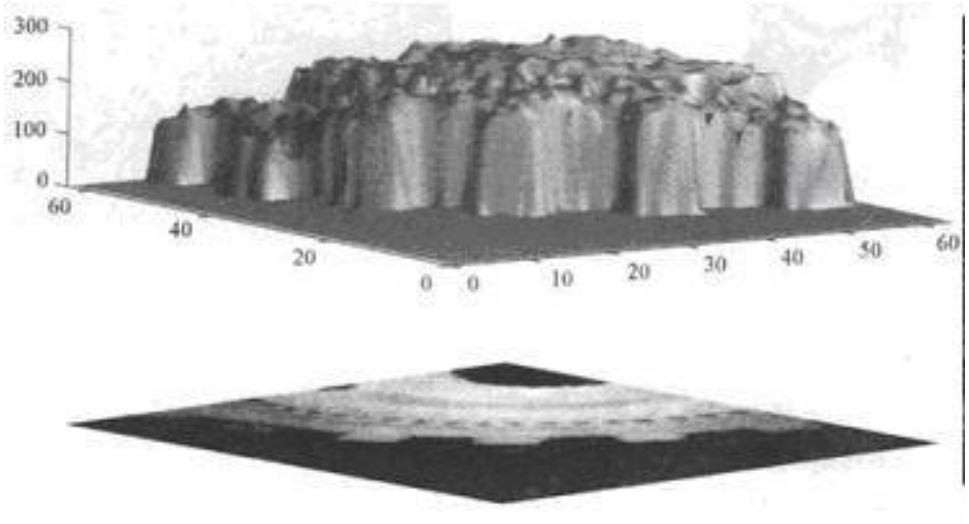


图 6．11 连续图像强度函数的图形表示[Jain 1995]

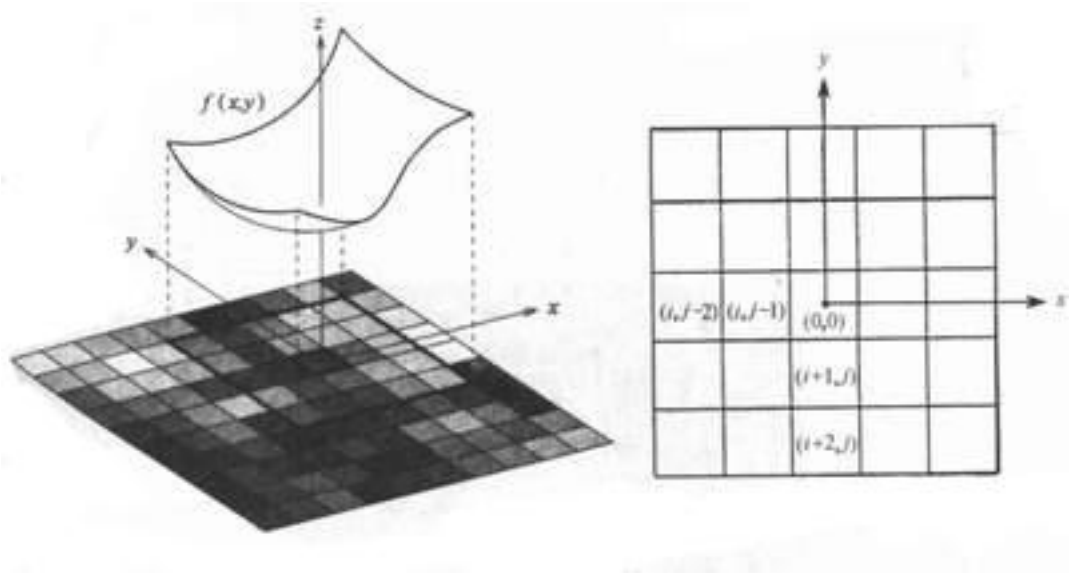


图 6．12 5×5 邻域逼近函数示意图

图 6．13 用 5×5 窗函数的小面模型坐标系统例子．注意像素点 $[i,j]$ 位于邻域的中心．

许多复杂解析函数可用来逼近图像强度．对于简单的图像，分段常量或分段双变量线性函数是强度函数的较好逼近．但对具有更复杂区域的图像，要用到二次、三次甚至更高次幂的函数．本例用下列三次多项式建立图像邻域模型：

$$f(x,y)=k_1+k_2x+k_3y+k_4x^2+k_5xy+k_6y^2+k_7x^3+k_8x^2y+k_9xy^2+k_{10}y^3 \quad (6.29)$$

其中 x 和 y 是相对于要逼近的图像平面中心点 $[0,0]$ 的坐标．

现在的目标是计算方程（6．29）中逼近函数的系数 $k_i(i=1,2,\cdots,10)$．可用最小二乘法通过奇异值分解（Singular-Value Decomposition, SVD）来计算系数 k_i，或者使用 5×5 邻域，用图 6．14 中所示的模板来直接计算三次逼近函数的系数[Haralick 1993].

边缘点出现在像素点邻域逼近函数一阶方向导数局部极值的位置，基于这一事实可以进行边缘检测．一阶导数的局部极值的存在会在一阶导数方向上产生二阶导数的零交叉点．

在方向 θ 上的一阶导数和二阶导数计算公式见(6．22)和(6．23)．

由于图像局部强度是由三次多项式来近似的，角 θ 可以被选作为逼近平面的方位角：

$$\sin\theta=\frac{k_2}{\sqrt{k_2^2+k_3^2}} \qquad \cos\theta=\frac{k_3}{\sqrt{k_2^2+k_3^2}} \quad (6.30)$$

根据式(6．23)在点 (x_0,y_0) 处方向 θ 上的二阶方向导数为

$$\begin{aligned} f''_\theta(x_0,y_0)=&2(3k_7\sin^2\theta+2k_8\sin\theta\cos\theta+k_9\cos^2\theta)x_0\\ &+2(k_8\sin^2\theta+2k_9\sin\theta\cos\theta+3k_{10}\cos^2\theta)y_0\\ &+2(k_4\sin^2\theta+k_5\sin\theta\cos\theta+k_6\cos^2\theta) \end{aligned} \quad (6.31)$$

由于只考虑方向角为 θ 的直线上的点，则 $x_0=\rho\cos\theta$，$y_0=\rho\sin\theta$，将它们代入上式有

$$\begin{aligned} f''_\theta(x_0,y_0)=&6(k_7\sin^3\theta+k_8\sin^2\theta\cos\theta+k_9\sin\theta\cos^2\theta+k_{10}\cos^3\theta)\rho\\ &+2(k_4\sin^2\theta+k_5\sin\theta\cos\theta+k_6\cos^2\theta)\\ =&A\rho+B \end{aligned} \quad (6.32)$$

如果对某一 ρ，有 $|\rho|<\rho_0$，则在图像中点 (x_0,y_0) 处存在边缘点．其中 ρ_0 是像素边长，

$$f_{\theta}^{''}(x_0, y_0; \rho) = 0 \tag{6．33}$$

和

$$f_{\theta}^{'}(x_0, y_0; \rho) \neq 0 \tag{6．34}$$

换言之，如果边缘位置落入像素点的边界之内，则将此边缘点标记为边缘点．若点位于像素点的边界之外，就不能标记为边缘点．用小面模型边缘检测器获得的边缘运算结果见图6．15.

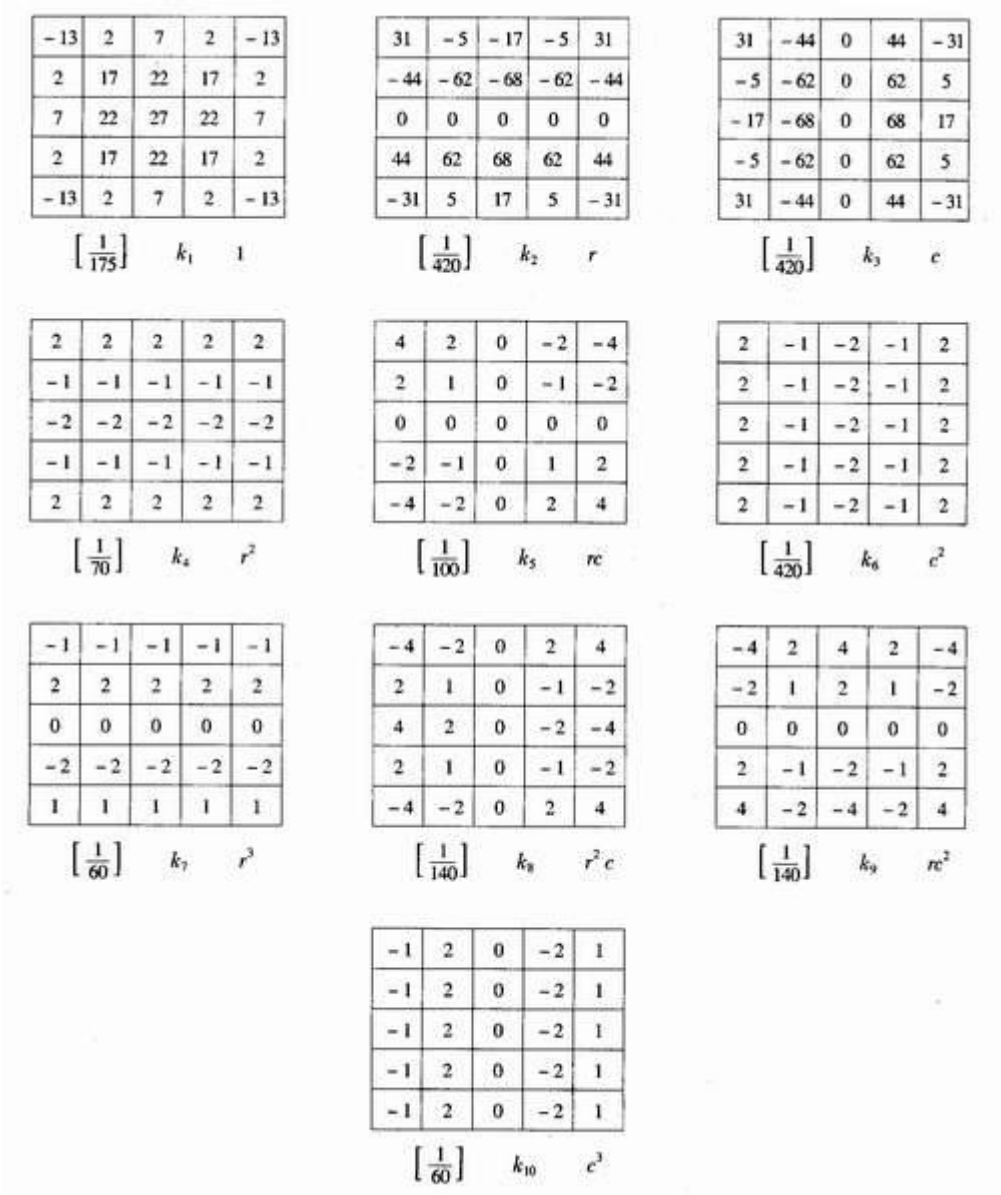
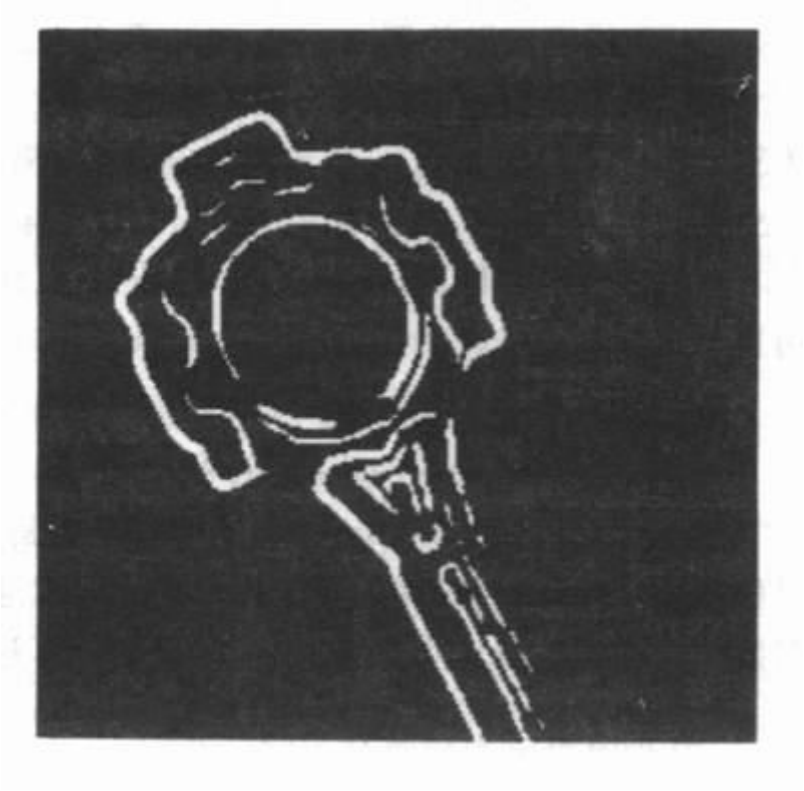


图 6．14　计算三次方逼近函数系数的模板[Haralick 1993].



（a） （b）

图 6．15 用小面模型边缘检测器获得的边缘，其中(a)是原始灰度图像[Jain 1995].

6．6 Canny 边缘检测器

检测阶跃边缘的基本思想是在图像中找出具有局部最大梯度幅值的像素点.检测阶跃边缘的大部分工作集中在寻找能够用于实际图像的梯度数字逼近.由于实际的图像经过了摄像机光学系统和电路系统（带宽限制）固有的低通滤波器的平滑，因此，图像中的阶跃边缘不是十分陡立. 图像也受到摄像机噪声和场景中不希望的细节的干扰. 图像梯度逼近必须满足两个要求：(1) 逼近必须能够抑制噪声效应，(2) 必须尽量精确地确定边缘的位置. 抑制噪声和边缘精确定位是无法同时得到满足的，也就是说，边缘检测算法通过图像平滑算子去除了噪声，但却增加了边缘定位的不确定性；反过来，若提高边缘检测算子对边缘的敏感性，同时也提高了对噪声的敏感性.有一种线性算子可以在抗噪声干扰和精确定位之间提供最佳折衷方案，它就是高斯函数的一阶导数，对应于图像的高斯函数平滑和梯度计算. 梯度的数值逼近可用 6．1 节中列出的 x 和 y 方向上的一阶偏导的有限差分来表示. 高斯平滑和梯度逼近相结合的算子不是旋转对称的. 这种算子在边缘方向上是对称的，在垂直边缘的方向上是反对称的(沿梯度方向). 这也意味着该算子对最急剧变化方向上的边缘特别敏感，但在沿边缘这一方向上是不敏感的，其作用就象一个平滑算子.

Canny 边缘检测器是高斯函数的一阶导数，是对信噪比与定位之乘积的最优化逼近算子[Canny 1986]. 我们将通过下面的符号对 Canny 边缘检测器算法作一概括说明. 用 $I[i, j]$ 表示图像. 使用可分离滤波方法求图像与高斯平滑滤波器卷积，得到的结果是一个已平滑数据阵列

$$S[i, j] = G[i, j; \sigma] * I[i, j]\,, \tag{6．36}$$

其中 σ 是高斯函数的散布参数，它控制着平滑程度.

已平滑数据阵列 $S[i, j]$ 的梯度可以使用 2×2 一阶有限差分近似式(见 6.1 节)来计算 x 与

y 偏导数的两个阵列 $P[i, j]$ 与 $Q[i, j]$：

$$P[i, j] \approx (S[i, j+1] - S[i, j] + S[i+1, j+1] - S[i+1, j]) / 2$$
$$Q[i, j] \approx (S[i, j] - S[i+1, j] + S[i, j+1] - S[i+1, j+1]) / 2 \quad (6.37)$$

在这个 2×2 正方形内求有限差分的均值，以便在图像中的同一点计算 x 和 y 的偏导数梯度．幅值和方位角可用直角坐标到极坐标的坐标转化公式来计算：

$$M[i, j] = \sqrt{P[i, j]^2 + Q[i, j]^2} \quad (6.38)$$

$$\theta[i, j] = \arctan(Q[i, j] / P[i, j]) \quad (6.39)$$

其中，反正切函数包含了两个参量，它表示一个角度，其取值范围是整个圆周范围内．为高效率地计算这些函数，尽量不用浮点运算．梯度的幅度和方向也可以通过查找表由偏导数计算．反正切函数的大多数计算使用的是定点运算，很少的几个计算是基本浮点运算，其中的浮点运算是由整数和定点算术通过软件实现的．Sedgewick[Sedgewick 1988]给出了一种对大多数应用都足够好的梯度角近似算法．

（1）非极大值抑制

幅值图像阵列 $M[i, j]$ 的值越大，其对应的图像梯度值也越大，但这还不足以确定边缘，因为这里仅仅把图像快速变化的问题转化成求幅值阵列 $M[i, j]$ 的局部最大值问题．为确定边缘，必须细化幅值图像中的屋脊带（Ridge），即只保留幅值局部变化最大的点．这一过程叫非极大值抑制（Non-Maxima Suppression，NMS ），它会生成细化的边缘．

非极大值抑制通过抑制梯度线上所有非屋脊峰值的幅值来细化 $M[i, j]$ 中的梯度幅值屋脊．这一算法首先将梯度角 $\theta[i, j]$ 的变化范围减小到圆周的四个扇区之一，如图 6．16 所示，

$$\zeta[i, j] = \mathrm{Sector}(\theta[i, j]) \quad (6.40)$$

四个扇区的标号为 0 到 3，对应着 3×3 邻域内元素的四种可能组合，任何通过邻域中心的点必通过其中一个扇区．梯度线可能方向的圆周分区用度来标记．该算法使用一个 3×3 邻域作用于幅值阵列 $M[i, j]$ 的所有点．在每一点上，邻域的中心像素 $M[i, j]$ 与沿着梯度线的两个元素进行比较，其中梯度线是由邻域的中心点处的扇区值 $\zeta[i, j]$ 给出的．如果在邻域中心点处的幅值 $M[i, j]$ 不比沿梯度线方向上的两个相邻点幅值大，则 $M[i, j]$ 赋值为零．这一过程可以把 $M[i, j]$ 宽屋脊带细化成只有一个像素点宽．在非极大值抑制过程中，保留了屋脊的高度值．

设

$$N[i, j] = \mathrm{NMS}(M[i, j], \zeta[i, j]) \quad (6.41)$$

表示非极大值抑制过程．$N[i, j]$ 中的非零值对应着图像强度阶跃变化处的对比度．尽管在边缘检测的第一步对图像进行了平滑，但非极大值抑制幅值图像 $N[i, j]$ 仍会包含许多由噪声和细纹理引起的假边缘段．实际中，假边缘段的对比度一般是很小的．

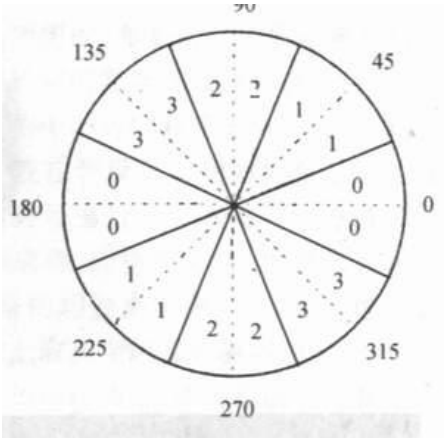


图 6．16 用于非最大值抑制的可能梯度方向划分示意图．

（2）阈值化

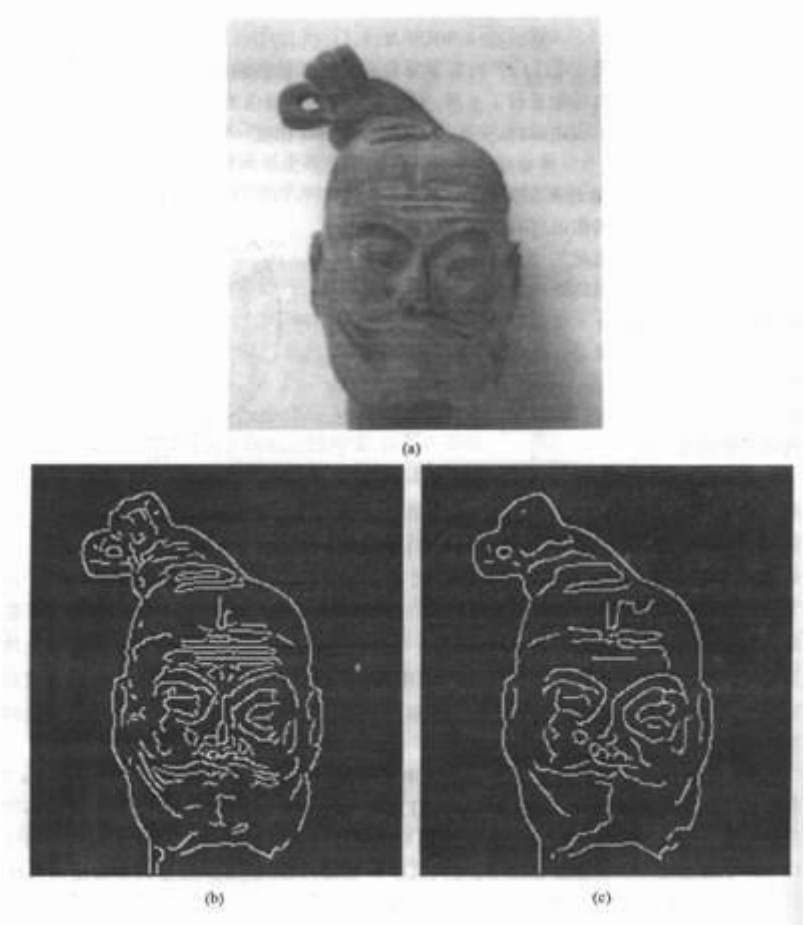
减少假边缘段数量的典型方法是对 $N[i,j]$ 使用一个阈值，将低于阈值的所有值赋零值．对非极大值抑制幅值进行阈值化的结果是一个图像 $I[i,j]$ 的边缘阵列．阈值化后得到的边缘阵列仍然有假边缘存在，原因是阈值τ 太低(假正确)以及阴影的存在，使得边缘对比度减弱，或阈值τ 取得太高而导致部分轮廓丢失(假错误)．选择合适的阈值是困难的，需要经过反复试验．一种更有效的阈值方案是选用两个阈值．

双阈值算法对非极大值抑制图像 $N[i,j]$ 作用双阈值 τ_1 和 τ_2，且 $\tau_2 \approx 2\tau_1$，得到两个阈值边缘图像 $T_1[i,j]$ 和 $T_2[i,j]$．由于图像 $T_2[i,j]$ 是用高阈值得到的，因此它含有很少的假边缘，但 $T_2[i,j]$ 可能在轮廓上有间断(太多的假错误)．双阈值法要在 $T_2[i,j]$ 中把边缘连接成轮廓，当到达轮廓的端点时，该算法就在 $T_1[i,j]$ 的 8-邻点位置寻找可以连接到轮廓上的边缘，这样，算法将不断地在 $T_1[i,j]$ 中收集边缘，直到将 $T_2[i,j]$ 中所有的间隙连接起来为止．这一算法是阈值化的副产物，并解决了阈值选择的一些问题．Canny 边缘检测算法列于算法 6．1 中．

本节所提出的边缘检测算法的试验结果见图 6．17．在计算梯度之前，首先用 7×7 高斯滤波器平滑图像（图 6.17(b)）和 31×31 高斯滤波器平滑图像（图 6.17(d)）．对较小尺度滤波器，非极大值抑制梯度幅值算法可以在边缘处检测出极其细腻的细节，但噪声和细纹理会导致过量的不希望的边缘段．对大尺度滤波器，只产生很少数量的不希望的边缘段，但丢失了边缘的大部分细节．这表明边缘定位和抗噪声之间需要一种折衷．

算法 6．1　Canny 边缘检测

1. 用高斯滤波器平滑图像．
2. 用一阶偏导的有限差分来计算梯度的幅值和方向．
3. 对梯度幅值应用非极大值抑制 ．
4. 用双阈值算法检测和连接边缘．



(d) (e)

图 6．17 高斯平滑滤波器作用于测试图像的边缘检测结果．(a)原始图像,（b） 经过 7×7 高斯滤波器平滑、梯度逼近和非极大值抑制后的灰度图像． (c) 一幅用黑点表示图(b)中大于零的点．(d) 经过 31×31 高斯滤波器平滑、梯度逼近和非极大值抑制后的灰度图像. (e)一 幅用黑点表示图(b)中大于零的点．

6．7 子像素级位置估计

在许多应用中，边缘位置的估计精度高于像素点间距（子像素分辨率）是很有必要的. 用于梯度边缘检测算法和用于二阶边缘检测算法的子像素分辨率精度实现方法是完全不同的，应该分开考虑．

首先，考虑一下二阶边缘检测器的输出，比如，用 LoG 边缘检测器．边缘由像素点之间的零交叉点给出．原则上，边缘位置可通过线性内插方法达到子像素级精度．在实际中，二阶边缘检测输出的噪声太大，以至于无法使用简单的内插方法得到精确的结果，即使是高斯预平滑函数，也无济于事．

使用基于梯度方法检测边缘之后，在边缘位置上获取子像素分辨率精度是很实际的，也是有效的. 把高斯平滑滤波器和一阶导数作用于一个理想的阶跃边缘得到的结果与高斯滤波器平滑结果形状完全一样．如果阶跃边缘不是理想的，而是由一个平面渐变到另一个平面，那么高斯平滑和一阶导数的结果可以用一个较宽的高斯滤波器逼近．

考虑一组服从正态分布的测量值．正态分布曲线的中心对应着正态分布的均值，因此可以通过求这组测量值的均值来估计正态分布中心.现在假设可得到的信息是这些测量值直方图而不是这些测量值本身.正态分布的均值估计可通过直方图每一个方条中心位置乘以该方条所包含的像素点数，再除以直方图面积求得．同样的方法，边缘位置子像素级分辨率精度估计可通过求高斯边缘检测器输出图像的均值来实现．为计算边缘位置到子像素分辨率水平，在边缘的任意一个边沿梯度方向上取高斯边缘检测器输出（不进行非极大值抑制）的幅度，直到梯度幅度低于某一阈值．用梯度幅度 g_i 作为权值来计算沿梯度方向的位置加权值．沿梯度方向的边缘位置子像素级校正用下式给出：

$$\delta d = \frac{\sum_{i=1}^{n} g_i d_i}{\sum_{i=1}^{n} g_i} \qquad (6．42)$$

其中，d_i 是一个像素点沿梯度方向与检测到的边缘点的距离，g_i 是梯度幅度．

一种更简单的算法是通过计算高斯边缘检测器的梯度幅度一阶矩来实现边缘位置 $(\delta x,\delta y)$ 精度到子像素级分辨率水平，其中 $(\delta x,\delta y)$ 是相对于已经检测到的边缘像素．把这一校正值 $(\delta x,\delta y)$ 加到原像素点坐标上就能得到更精确的边缘位置．

沿梯度幅度图方向计算均值的算法，虽然复杂些，但具有以下优点：可用统计方法实现该梯度图与理想梯度图的比较，而比较的结果还能作为边缘检测判据．如果梯度幅度图不是十分接近高斯函数曲线形状，则边缘就不对应理想的阶跃边缘模型．在这种情况下，就不可能用本节提出的方法精确估计边缘位置．

6．8 边缘检测器性能

Abdou 和 Patt[Abdou 1979]、DeMicheli 等人[DeMicheli 1989]已经给出了评价边缘检测器性能的测度公式．在评价边缘检测器性能时应该考虑的判据包括：

1. 假边缘概率，
2. 丢失边缘概率，
3. 边缘角估计误差，
4. 边缘估计值到真边缘的距离平方均值，
5. 扭曲边缘和其它诸如角点和结点的误差范围．

前两条判据涉及边缘检测器算法的性能．中间两条判据涉及边缘定位、方位估计算法的性能．最后一条判据关心的是边缘算法偏离理想模型的误差范围．

6．8．1 性能评价方法

边缘检测器的性能评价可分为两个阶段：计算假边缘与丢失边缘的数目；测量用于估计位置和方位的误差(或误差分布)．

为了测试性能，可以选择一幅合成图像，并已知真实的边缘位于一个轮廓上，该轮廓可由简单曲线的数学公式来建模，比如一个实心矩形，它的轮廓边缘可用线段建模，或两个实心矩形，它们的间隙是已知的．通过把边缘检测器的结果与原始(合成)图像比较，得出真、假、丢失边缘的数目，这件事做起来要比想象的难多了．边缘检测的结果随阈值、滤波器尺度、边缘相互影响和其它因素的变化而变化．如果将一个边缘检测器作用在一幅没有加性噪声、也不平滑且边缘之间互不相连的测试图像上，那么就可以得到一个完善的边缘集(没有假边缘和丢失边缘)．这种集合可以作为用于比较的标准集．

现在考虑一组边缘，该组边缘是从具有加性噪声图像或其它导致产生假边缘的扭曲图像中获取的．在欧基里德距离判据的基础上，将图像中的边缘和标准集中的边缘一对一地进行匹配．离标准集中边缘太远的边缘是假边缘，与标准集中某一边缘非常接近的边缘是真边缘．经过这一步骤，标准集中没有与之相匹配的边缘是在试验条件下丢失的边缘．

测试边缘检测器的上述步骤仅仅根据边缘存在与否来评价其性能，没有说明边缘定位和方位估计的精度．标准集中的正确边缘(前面所计算的)与原始试验图像中边缘的位置和方位进行比较，需要测试模型是可得到的．对实心矩形来说，线段模型构成了它的各个边．边缘位置和方位必须用场景轮廓的数学模型来比较．边缘位置 (x,y) 能与沿轮廓的任意点相对应，但与轮廓上最近的点才是对应点．边缘点与对应点间的距离是可以计算的．对一个线段，用 6．4 节中的公式．从错误定位直方图中估计错误分布，或列表计算样本方差(其中的 n 为边缘点数)来估算偏差．通过比较边缘段最近点的方位和场景轮廓标准曲线方位来计算方位误差．

6．8．2 品质因数

判断边缘检测器性能的方法是先看边缘图像，再评价其性能． 但这并不能给出性能的客观指标．为定量地评价不同边缘检测器的性能，应该建立在可控条件下判断相关性的指标．下面看一下边缘检测器响应中的三种主要误差：

1. 丢失有效的边缘，
2. 边缘定位误差，
3. 将噪声判断为边缘．

边缘检测器的品质因数（Figure of Merit, FM）应该考虑上述三个误差．下面是一种品质因数公式，称为 Pratt 品质因数[Pratt 1991]：

$$FM = \frac{1}{\max(I_A, I_I)} \sum_{i=1}^{I_A} \frac{1}{1+\alpha d_i^2} \tag{6．43}$$

其中，I_A，I_I，d 和α 分别是检测到的边缘、理想边缘、实际边缘与理想边缘间的距离和用于惩罚错位边缘的设计常数．

注意，由于这种品质因数包括了丢失边缘点、边缘点位置和错误边缘点，因此它只能用于有限的几类图像．我们可以构造一些在已知位置上的对比度得到控制的物体，然后使用上述品质因数公式．通过引入随机噪声来评价合成图像的性能是一种非常实用方法．信噪比随品质因数的变化曲线表示检测器性能的退化程度．

6．9 线条检测

本章仅仅讨论了如何检测图像中的阶跃边缘．线条也是图像中非常重要的特征．一条线段可以用非常靠近但极性相反的两个边缘来建模，其中两个边缘之间的距离比平滑滤波器的宽度要窄．

边缘检测滤波器是用来响应阶跃边缘的，对线条不能产生任何有意义的响应．因此，必须使用单独的算法来检测线条，而且该算法对阶跃边缘不产生任何有意义的响应．如何将边缘检测算法和线条检测算法组合在一个系统里是机器视觉领域中的一个突出问题．

线条可以用 Canny 修正算法来检测，即在平滑图像上而不是在梯度幅值图像上进行非极大值抑制．线条是阶跃边缘的一阶导数，因此，在 Canny 算法中，不必进行一阶导数计算．

思考题

5.1 什么是边缘？它与物体的边界有什么关系？与区域的边界有何关系？
5.2 怎样给图像中的边缘建模？哪一种是边缘检测中最常用的模型？为什么？
5.3 写出边缘检测的所有步骤．在进行边缘检测时，是否可以省略一步或几步？为什么？

5.4 为什么拉普拉斯算法不是一类好的边缘检测算法？

5.5 描述一下高斯拉普拉斯边缘检测器．考虑一下边缘检测的不同步骤，指出拉普拉斯算法为什么不是一类好的算法，而高斯拉普拉斯算法却是比较好的算法？

5.6 在LoG算法中你怎样选择合适的尺度？在选择时应考虑些什么因素？能否在算法中使用自动选择尺度技术，为什么？

5.7 图像的小平面模型是什么？怎样用它来进行边缘检测？对区域能用这种模式吗？

5.8 怎样实现边缘在子像素分辨率水平上定位？考虑一下梯度、拉普拉斯、小平面模型下的子象素分辨率水平下的边缘定位估计．

5.9 假定一幅图像用n×n高斯滤波器平滑，在平滑时，矩形滤波窗在图像上移过．在窗函数左上角位置[i,j]处的点用平滑值取代．平滑之后，梯度幅值用6．1节中的逼近式计算．当2×2算子移过平滑图像时，在窗左上角位置[i,j]处的点用梯度幅值取代．边缘检测之后，每一边缘点的位置$[x_{ij}, y_{ij}]$计算到子像素分辨率水平．在原始(未平滑)图像坐标系统中，边缘位置在何处？

计算机练习题

5.1 应用Roberts、Sobel和 Prewitt算子测试不同的图像．(1)在预期的位置得到了边缘吗？(2)人工确定几条边缘段并做上记号，比较一下程序中给出的边缘段位置与人工标出的位置，改变检测阶跃的阈值并观察结果．

5.2 将canny边缘检测器作用于上题所用的图象，比较Canny算子与Sobel算子的性能．