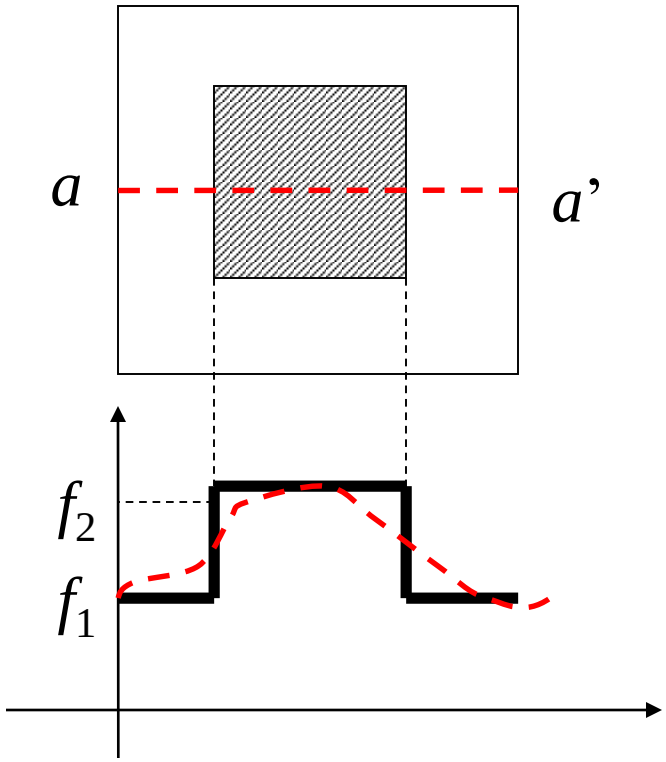
2.3 锐化

一、锐化：

指增强图像的边缘和线条，使图像的边缘和线条由模糊转化为清晰的处理方法。

- **图像清晰：**指图像中边缘和线条相邻像素间有较大的灰度差(即存在灰度突变)如f_1到f_2的变化；
- **图像模糊：**指边缘的灰度突变消失了，从背景到轮廓的灰度变化趋于平缓，如右图红色虚线所示；



a-*a*’线上的灰度分布

Lena 原始图像

用 Soble 算子处理后的图像

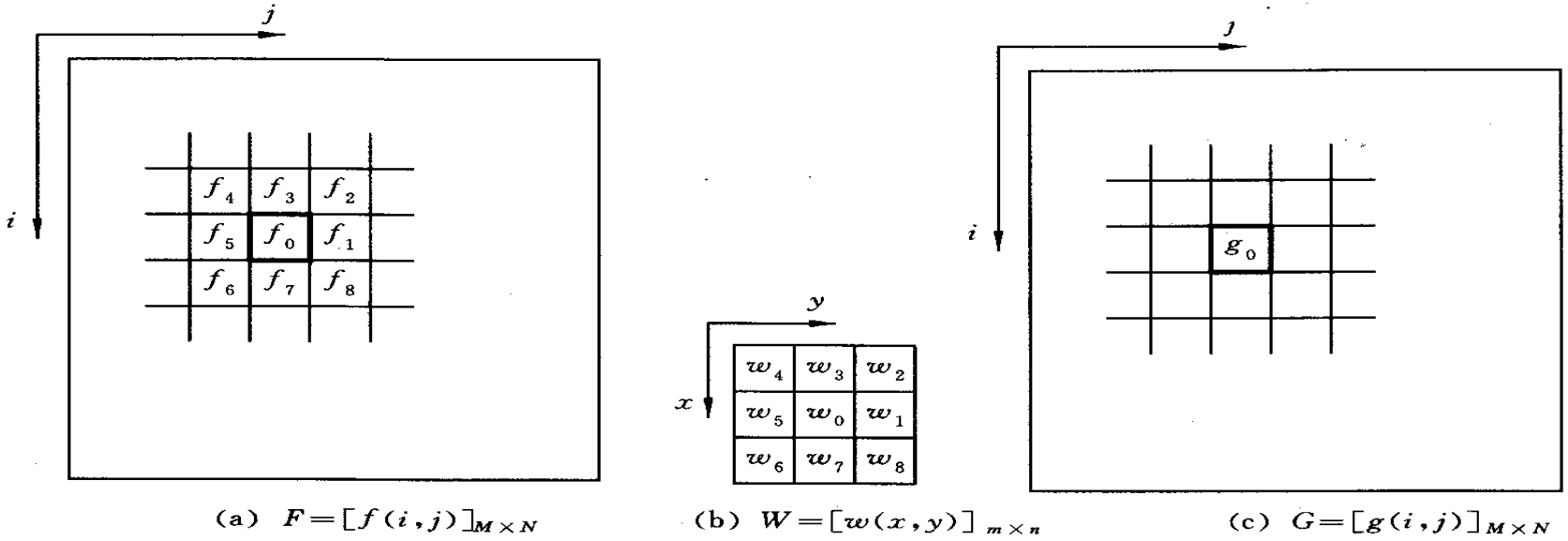
二、锐化处理方法：

1、卷积：指对图像各像素的邻域进行加权求和的计算

- 卷积时的权值 ω 称为卷积权函数；
- **卷积权函数形式**：矩阵、模板、算子等；
- **邻域范围（$m \times n$）**：由运算的模板(算子等)的大小来确定，

  常用的有3×3、5×5、7×7、9×9等

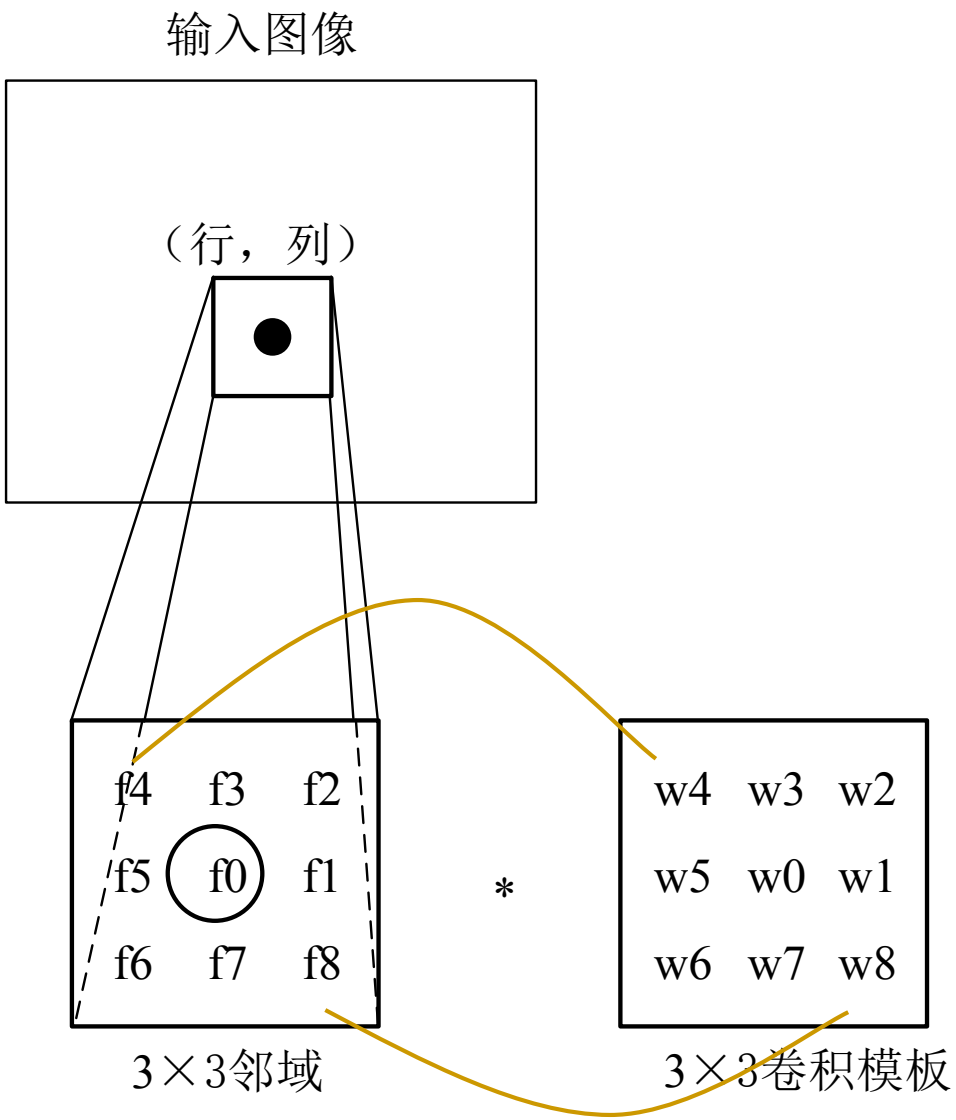
卷积运算：



(a) $F=[f(i,j)]_{M\times N}$　(b) $W=[w(x,y)]_{m\times n}$ $m\times n=3\times 3$　(c) $G=[g(i,j)]_{M\times N}$

图　卷积

(a) 输入图像；(b) 卷积权函数；(c) 输出图像



加权求和计算：

f0×w0+
f1×w1+
f2×w2+
f3×w3+
f4×w4+
f5×w5+
f6×w6+
f7×w7+
f8×w8

---

g0的新值

卷积运算示意图

卷积结果：

输出图像：

对应于f_0的卷积值$g_0 = \omega_0 * f_0 + \omega_1 * f_1 + \omega_2 * f_2 + \ldots.. + \omega_8 * f_8$

$$g(i,j) = \sum_{x=1}^{m}\sum_{y=1}^{n} \varpi(x,y) f\left(i + x - \frac{m+1}{2}, j + y - \frac{n+1}{2}\right)$$

存在问题与处理方法：

问题1：卷积值可能大大超过了允许的灰度范围；
对策：按比例缩小所有像素的灰度值。

问题2：卷积结果出现负值；
对策：取绝对值、置0或加一个常数。

问题3：边界像素的邻域问题；
对策：边界点不处理、数据外推法、边界外灰度值置0

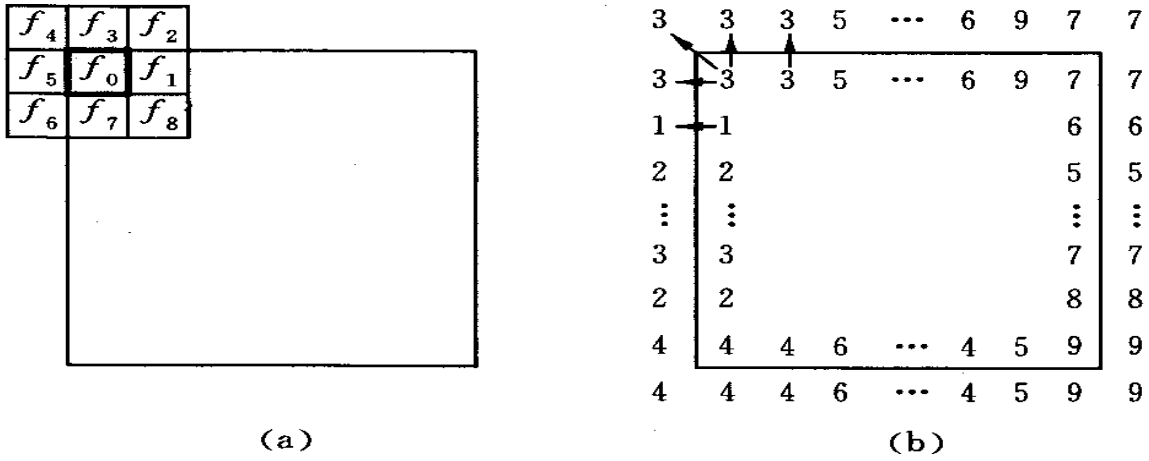


图　图像边界像素的邻域

(a) 边界像素的邻域值；(b) 数据外推法

说明卷积过程中存在的问题及其解决方法，并选择其中解决边界问题的方法之一，对给定的图像(图a)用卷积模板(图b)进行卷积处理，求卷积后图像

1	1	1	1
1	1	4	1
1	1	1	1
1	1	1	1

(图a)

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

(图b)

1	1	1	1
1	1	4	1
1	1	1	1
1	1	1	1

卷积处理

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

请在实验课上实现

2、差分法

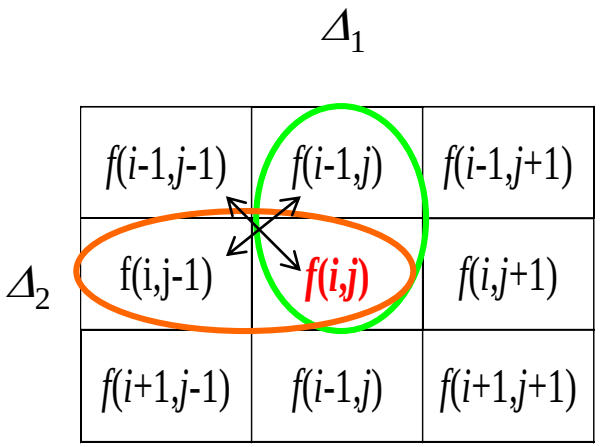
差分法是数学上用离散函数的数值计算方法对连续函数微分运算的一种近似，而数字图像就是模拟图像的离散形式，可以用差分法的原理对数字图像进行处理。

差分与微分一样有一阶差分和二阶差分。

- 一阶差分：指数字图像上相邻像素间的灰度差；
- 二阶差分：指一阶差分的差值 。

一阶差分的基本形式：

$$\Delta_1 f(i,j) = f(i,j) - f(i-1,j)$$
$$\Delta_2 f(i,j) = f(i,j) - f(i,j-1)$$
$$\Delta_3 f(i,j) = f(i-1,j-1) - f(i,j)$$
$$\Delta_4 f(i,j) = f(i-1,j) - f(i,j-1)$$



3X3邻域中各像素的行列坐标

Δ_1垂直方向的灰度差

Δ_2水平方向的的灰度差

Δ_3、Δ_4为两个对角边上的灰度差

一阶差分的卷积模板：

垂直差分法　　水平差分法　　交叉差分法

0	-1	0
0	1	0
0	0	0

Δ_1

0	0	0
-1	1	0
0	0	0

Δ_2

1	0	0
0	-1	0
0	0	0

Δ_3

0	1	0
-1	0	0
0	0	0

Δ_4

二阶差分的基本形式与模板：

垂直方向：

$$\Delta_1^2 f(i,j) = \Delta_1 f(i+1,j) - \Delta_1 f(i,j)$$
$$= [f(i+1,j) - f(i,j)] - [f(i,j) - f(i-1,j)]$$
$$= f(i+1,j) + f(i-1,j) - 2f(i,j)$$

0	1	0
0	- 2	0
0	1	0

Δ_1^2

水平方向：

$$\Delta_2^2 f(i,j) = f(i,j+1) + f(i,j-1) - 2f(i,j)$$

0	0	0
1	- 2	1
0	0	0

Δ_2^2

差分算子-1：

- 差分法可通过差分算子来实现增强图像边缘的锐化：

A、梯度算子$E(i, j)$：指用垂直方向和水平方向的一阶差分值计算最大差分方向的一阶差分算子$E_1(i, j)$

- 其中：

$$E_1(i, j) = \sqrt{[\Delta_1 f(i, j)]^2 + [\Delta_2 f(i, j)]^2}$$

- 梯度算子的其它形式：

一阶差分**绝对值算子**、一阶差分最大值算子

$$\approx |f(i, j) - f(i+1, j)| + |f(i, j) - f(i, j+1)|$$

3、图像锐化

一、梯度法

第1种：各点的灰度$g(x, y)$等于该点的梯度幅度

$$g(x, y)=E\left[f(x, y)\right]$$

特点是增强的图像仅显示灰度变化比较陡的边缘轮廓，而灰度变化平缓的区域则呈黑色(0)。

一、梯度法

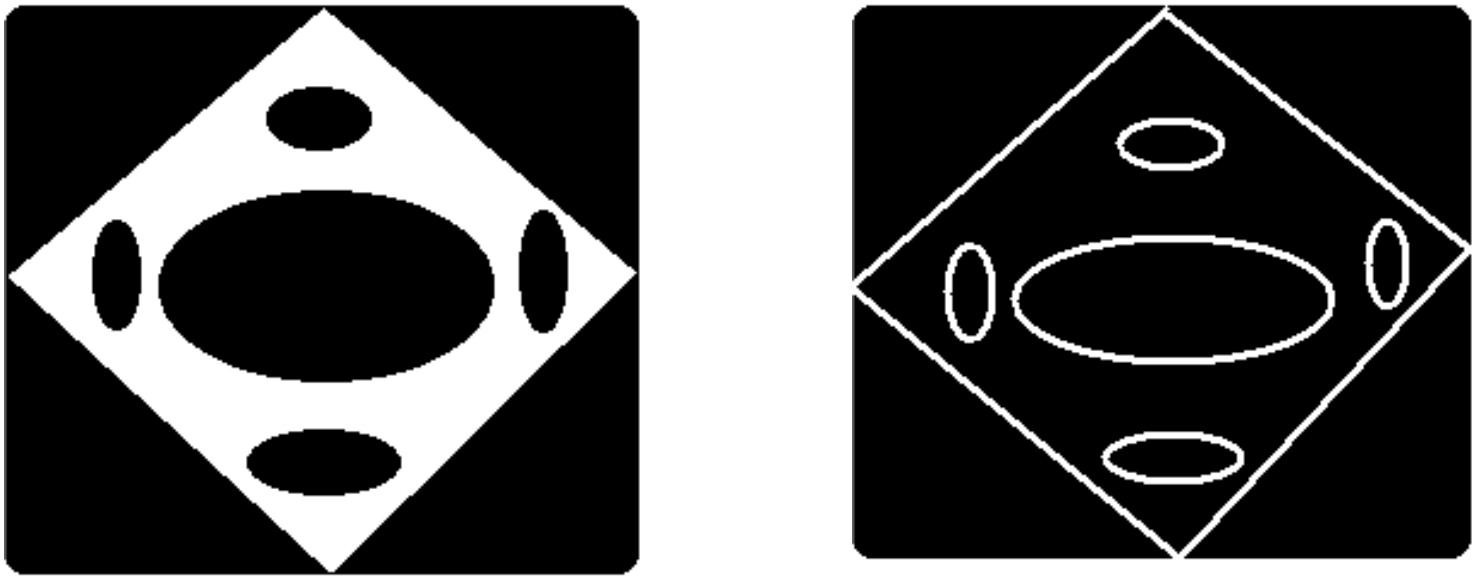
```
clc
ImageIn = imread('f:\pics\lenaGray.tiff');
ImageOut = ImageIn;
for i1 = 1 : size(ImageIn,1)-1
  for i2 = 1 : size(ImageIn,2)-1
    c = ImageIn(i1, i2);
    c1 = ImageIn (i1 + 1, i2);
    c2 = ImageIn (i1, i2 + 1);
    ImageOut(i1,i2)= abs(c - c1) + abs(c - c2);
  end
end

imwrite(ImageOut,'f:\pics\lenaGraySharp.tiff')

subplot(1,2,1);%'原图像'
imshow(ImageIn)
title('\fontsize{16}\fontname{黑体}原图像')
subplot(1,2,2);%'新图像'
imshow(ImageOut)
title('\fontsize{16}\fontname{黑体}新图像')
```

$$|f(i,j)-f(i+1,j)|+|f(i,j)-f(i,j+1)|$$

一、梯度法



图像梯度锐化结果
(a) 二值图像； (b) 梯度运算结果

一、梯度法

第2种：增强的图像

$$g(x,y)=\begin{cases}E[f(x,y)] & E[f(x,y)]\geq T\\ f(x,y) & \text{其他}\end{cases}$$

式中：T 是一个非负的阈值，适当选取T，即可使明显的边缘轮廓得到突出，又不会破坏原灰度变化比较平缓的背景。

一、梯度法

```
clc
ImageIn = imread('f:\pics\lenaGray.tiff');
ImageOut = ImageIn;
for i1 = 1 : size(ImageIn,1)-1
  for i2 = 1 : size(ImageIn,2)-1
    c = ImageIn(i1, i2); c1 = ImageIn (i1 + 1, i2); c2 = ImageIn
    b = abs(c - c1) + abs(c - c2);
    if b > 10
      ImageOut(i1,i2)= b;
    end
  end
end

subplot(1,2,1);%'原图像'
imshow(ImageIn)
title('\fontsize{16}\fontname{黑体}原图像')
subplot(1,2,2);%'新图像'
imshow(ImageOut)
title('\fontsize{16}\fontname{黑体}新图像')
```

$$g(x,y)=\begin{cases}E[f(x,y)] & E[f(x,y)]\geq T\\ f(x,y) & \text{其他}\end{cases}$$

一、梯度法

第3种：

$$g(x,y)=\begin{cases}L_G & E[f(x,y)]\geq T\\ f(x,y) & \text{其他}\end{cases}$$

式中：T是根据需要指定的一个灰度级，它将明显边缘用一固定的灰度级 L_G 来实现。

一、梯度法

```
ImageIn = imread('f:\pics\lenaGray.tiff');
ImageOut = ImageIn;
for i1 = 1 : size(ImageIn,1)-1
  for i2 = 1 : size(ImageIn,2)-1
    c = ImageIn(i1, i2);
    c1 = ImageIn (i1 + 1, i2);
    c2 = ImageIn (i1, i2 + 1);
    b = abs(c - c1) + abs(c - c2);
    if b > 10
      ImageOut(i1,i2) = 0;
    end
  end
end

subplot(1,2,1);%'原图像'
imshow(ImageIn)
title('\fontsize{16}\fontname{黑体}原图像')
subplot(1,2,2);%'新图像'
imshow(ImageOut)
title('\fontsize{16}\fontname{黑体}新图像')
```

$$g(x,y)=\begin{cases} L_G & E[f(x,y)]\geq T \\ f(x,y) & \text{其他} \end{cases}$$

一、梯度法

第4种：

$$g(x,y)=\begin{cases}E[f(x,y)] & E[f(x,y)]\geq T\\ L_G & \text{其他}\end{cases}$$

此法将背景用一个固定灰度级 L_G 来实现，便于研究边缘灰度的变化。

一、梯度法

```
ImageIn = imread('f:\pics\lenaGray.tiff');
ImageOut = ImageIn;
for i1 = 1 : size(ImageIn,1)-1
  for i2 = 1 : size(ImageIn,2)-1
    c = ImageIn(i1, i2);
    c1 = ImageIn (i1 + 1, i2);
    c2 = ImageIn (i1, i2 + 1);
    b = abs(c - c1) + abs(c - c2);
    if b > 10
      ImageOut(i1,i2)= b;
    else
      ImageOut(i1,i2)= 128;
    end
  end
end

subplot(1,2,1);%'原图像'
imshow(ImageIn)
title('\fontsize{16}\fontname{黑体}原图像')
subplot(1,2,2);%'新图像'
imshow(ImageOut)
title('\fontsize{16}\fontname{黑体}新图像')
```

$$g(x,y)=\begin{cases} E[f(x,y)] & E[f(x,y)]\geq T \\ L_G & \text{其他} \end{cases}$$

一、梯度法

第5种：

$$g(x,y)=\begin{cases}L_G & E[f(x,y)]\geq T\\ L_B & \text{其他}\end{cases}$$

此法将背景和边缘用二值图像表示， 便于研究边缘所在位置。

一、梯度法

```
ImageIn = imread('f:\pics\lenaGray.tiff');
ImageOut = ImageIn;
for i1 = 1 : size(ImageIn,1)-1
  for i2 = 1 : size(ImageIn,2)-1
    c = ImageIn(i1, i2); c1 = ImageIn (i1 + 1, i2); c2 = Imag
    b = abs(c - c1) + abs(c - c2);
    if b > 10
      ImageOut(i1,i2)= 0;
    else
      ImageOut(i1,i2)= 255;
    end
  end
end

subplot(1,2,1);%'原图像'
imshow(ImageIn)
title('\fontsize{16}\fontname{黑体}原图像')
subplot(1,2,2);%'新图像'
imshow(ImageOut)
title('\fontsize{16}\fontname{黑体}新图像')
```

$$g(x,y)=\begin{cases} L_G & E[f(x,y)]\geq T \\ L_B & \text{其他} \end{cases}$$

二、差分算子-罗伯特(Roberts)算子

罗伯特算子是一阶差分算子,又称交叉差分算子

- 计算式:

$$R_1(i,j)=\sqrt{[\Delta_3 f(i,j)]^2+[\Delta_4 f(i,j)]^2}$$

- 罗伯特算子其它两种形式:

  一阶差分绝对值算子:

$$R_2(i,j)=\left|\Delta_3 f(i,j)\right|+\left|\Delta_4 f(i,j)\right|$$

  一阶差分最大值算子:

$$R_3(i,j)=\max\{\left|\Delta_3 f(i,j)\right|,\left|\Delta_4 f(i,j)\right|\}$$

三、差分算子-索伯尔(Sobel)算子

采用梯度微分锐化图像，同时会使噪声、条纹得到增强，索伯尔(Sobel)算子则在一定程度上克服了这个问题。

用Sobel算子计算3×3窗口的灰度(卷积)， 将其作为变换后图像$g(i,j)$的灰度。

$$
\begin{array}{ccc}
f(i-1,j-1) & f(i-1,j) & f(i-1,j+1) \\
\bullet & \bullet & \bullet \\
f(i,j-1) & f(i,j) & f(i,j+1) \\
\bullet & \bullet & \bullet \\
f(i+1,j-1) & f(i+1,j) & f(i+1,j+1) \\
\bullet & \bullet & \bullet
\end{array}
$$

三、差分算子-索伯尔(Sobel)算子

方向模板：

其中d_1f（i，j）、d_2f（i，j）为d_1、d_2与像素f（i，j）的卷积

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

d_1

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

d_2

- 计算式:

$$S_1(i,j)=\sqrt{[d_1f(i,j)]^2+[d_2f(i,j)]^2}$$

$$S_2(i,j)=|d_1f(i,j)|+|d_2f(i,j)|$$

三、差分算子-索伯尔(Sobel)算子

Sobel算子不像普通梯度算子那样用两个像素的差值，

因此有了以下两个优点：

（1） 由于引入了平均因素，因而对图像中的随机噪声有一定的平滑作用。

（2） 由于它是相隔两行或两列之差分，故边缘两侧元素得到了增强，边缘显得粗而亮。

三、差分算子-索伯尔(Sobel)算子

```
clc
ImageIn = imread('f:\pics\lenaGray.tiff');
subplot(1,2,1);%'原图像'
imshow(ImageIn)
ImageOut = ImageIn;
ImageIn = double(ImageIn);
Sobel1 = [-1,0,1;-2,0,2;-1,0,1];
Sobel2 = Sobel1';
for i1 = 2 : size(ImageIn,1)-1
  for i2 = 2 : size(ImageIn,2)-1
    ImageOut(i1,i2) = abs(sum(sum(ImageIn(i1-1:i1+1,i2-1:i2+1) .* Sobel1))) +
abs(sum(sum(ImageIn(i1-1:i1+1,i2-1:i2+1) .* Sobel2)));
  end
end

title('\fontsize{16}\fontname{黑体}原图像')
subplot(1,2,2);%'新图像'
imshow(ImageOut)
title('\fontsize{16}\fontname{黑体}新图像')
```

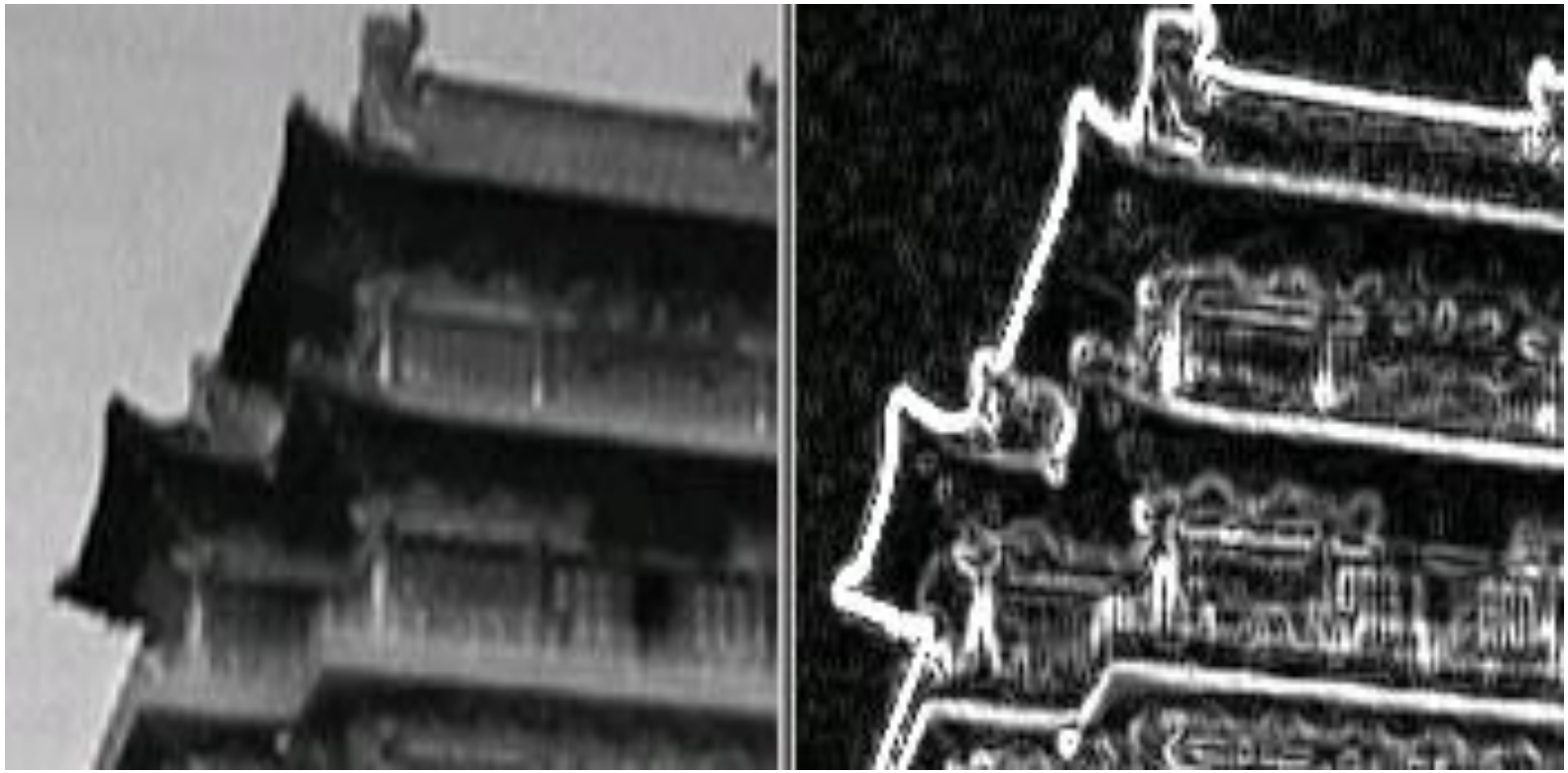
-1	0	1
-2	0	2
-1	0	1

d_1

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

d_2

三、差分算子-索伯尔(Sobel)算子



四、差分算子-拉普拉斯(Laplacian)算子

- 拉普拉斯算子是二阶差分算子,为垂直方向和水平方向二阶差分之和；

$$L(i,j)=\Delta_1^2 f(i,j)+\Delta_2^2 f(i,j)=[f(i+1,j)+f(i-1,j)-2f(i,j)]+[f(i,j+1)+f(i,j-1)-2f(i,j)]$$
$$=f(i+1,j)+f(i-1,j)+f(i,j+1)+f(i,j-1)-4f(i,j)$$

- **锐化后的图像*g*为 *g=f(i,j)-L(i,j)***

- 模板表示：

0	1	0
0	-2	0
0	1	0

+

0	0	0
1	-2	1
0	0	0

=

0	1	0
1	-4	1
0	1	0

是一种各向同性（旋转不变性）的线性运算

四、差分算子-拉普拉斯(Laplacian)算子

```
clc
ImageIn = imread('f:\pics\lenaGray.tiff');
subplot(1,2,1);%'原图像'
imshow(ImageIn)
title('\fontsize{16}\fontname{黑体}原图
ImageOut = ImageIn;
ImageIn = double(ImageIn);
Laplacian = [0,1,0;1,-4,1;0,1,0];
iOperator = Laplacian;
for i1 = 2 : size(ImageIn,1)-1
  for i2 = 2 : size(ImageIn,2)-1
    ImageOut(i1,i2) = sum(sum(ImageIn(i1-1:i1+1,i2-1:i2+1)
  end
end
imwrite(ImageOut,'f:\pics\lenaGraySharp.tiff')
subplot(1,2,2);%'新图像'
imshow(ImageOut)
title('\fontsize{16}\fontname{黑体}新图像')
```

0	1	0
1	-4	1
0	1	0

四、差分算子-拉普拉斯(Laplacian)算子

拉普拉斯锐化结果

（a） 二值图像；（b） 拉普拉斯运算结果

四、差分算子-拉普拉斯(Laplacian)算子

其它常用的拉普拉斯算子（模板）：

$$H=\begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix} \qquad H=\begin{bmatrix}1 & -2 & 1\\ -2 & 4 & -2\\ 1 & -2 & 1\end{bmatrix} \qquad H=\begin{bmatrix}0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{bmatrix}$$

模板之和为**0**，在灰度平稳地方是黑色

模板之和不为**0**（一般为**1**），在灰度平稳地方是其它灰度值。

五、高通滤波法

由于边缘的高频特性可知，为使图像得到锐化效果，可对图像进行高通滤波，使高频分量顺利通过，而低频分量受到抑制，从而达到突出高频分量的目的。

- **空域高通滤波：**

  指在空间域对图像进行高通滤波处理的方法；

- 频域高通滤波：

  指在频率域对图像进行高通滤波处理的方法 。

五、高通滤波法

- 空域高通滤波：

  空域高通滤波采用高通滤波模板的卷积运算来实现，常用高通滤波模板：

0	-1	0
-1	5	-1
0	-1	0

H_1

-1	-1	-1
-1	9	-1
-1	-1	-1

H_2

1	-2	1
-2	5	-2
1	-2	1

H_3

-1	$\frac{1}{2}$	-1
$\frac{1}{2}$	3	$\frac{1}{2}$
-1	$\frac{1}{2}$	-1

H_4

习　题

1、图像锐化的目的是什么？有哪些可以实现的方法？

2、编程实现卷积处理（3×3）。