

基元检测

图像中的基元泛指图像中有比较明显特点的基本单元。

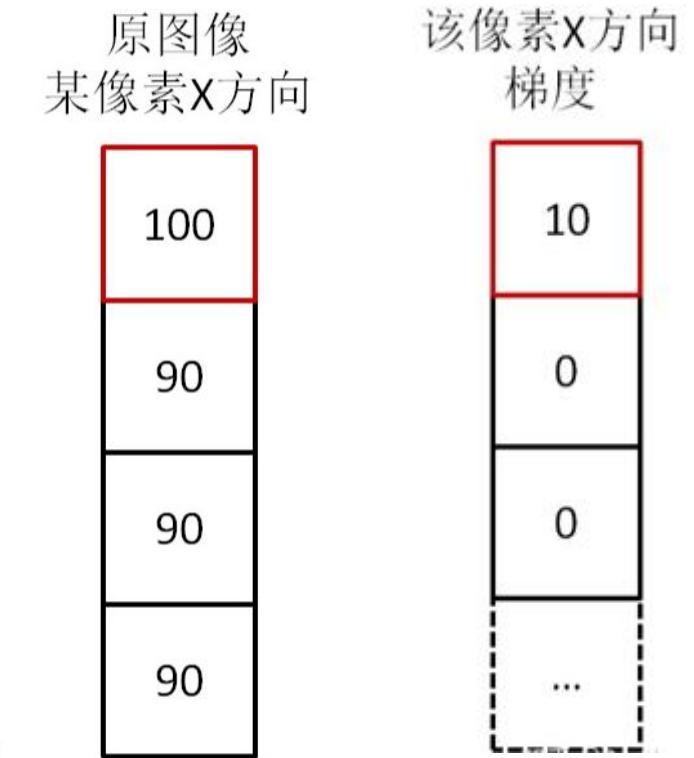
主要基元 (Primitives) : **边缘、角点、直线段、圆、孔、椭圆**以及其他**兴趣点**等 (也包括它们的一些结合体) 。

这些基本单元也常被称为**特征**，所以基元检测也可称特征检测，这些Primitives属于 **Low-level Feature**。

图像梯度：

1. 如果把灰度值看成二元函数值，那么灰度值的变化可以用二元函数的“导数”（或称梯度）来描述。
2. 由于图像是离散数据，导数可以用差分值来表示，即**灰度差**（两个像素的差值）。
3. **图像梯度：图像灰度的变化率。** 把图像看成二维离散函数，图像梯度即这个二维离散函数的求导。

例如，100与90之间相差的灰度值为10，即当前像素点在X轴方向上的梯度为10，而其它点均为90，则梯度全为0，因此可以发现在数字图像处理中，因其像素性质的特殊性，**微积分**在图像处理表现的形式为**计算当前像素点沿偏微分方向的差值**，所以实际的应用是不需要用到求导的，只需进行简单的加减运算。



图像函数 $f(x, y)$ 梯度表达式：

$$\nabla f(x, y) = [G_x, G_y]^T = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]^T$$

幅度：

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

方向角：

$$\theta(x, y) = \arctan \left| \frac{\frac{\partial f}{\partial x}}{\frac{\partial f}{\partial y}} \right|$$

对于数字图像来说，相当于是二维离散函数求梯度，使用差分来近似导数：

$$\begin{aligned} G_x(x, y) &= H(x + 1, y) - H(x - 1, y) \\ G_y(x, y) &= H(x, y + 1) - H(x, y - 1) \end{aligned}$$

获得两个方向的偏导分量后，将它们结合起来构成一个梯度矢量 ∇f

设沿X方向和沿Y方向的偏导分量分别为 G_x 和 G_y ，则梯度矢量为：

$$\nabla f = [G_x, G_y]$$

实际应用中，常使用这个梯度算子输出**梯度值**（矢量的幅度/模），即像素点 (x, y) 处的梯度，可以采用2范数、1范数以及无穷范数来计算：

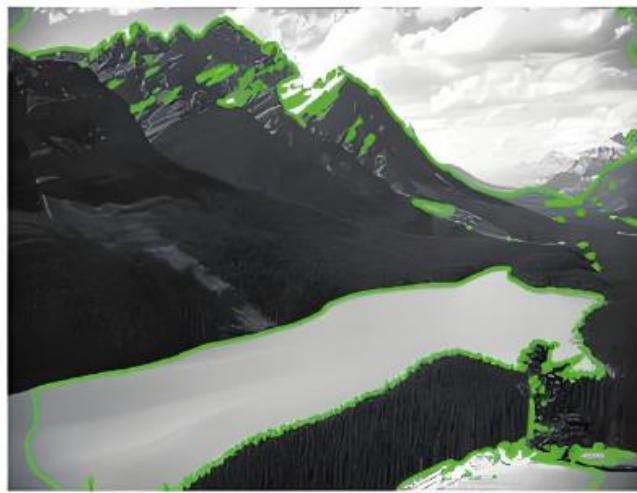
$$|\nabla f_{(2)}| = mag(\nabla f) = (G_x^2 + G_y^2)^{1/2}$$

$$|\nabla f_{(1)}| = |G_x| + |G_y|$$

$$|\nabla f_{(\infty)}| = \max\{|G_x|, |G_y|\}$$

1 边缘检测

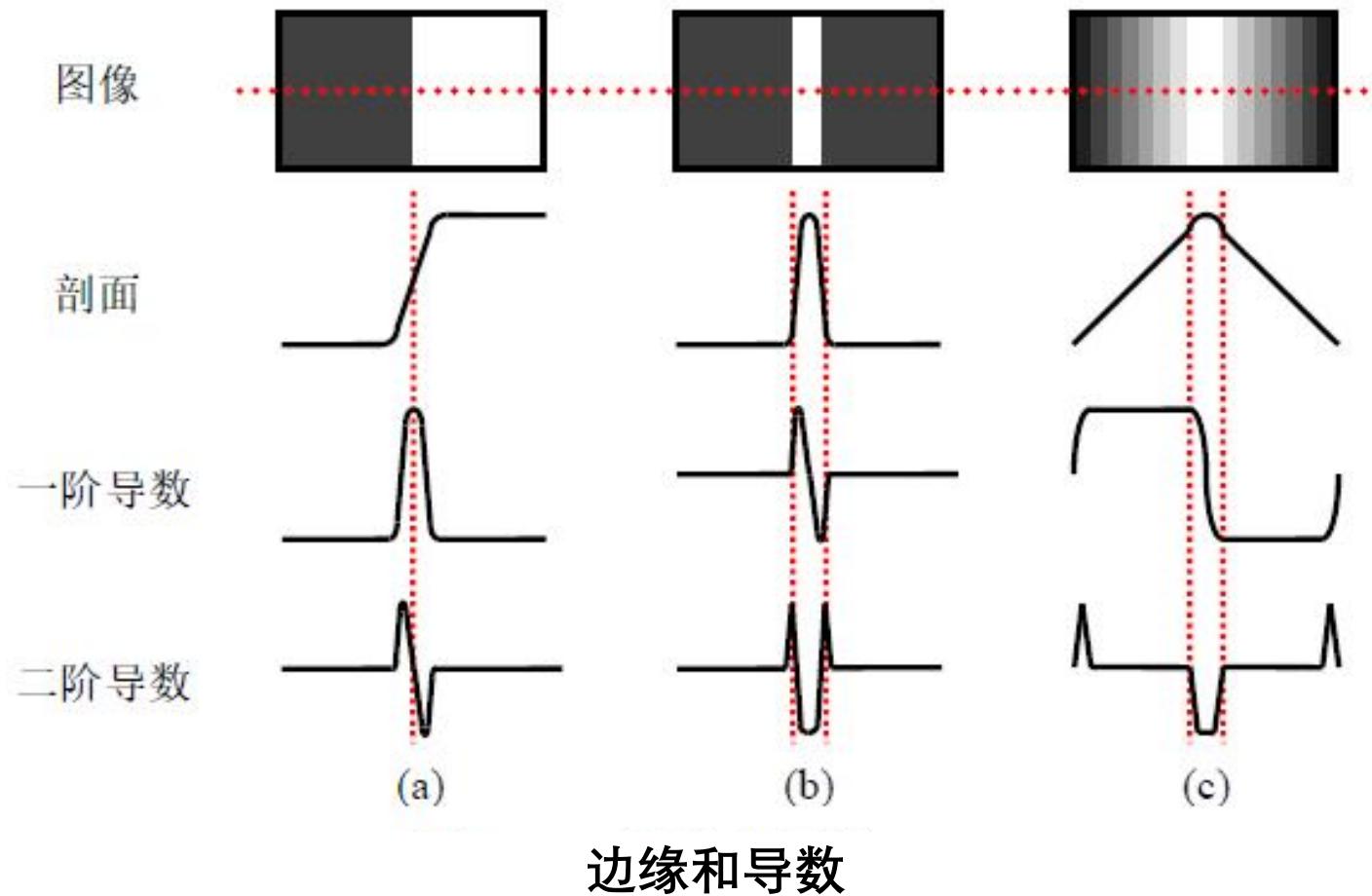
术语定义：



- 1. 边缘：**灰度或结构等信息的突变处，边缘是一个区域的结束，也是另一个区域的开始，利用该特征可以分割图像。
- 2. 边缘点：**图像中具有坐标 $[x, y]$ ，且处在强度显著变化的位置上的点。
- 3. 边缘段：**对应于边缘点坐标 $[x, y]$ 及其方位，边缘的方位可能梯度角。

1 边缘检测

常见的边缘有：① 阶梯状边缘，② 脉冲状边缘，③ 屋顶状边缘



1 边缘检测

一阶导数算子

一阶微分算子给出梯度，所以也称**梯度算子**

它分别计算沿X和Y方向的两个偏导分量

索贝尔算子 (**Sobel**)

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

普利维特算子

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

罗伯茨交叉算子

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

几种常用梯度算子的模板

Sobel算子

Sobel算子：一阶导数的边缘检测算子，即一阶离散性差分算子，用来运算图像亮度函数的灰度值近似值，通过 3×3 模板作为核与图像中的每个像素点做卷积和运算，然后选取合适的阈值以提取边缘。

Sobel卷积因子为：

-1	0	+1
-2	0	+2
-1	0	+1

G_x

+1	+2	+1
0	0	0
-1	-2	-1

G_y

Sobel算子

-1		1
-2		2
-1		1

1	2	1
-1	-2	-1

该算子包含两组3x3的矩阵， 分别为横向及纵向， 将之与图像作平面卷积， 可分别得出横向及纵向的亮度差分近似值。如果以A代表原始图像， G_x 和 G_y 分别代表经横向及纵向边缘检测的图像灰度值， 其公式如下：

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A, \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

$$\begin{aligned} G_x &= (-1) * f(x-1, y-1) + 0 * f(x, y-1) + 1 * f(x+1, y-1) \\ &\quad + (-2) * f(x-1, y) + 0 * f(x, y) + 2 * f(x+1, y) \\ &\quad + (-1) * f(x-1, y+1) + 0 * f(x, y+1) + 1 * f(x+1, y+1) \\ &= [f(x+1, y-1) + 2 * f(x+1, y) + f(x+1, y+1)] \\ &\quad - [f(x-1, y-1) + 2 * f(x-1, y) + f(x-1, y+1)] \end{aligned}$$

其中， $f(a, b)$ 表示图像(a, b)点的灰度值。

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A, \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

$$\begin{aligned}
 G_y &= 1 * f(x-1, y-1) + 2 * f(x, y-1) + 1 * f(x+1, y-1) \\
 &\quad + (-2) * f(x-1, y) + 0 * f(x, y) + 2 * f(x+1, y) \\
 &\quad + (-1) * f(x-1, y+1) + 0 * f(x, y+1) + 1 * f(x+1, y+1) \\
 &= [f(x+1, y-1) + 2 * f(x+1, y) + f(x+1, y+1)] \\
 &\quad - [f(x-1, y-1) + 2 * f(x-1, y) + f(x-1, y+1)]
 \end{aligned}$$

$$\begin{aligned}
 G_y &= 1 * f(x-1, y-1) + 2 * f(x, y-1) + 1 * f(x+1, y-1) \\
 &\quad + 0 * f(x-1, y) + 0 * f(x, y) + 0 * f(x+1, y) \\
 &\quad + (-1) * f(x-1, y+1) + (-2) * f(x, y+1) + (-1) * f(x+1, y+1) \\
 &= [f(x-1, y-1) + 2 * f(x, y-1) + f(x+1, y-1)] \\
 &\quad - [f(x-1, y+1) + 2 * f(x, y+1) + f(x+1, y+1)]
 \end{aligned}$$

其中, $f(a, b)$ 表示图像 (a, b) 点的灰度值。

图像的每一个像素的横向及纵向灰度值通过以下公式计算该点灰度的大小：

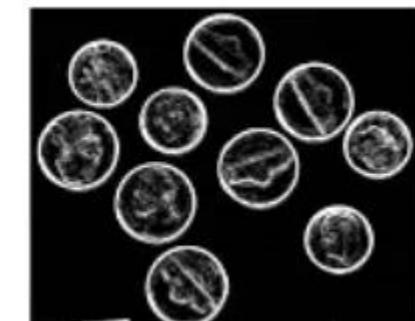
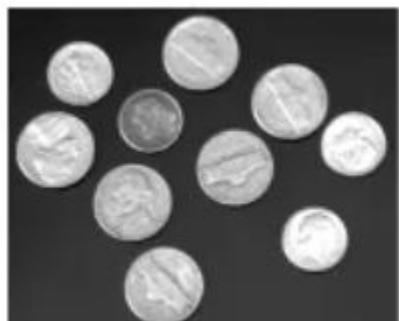
$$G = \sqrt{G_x^2 + G_y^2}$$

通常，为了提高效率，使用不开平方的近似值：

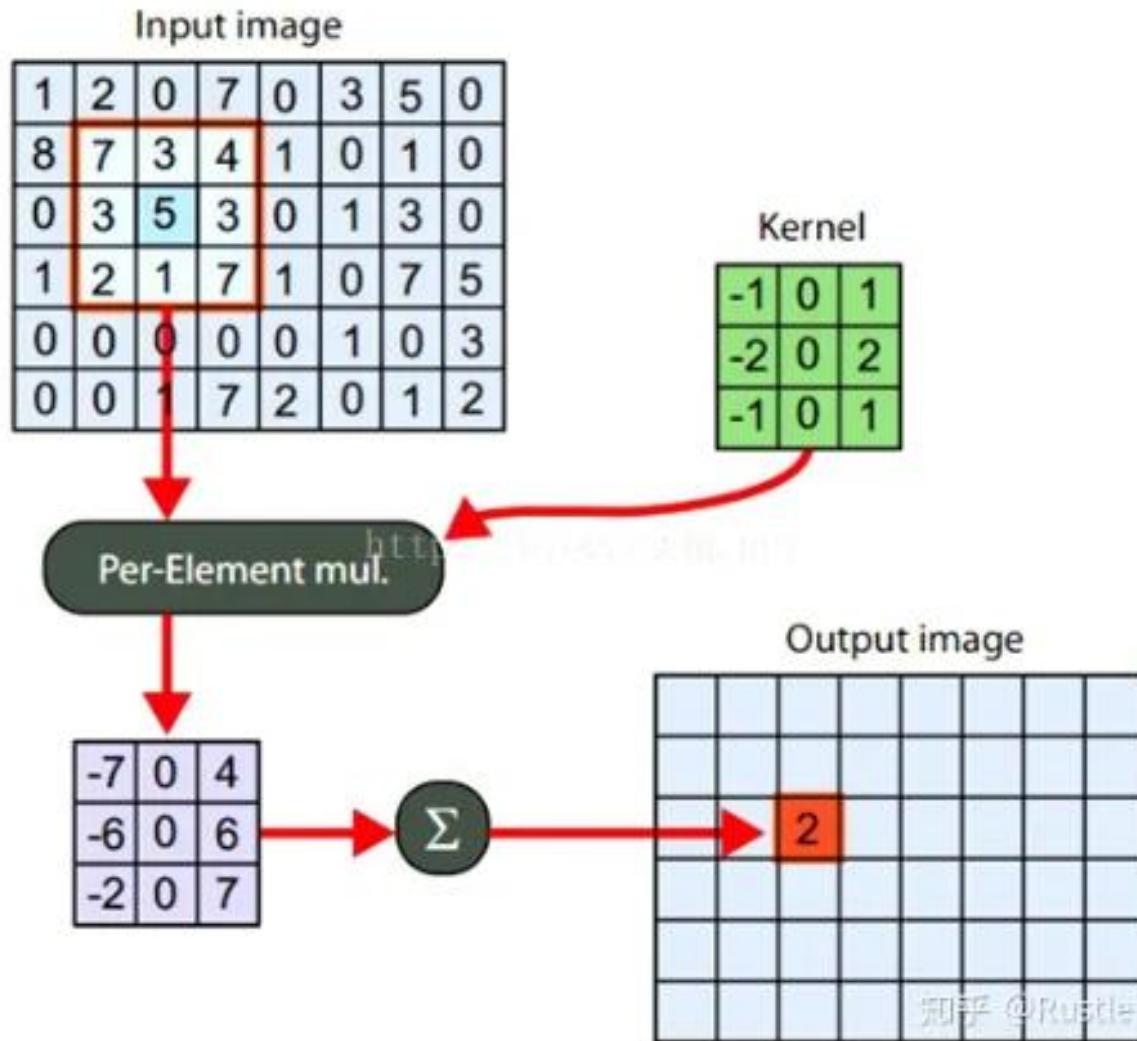
$$|G| = |G_x| + |G_y|$$

如果梯度G大于某一阀值，则认为该点(x,y)为边缘点。

计算梯度方向： $\theta = \arctan\left(\frac{G_y}{G_x}\right)$



原图像、沿X轴梯度图像、沿Y轴梯度图像、梯度图像的可视化



Sobel算子根据像素点上下、左右邻点灰度加权差，在边缘处达到极值这一现象检测边缘。对噪声具有平滑作用，提供较为精确的边缘方向信息，但**边缘定位精度不够高**。

优点：**计算简单，速度快**。但是由于只采用了2个方向的模板，只能检测水平和垂直方向的边缘，因此这种算法对于纹理较为复杂的图像，其边缘检测效果不是很理想。该算法认为：凡灰度新值大于或等于阈值的像素点都是边缘点。这种判断欠合理，会造成边缘点的误判，因为许多**噪声点**的灰度值也很大。

- Sobel算子检测方法对灰度渐变和噪声较多的图像处理效果较好，Sobel算子对边缘定位不是很准确，图像的边缘不止一个像素；当对精度要求不是很高时，是一种较为常用的边缘检测方法。对于细小边缘检测要求较高的，可以采用Scharr算子，它是在Sobel算子的基础上进行改进，在不影响检测速度的前提下，提高了边缘检测的精度。
- Sobel算子和Scharr算子的卷积核大小一样，这意味着计算量一样。

sobel算子

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

scharr算子

-3	0	3
-10	0	10
-3	0	3

-3	-10	-3
0	0	0
3	10	3

普利维特算子(Prewitt operator)

-1	0	+1
-1	0	+1
-1	0	+1

Gx

+1	+1	+1
0	0	0
-1	-1	-1

Gy

-1		1
-2		2
-1		1

1	2	1
-1	-2	-1

Sobel算子

Prewitt算子利用像素点上下、左右邻点灰度差，在边缘处达到极值检测边缘。
对噪声具有平滑作用，定位精度不够高。

罗伯茨交叉算子 (Roberts Cross operator)

灰度公式为：

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

近似公式为：

$$|G| = |G_x| + |G_y|$$

具体计算如下：

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

灰度方向 计算公式为： $\theta = \arctan\left(\frac{G_y}{G_x}\right) + (1/4)\pi$

Roberts算子采用对角线方向相邻两像素之差近似梯度幅值检测边缘。检测水平和垂直边缘的效果好于斜向边缘，定位精度高，对噪声敏感。

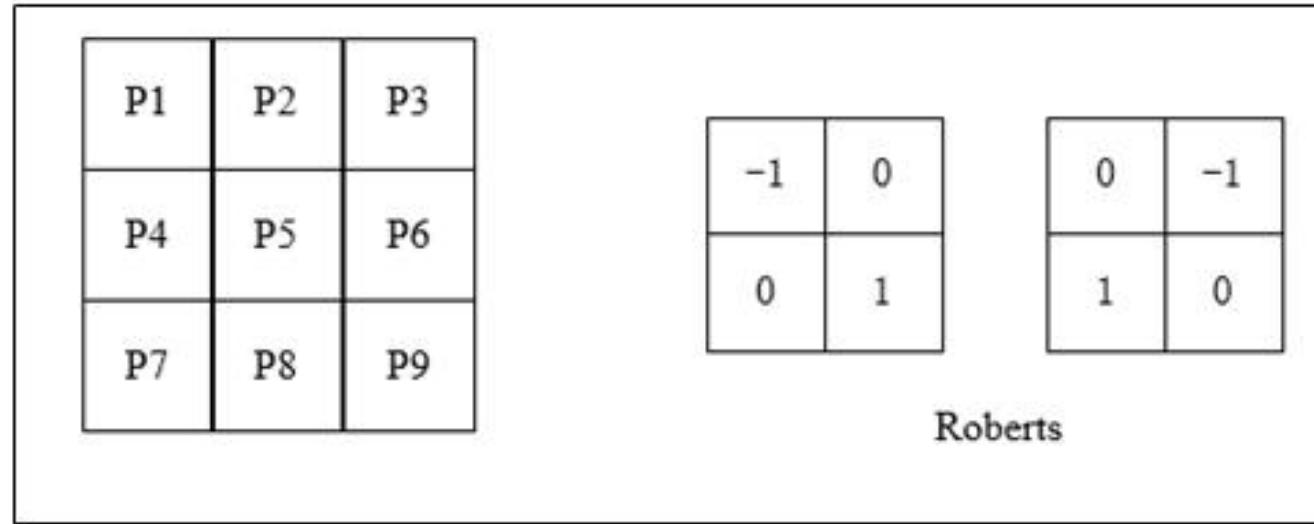
+1	0
0	-1

G_x

0	+1
-1	0

G_y

下面给出Roberts算子的模板，在像素点P₅处 x 和 y 方向上的梯度大小为



$$g_x = \frac{\partial f}{\partial x} = P9 - P5$$

$$g_y = \frac{\partial f}{\partial y} = P8 - P6$$

1 边缘检测

坎尼算子Canny: (1986 年 John F. Canny 提出) 把边缘检测问题转换为检测单位函数**极大值**的问题来考虑。

Canny提出了最优边缘检测的三个主要评价标准:

- 1.低错误率:** 标识出尽可能多的实际边缘，同时尽可能的减少噪声产生的误报。
- 2.高定位性:** 标识出的边缘要与图像中的实际边缘尽可能接近。
- 3.最小响应:** 图像中的边缘只能标识一次，并且可能存在的图像噪声不应标识为边缘。

1 边缘检测

坎尼算子Canny的实现步骤：

- 1、在空间进行高斯滤波：**平滑图像，滤除噪声，滤波器模板尺寸可随尺度不同而改变。
- 2、使用一阶微分检测滤波图像中灰度梯度的大小和方向：**可使用Sobel算子检测边缘

$$G = \sqrt{G_x^2 + G_y^2}, \quad \theta = \arctan\left(\frac{G_y}{G_x}\right)$$

- 3、非极大值抑制 (Non-Maximum Suppression) :**排除非边缘像素，仅仅保留一些细线条 (候选边缘)

- 4、滞后阈值化/双阈值 (Double-Threshold) :**选取两个阈值 (高阈值和低阈值) 并借助滞后阈值化方法最后确定边缘点。

- (1) 若某像素的梯度值超过高阈值，该像素被保留为边缘像素，将其标记为强边缘像素。
- (2) 若某像素的梯度值小于低阈值，该像素被抑制/排除。
- (3) 若某像素的梯度值在两个阈值之间，标记为虚边缘 (需要保留)，若其与强边缘连接，则将该边缘处理为边缘，若与强边缘无连接，则该边缘为弱边缘，将其抑制。

应用高斯滤波去除图像噪声

23	158	140	115	131	87	131
238	0	67	16	247	14	220
199	197	25	106	156	159	173
94	149	40	107	5	71	171
210	163	198	226	223	156	159
107	222	37	68	193	157	110
255	42	72	250	41	75	184
77	150	17	248	197	147	150
218	235	106	128	65	197	202

O

$\times \frac{1}{56}$

1	1	2	1	1
1	3	4	3	1
2	4	8	4	2
1	3	4	3	1
1	1	2	1	1

$\times \frac{1}{56}$

=

T

D

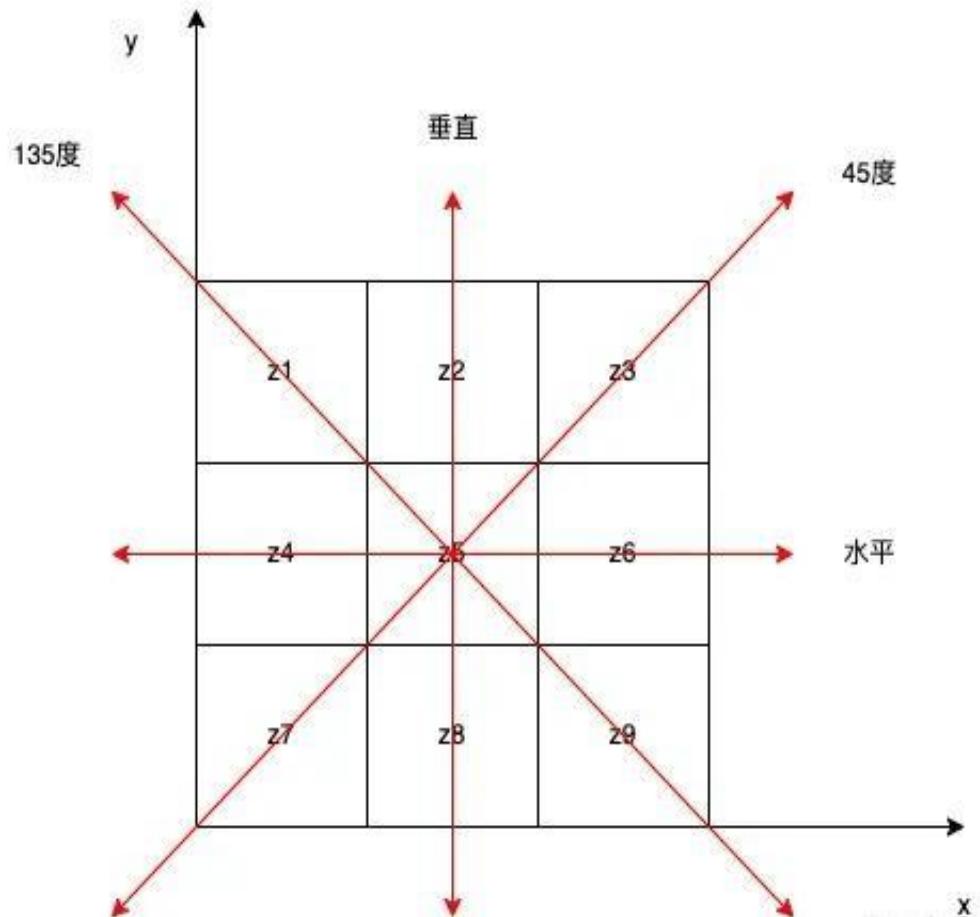
O为原图，T为高斯滤波核，D是滤波后的结果图

对于高斯滤波器 T，越临近中心的点，权值越大。高斯核的大小对于边缘检测的效果具有很重要的作用。

- 优点：高斯核尺寸越大，边缘信息对于噪声的敏感度就越低。
- 缺点：高斯核尺寸越大，边缘检测的定位错误也会随之增加。
- 一般地，一个 5×5 的高斯核可以满足大多数的情况。

非极大值抑制 (NMS)

- 在 Canny 边缘检测算法中，非极大值抑制 (Non-Maximum Suppression, NMS) 是指在一个邻域内，对于一个像素点，如果其梯度值小于其邻域内同方向梯度的最大值，则该像素点不是边缘点。在 8 邻域内，非极大值抑制就只在 0 度、90 度、45 度、135 度四个梯度方向上进行，每个像素点梯度方向按照相近程度用这四个方向来代替。



非极大值抑制 (NMS)

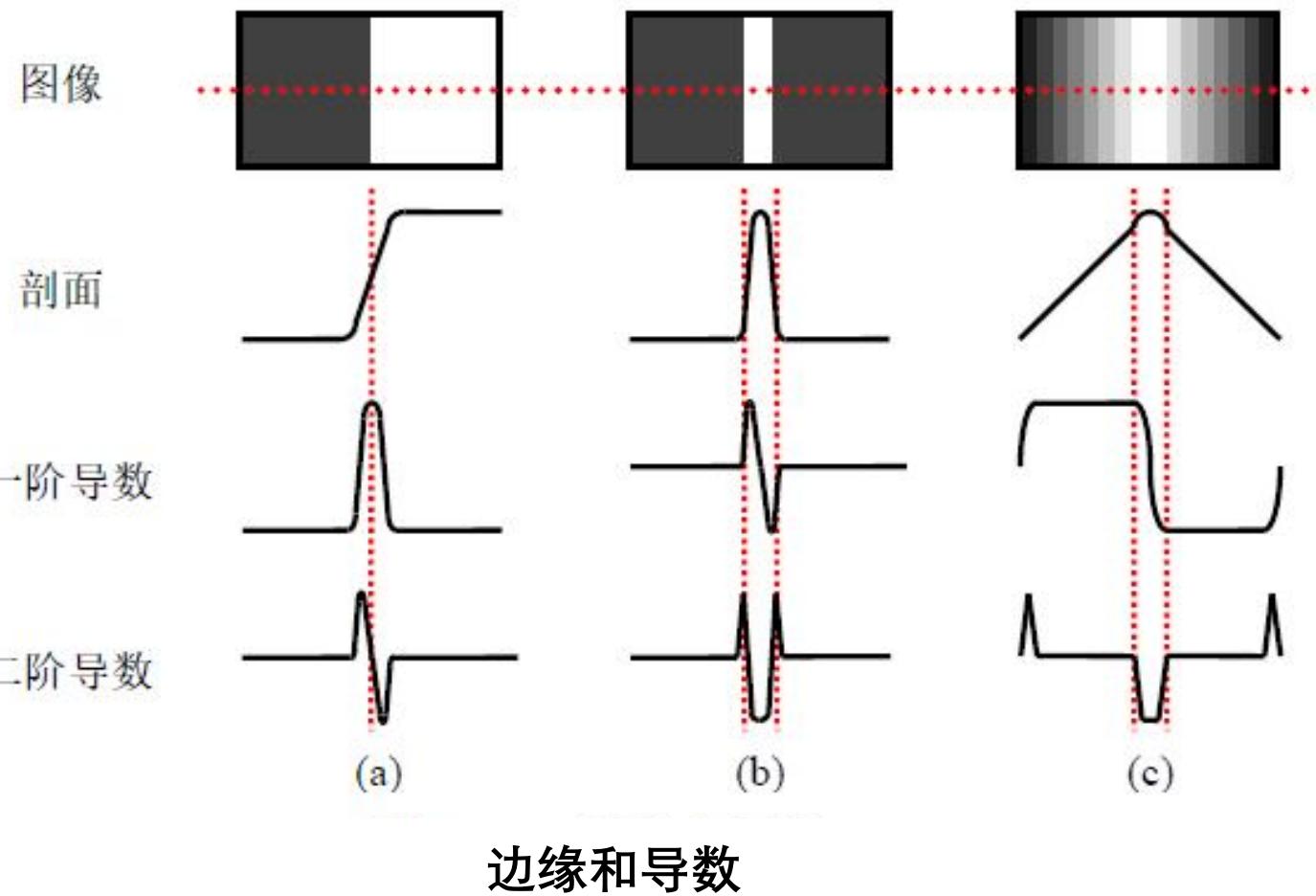
在目标检测任务中，一个目标可能会被多个边界框检测到，这些边界框可能会有不同的位置和大小，但表示同一个目标。**非极大值抑制 (NMS)** 是一种常用的方法，**用于抑制这些重叠的边界框，只保留置信度最高的那个边界框**，从而得到最终的目标检测结果。

- 1) 首先，对所有的边界框按照其**置信度**进行排序，置信度最高的边界框排在最前面。
- 2) 从置信度最高的边界框开始，依次遍历其余边界框。
- 3) 对于当前遍历到的边界框，如果它与前面已经保留的边界框的重叠程度（通过计算 IOU值）大于一定阈值（如0.5），那么就将其抑制掉，不保留。
- 4) 继续遍历下一个边界框，重复上述过程，直到所有的边界框都被处理完毕。

1 边缘检测

二阶导数算子

常见的边缘有：① 阶梯状边缘，② 脉冲状边缘，③ 屋顶状边缘



1 边缘检测

二阶导数算子 (Laplacian)

利用二阶导数的过零点可以确定边缘的位置，所以二阶导数算子也可用于检测边缘。用二阶导数算子检测阶梯状边缘需要将算子模板与图像卷积，并确定算子输出值的过零点。

拉普拉斯算子

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

模板基本要求：对应中心像素的系数是正的，而对应中心像素邻近像素的系数是负的，且所有系数的总和应该是零。

0	-1	0
-1	4	-1
0	-1	0

(a)

-1	-1	-1
-1	8	-1
-1	-1	-1

(b)

拉普拉斯算子四邻域模板和八邻域模板

通过Laplacian算子的模板可以发现：

- 1) 当邻域内像素灰度**相同时**，模板的卷积运算结果为**0**；
- 2) 当中心像素灰度**高于**邻域内其他像素的平均灰度时，模板的卷积运算结果为**正数**；
- 3) 当中心像素的灰度**低于**邻域内其他像素的平均灰度时，模板的卷积为**负数**。

对卷积运算的结果**用适当的衰弱因子处理并加在原中心像素上**，就可以实现图像的**锐化**处理。

1 边缘检测

二阶导数算子

LOG算子：是在拉普拉斯算子的基础上实现的边缘检测算子。

1980年，Marr提出将Laplace算子与高斯低通滤波相结合，提出了高斯拉普拉斯算子LOG (Laplacian of 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。LOG 算子是对阶跃边缘用二阶导数过零点来检测的较好的算子。

数学公式

LOG算子

1. 高斯函数

高斯函数 $g(x, y; \sigma)$ 可以表示为：

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

其中， σ 是标准差，控制着高斯核的：

2. 拉普拉斯算子

二维空间中的拉普拉斯算子可以表示：

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

3. LoG算子

将高斯函数 $g(x, y; \sigma)$ 和拉普拉斯算子结合起来，得到LoG算子：

$$\nabla^2 g(x, y; \sigma) = \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right) (x, y; \sigma)$$

计算 $\nabla^2 g(x, y; \sigma)$ 的具体形式如下：

$$\nabla^2 g(x, y; \sigma) = \frac{1}{\sigma^2} \left(1 - \frac{x^2+y^2}{2\sigma^2} \right) \cdot g(x, y; \sigma)$$

简化后可得：

$$\nabla^2 g(x, y; \sigma) = \left(1 - \frac{x^2+y^2}{2\sigma^2} \right) \cdot \frac{1}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

1. 二维高斯函数: $G(x, y) = \frac{1}{2\pi\delta^2} e^{-\frac{x^2+y^2}{2\delta^2}}$

高斯函数的一阶导数和二阶导数，在很多算子中都会用到。例如，一阶导数应用在 Canny 算子，二阶导数应用在 LoG 算子。

2. 推导其一阶导数:

$$\begin{aligned} \frac{\partial G}{\partial x} &= \frac{1}{2\pi\delta^2} \frac{\partial e^{-\frac{x^2+y^2}{2\delta^2}}}{\partial x} \\ &= \frac{1}{2\pi\delta^2} e^{-\frac{x^2+y^2}{2\delta^2}} * \left(-\frac{2x}{2\delta^2}\right) \\ &= \left(-\frac{1}{2\pi\delta^4}\right) x e^{-\frac{x^2+y^2}{2\delta^2}} \end{aligned}$$

同理: $\frac{\partial G}{\partial y} = \left(-\frac{1}{2\pi\delta^4}\right) y e^{-\frac{x^2+y^2}{2\delta^2}}$

3. 推导其二阶导数:

$$\begin{aligned} \frac{\partial^2 G}{\partial x^2} &= \frac{\partial \left(-\frac{1}{2\pi\delta^4}\right) x e^{-\frac{x^2+y^2}{2\delta^2}}}{\partial x} = \left(-\frac{1}{2\pi\delta^4}\right) \frac{\partial x e^{-\frac{x^2+y^2}{2\delta^2}}}{\partial x} \\ &= \left(-\frac{1}{2\pi\delta^4}\right) \left(1 * e^{-\frac{x^2+y^2}{2\delta^2}} - x e^{-\frac{x^2+y^2}{2\delta^2}} * \frac{2x}{2\delta^2}\right) \\ &= \left(-\frac{1}{2\pi\delta^4}\right) \left(1 - \frac{x^2}{\delta^2}\right) e^{-\frac{x^2+y^2}{2\delta^2}} \end{aligned}$$

同理: $\frac{\partial^2 G}{\partial y^2} = \left(-\frac{1}{2\pi\delta^4}\right) \left(1 - \frac{y^2}{\delta^2}\right) e^{-\frac{x^2+y^2}{2\delta^2}}$

• 高斯函数代入（二维空间中的）拉普拉斯算子，

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

可得 LoG 算子：

$$\begin{aligned}\nabla^2 G(x, y) &= \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2} \\ &= \left(-\frac{1}{2\pi\delta^4}\right)\left(1 - \frac{x^2}{\delta^2}\right)e^{-\frac{x^2+y^2}{2\delta^2}} + \left(-\frac{1}{2\pi\delta^4}\right)\left(1 - \frac{y^2}{\delta^2}\right)e^{-\frac{x^2+y^2}{2\delta^2}} \\ &= \frac{x^2 + y^2 - 2\delta^2}{2\pi\delta^6} e^{-\frac{x^2+y^2}{2\delta^2}}\end{aligned}$$

LOG算子

1. 首先让 LoG 核与一幅输入图像卷积：

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

2. 寻找 $g(x, y)$ 的过零点来确定 $f(x, y)$ 的边缘位置。因为拉普拉斯变换和卷积都是线性运算，因此上式可以改成：

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

其中， $f(x, y)$ 是输入图像， $g(x, y)$ 是输出图像。

1 边缘检测

二阶导数算子

LOG 算子的卷积模板通常采用 5×5 的矩阵，

如：

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

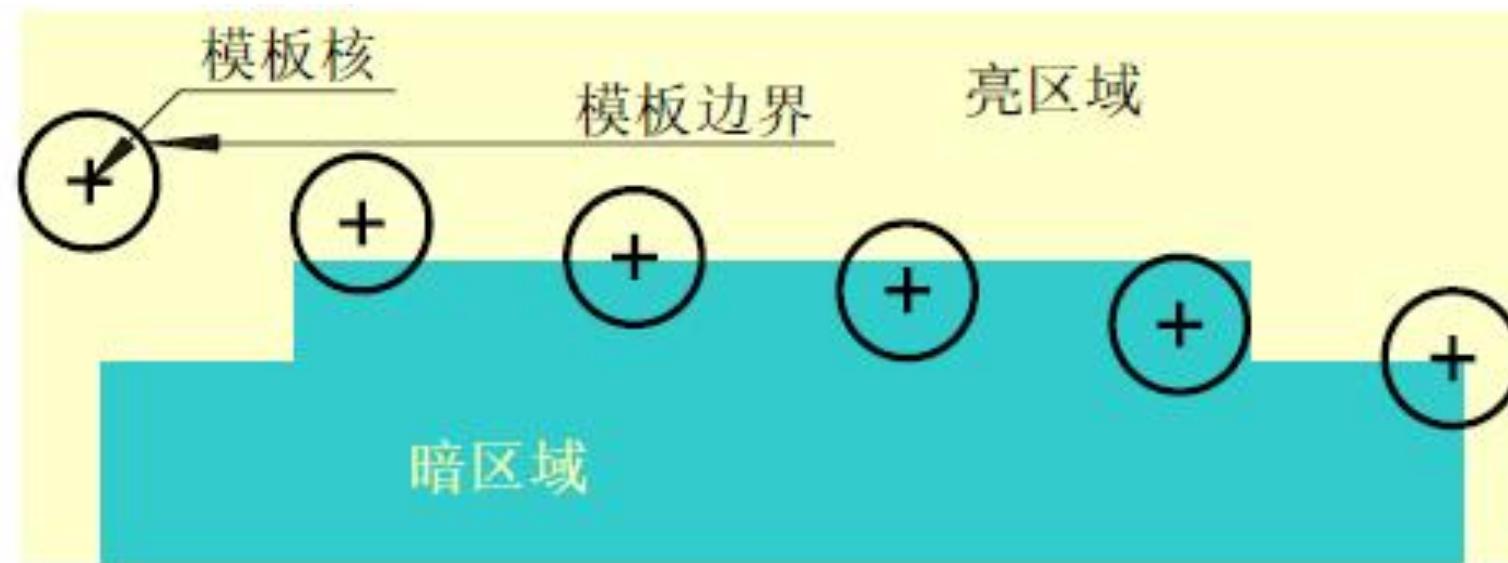
在每个分辨率上进行如下计算：

- (1) 用一个2-D的高斯平滑模板与原图像卷积
- (2) 计算卷积后图像的拉普拉斯值
- (3) 检测拉普拉斯图像中的过零点作为边缘点

2 角点检测

SUSAN算子 (Smallest Univalue Segment Assimilating Nucleus)：使用一个圆形模板得到各向同性的响应。不仅可以检测出图像中的边缘点，而且可以检测出图像中的角点（局部曲率较大的点）。

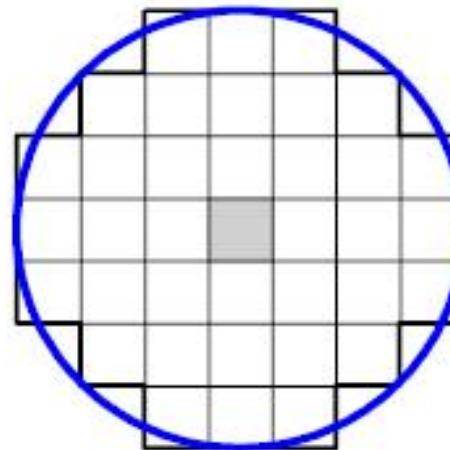
核同值区(uni-value segment assimilating nucleus, USAN) 即与核有相同灰度值的区域。



圆形模板在图像中的不同位置

SUSAN算子的角点检测

圆形模板



37个像素的圆形模板

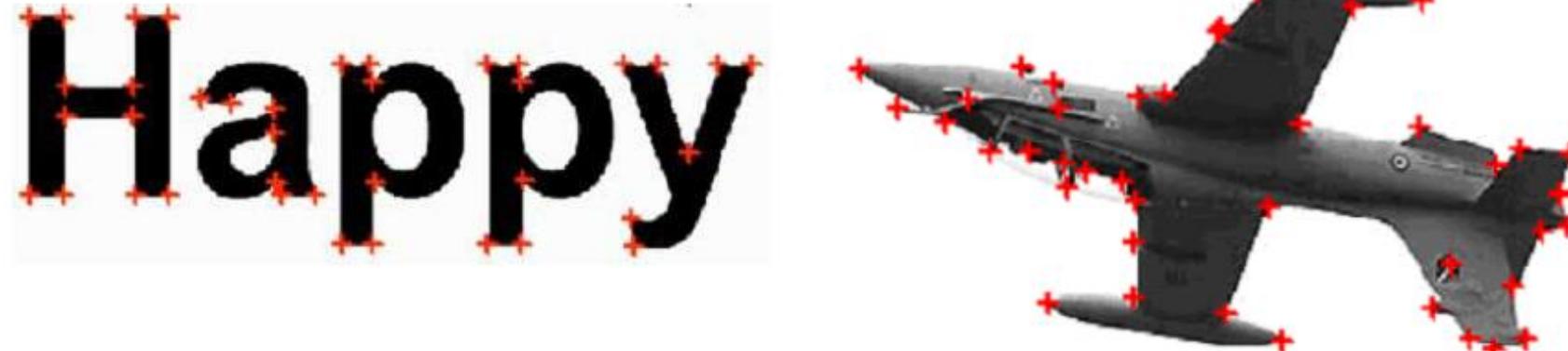
将模板内每个像素的灰度值与核的灰度值进行比较

$$C(x_0, y_0; x, y) = \begin{cases} 1 & \text{当 } |f(x_0, y_0) - f(x, y)| \leq T \\ 0 & \text{当 } |f(x_0, y_0) - f(x, y)| > T \end{cases}$$

SUSAN算子的角点检测

更稳定的计算 $C(\cdot, \cdot)$ 的公式

$$C(x_0, y_0; x, y) = \exp \left\{ - \left[\frac{f(x_0, y_0) - f(x, y)}{T} \right]^2 \right\}$$



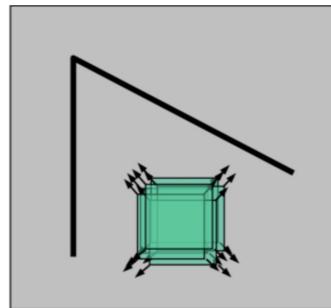
用 SUSAN 算子检测到的角点

SUSAN算子的特点

1. 进行检测时不需要计算微分，这可帮助解释为什么在有噪声时
SUSAN算子的性能会较好。
2. 对边缘的响应将随着边缘的平滑或模糊而增强。
3. SUSAN检测算子能提供不依赖于模板尺寸的边缘精度。
4. 控制参数的选择很简单，且任意性较小。

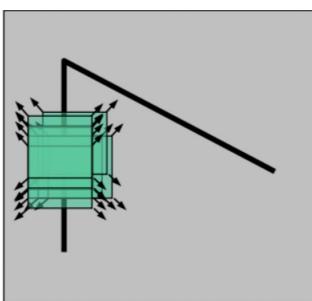
哈里斯角点算子 (Harris)

Harris角点定义：如果在各个方向上移动这个特征的小窗口，窗口内区域的灰度发生了较大的变化，那么就认为在窗口内遇到了**角点**。



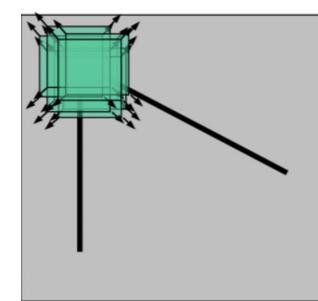
(1) 平坦区域

任意方向移动，无灰度变化



(2) 边缘

沿着边缘方向移动，无灰度变化



(3) 角点

沿任意方向移动，明显灰度变化

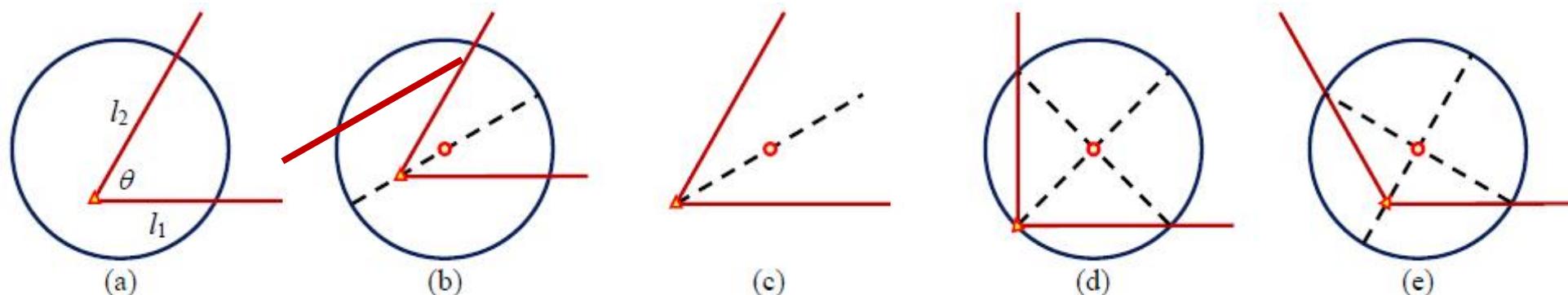
哈里斯角点算子 (Harris)

Harris角点算子其表达矩阵可借助图像中的局部模板里两个方向梯度 I_x 和 I_y 来定义。常见的Harris矩阵：

$$\text{Harris矩阵: } H = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \quad \text{Harris角点强度: } C = \frac{\det(H)}{\text{trace}(H)}$$

在进行角点检测时，矩阵变为：

$$H = \begin{bmatrix} l_2 g^2 \sin^2 \theta & l_2 g^2 \sin \theta \cos \theta \\ l_2 g^2 \sin \theta \cos \theta & l_2 g^2 \cos^2 \theta + l_1 g^2 \end{bmatrix}$$



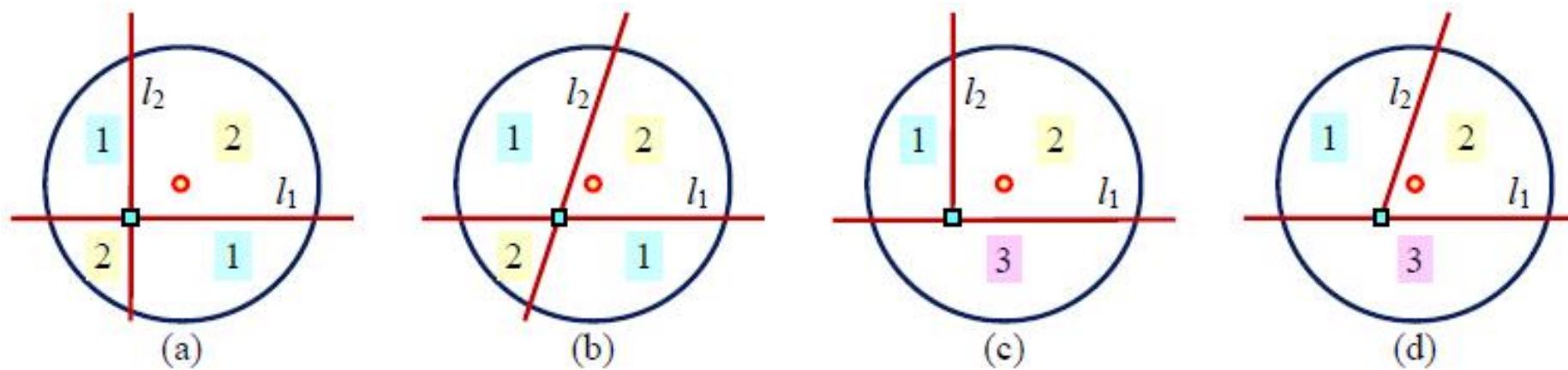
角点与模板的各种位置关系

哈里斯角点算子 (Harris)

交叉点和T型交点检测

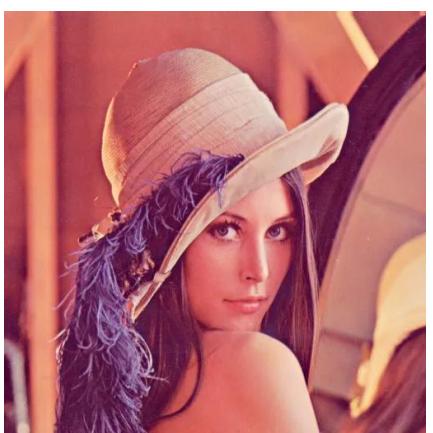
T型交点涉及到三个有不同灰度的区域

$$C = \frac{l_1 l_2 g_1^2 g_2^2}{l_1 g_1^2 + l_2 g_2^2} \sin^2 \theta$$

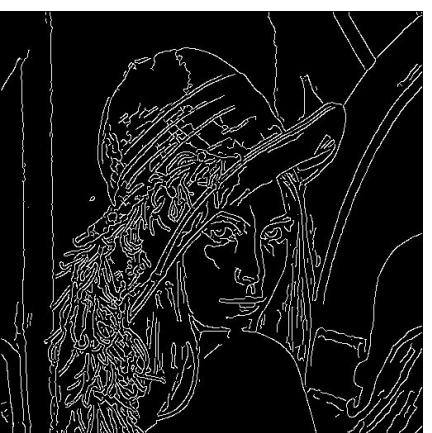


交叉点和T型交点

各算子的实验结果



(a) 原图



(b) Canny算子



(c) Harris算子



(d) Susan算子



(e) prewitt算子



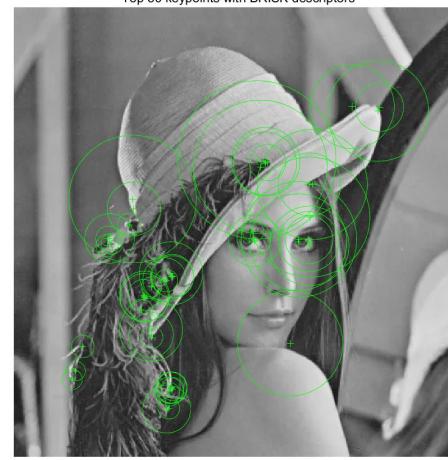
(f) Roberts算子



(g) 马尔算子



(h) FREAK算子



(i) BREAK算子



(j) FAST算子

算法：尺度不变特征变换 (SIFT)

- 尺度不变特征变换 (**Scale-Invariant Feature Transform, SIFT**) 是一种用于图像处理和计算机视觉中的**关键点检测和描述**的算法。它由David Lowe在1999年提出，并在2004年进一步改进。SIFT算法的主要特点是对尺度和旋转的不变性，这使得它在图像匹配、目标识别和3D重建等领域非常流行。

算法：尺度不变特征变换 (SIFT)

SIFT算法主要步骤：

- 1. 尺度空间极值检测：**通过构建尺度空间 (scale-space)，即在不同尺度下观察图像，检测关键点。尺度空间是通过高斯模糊和下采样原始图像来构建的。
- 2. 关键点定位：**在尺度空间中，通过比较每个像素点与其邻域内的点（包括不同尺度和方向）来确定关键点的位置。
- 3. 方向赋值：**为每个关键点分配一个主方向，通常是通过计算关键点邻域内的梯度方向直方图来实现。
- 4. 关键点描述：**生成关键点的描述符，通常是一个向量，包含了关键点周围区域的梯度信息。这个描述符对图像的尺度、旋转和亮度变化具有鲁棒性。
- 5. 匹配：**使用关键点的描述符来匹配不同图像中的关键点，从而实现图像之间的对应关系。

算法：尺度不变特征变换 (SIFT)

SIFT算法的优点：

- **尺度不变性**：通过在多个尺度上检测关键点，SIFT能够识别出在不同尺度下的特征。
- **旋转不变性**：关键点的方向赋值使得SIFT能识别出旋转后的特征。
- **亮度和对比度变化的鲁棒性**：SIFT描述符的设计使得它对图像的亮度和对比度变化不敏感。
- **特征点的稳定性**：SIFT算法能够检测出稳定的关键点，即使在噪声和部分遮挡的情况下也能保持较好的性能。

SIFT应用场景

- 1. 图像搜索与识别：**SIFT算法可以用于图像搜索引擎，帮助用户通过上传图片快速找到相似的图像或相关产品。
- 2. 增强现实 (AR)：**在增强现实应用中，SIFT算法可以用来识别现实世界中的物体或场景，从而在用户的视野中叠加虚拟信息或图像。
- 3. 机器人导航与地图构建：**SIFT算法在机器人领域中用于环境感知，帮助机器人通过图像匹配进行定位和路径规划。
- 4. 全景图像拼接：**SIFT算法能够处理不同视角和尺度下的图像，使其成为全景图像拼接的理想选择，广泛应用于虚拟现实 (VR) 和街景地图服务。
- 5. 物体识别与追踪：**在零售和安防监控领域，SIFT算法可以用于物体的识别和追踪，例如在视频监控中识别和跟踪人员或车辆。
- 6. 3D重建：**SIFT算法在3D建模中用于从多个2D图像中提取特征点，进而重建出3D模型，这在电影制作、游戏开发和建筑可视化中非常有用。
- 7. 无人机导航：**无人机使用SIFT算法进行环境映射和目标识别，以实现自主导航和避障。
- 8. 医学成像分析：**在医疗领域，SIFT算法可以帮助分析医学图像，比如通过比较不同时间点的图像来检测病变或进行手术规划。
- 9. 遥感图像处理：**在遥感领域，SIFT算法用于从卫星或航空图像中提取特征，进行地理信息系统 (GIS) 分析和环境监测。
- 10. 内容创作与管理：**SIFT算法在内容管理系统中用于图像版权检测、图像分类和图像数据库管理。

算法：SURF (Speeded Up Robust Features)

- **SURF (Speeded-Up Robust Features)** 是SIFT算法的加速优化版，核心目标是在保持尺度、旋转不变性的同时，大幅提升特征提取速度。
 1. **Hessian矩阵关键点检测**：通过计算图像的Hessian矩阵行列式（衡量局部灰度变化的剧烈程度）定位稳定关键点；
 2. **积分图像加速**：利用积分图像快速计算Hessian矩阵，将卷积运算复杂度降低；
 3. **主方向分配**：通过Harr小波响应确定关键点的主方向（旋转不变性的基础）；
 4. **特征描述符生成**：以关键点为中心，提取64/128维向量作为描述符（区分不同特征）。

SURF算法由Bay、Tuytelaars和Van Gool于2006年提出，是经典SIFT算法的优化版——既保留了SIFT的鲁棒性，又通过盒式滤波器（Box Filter）替代高斯滤波器，将计算速度提升了3倍以上。