# 第7章 图像锐化

主讲教师　张涛

电子信息与电气工程

## 问题的提出：

- 对人眼视觉系统的研究表明，人类对形状的感知一般通过识别边缘、轮廓、前景和背景形成。因此，在图像处理中，边缘信息十分重要，数字图像的边缘检测是图像锐化、图像分割、区域形状特征提取等技术的重要基础。
- 如何检测边缘轮廓信息？

# 第7章 图像锐化

## 问题的提出：

- 边缘是图像中亮度突变的区域，可通过计算局部图像区域的亮度差异，检测出不同目标或场景各部分之间的边界。
- 图像锐化（**Image Sharpening**）：加强图像中景物的边缘和轮廓，突出图像中的细节或者增强被模糊了的细节。

# 主要内容

7.1 图像边缘分析

7.2 一阶微分算子

7.3 二阶微分算子

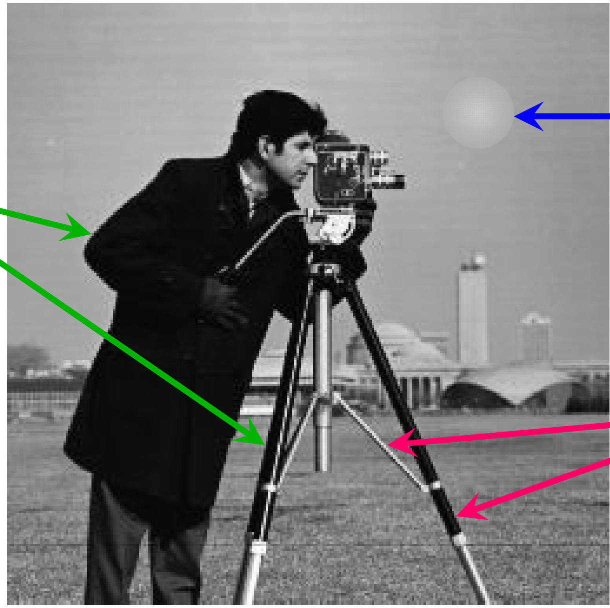
7.4 高斯滤波与边缘检测

7.5 频域高通滤波

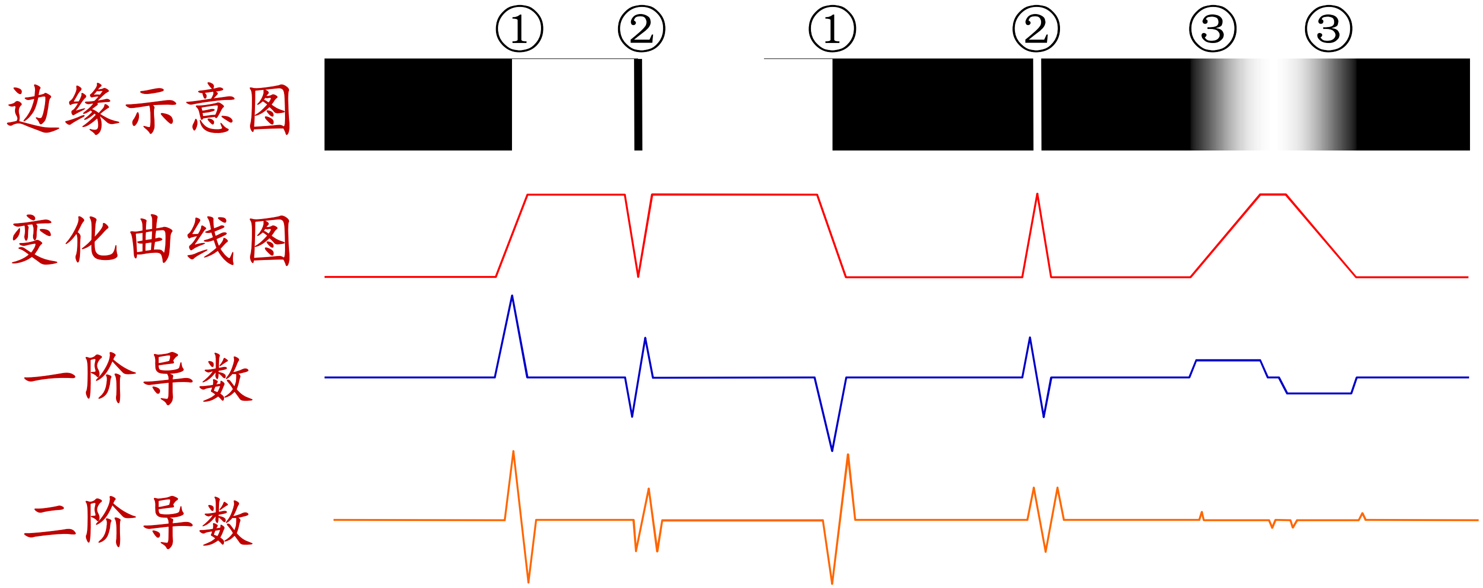
7.6 基于小波变换的边缘检测

# 7.1 图像边缘分析

- 图像中的边缘主要有以下几种类型：细线型边缘、突变型边缘和渐变型边缘



# 7.1 图像边缘分析




- 突变型细节：检测一阶微分极值点，二阶微分过0点
- 细线型细节：检测一阶微分过0点，二阶微分极值点
- 渐变型细节：难检测，二阶微分信息略多于一阶微分

# 7.2 一阶微分算子

7.2.1 梯度算子

7.2.2 Robert算子

7.2.3 Sobel算子

7.2.4 Prewitt算子

## 7.2.1 梯度算子

### （1）定义

- 在图像处理中应用微分最常用的方法是计算梯度
- 对于图像函数*f*(*x*,*y*)，
  它在(*x*,*y*)处的梯度为 $G[f(x,y)]=\begin{bmatrix}\dfrac{\partial f}{\partial x} & \dfrac{\partial f}{\partial y}\end{bmatrix}^T$
  用梯度的幅度来代替

$$G[f(x,y)]=\left[\left(\frac{\partial f}{\partial x}\right)^2+\left(\frac{\partial f}{\partial y}\right)^2\right]^{\frac{1}{2}}$$ 或 $$G[f(x,y)]=\left|\frac{\partial f}{\partial x}\right|+\left|\frac{\partial f}{\partial y}\right|$$

## 7.2.1 梯度算子

### （1）定义

- 离散的数字矩阵，用差分来代替微分

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

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

$$g(x,y)=\left|f(x+1,y)-f(x,y)\right|+\left|f(x,y+1)-f(x,y)\right|$$

$g(x,y)$ 称为梯度图像

（2）边缘检测

- 对梯度图像进行阈值化，检测局部变化极值

固定边界灰度

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

突出边界

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

二值化边界与背景

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

此处，阈值的选择决定边缘检测效果

（3）示例

| | | | | |
|---|---|---|---|---|
| 3 | 3 | 3 | 3 | 3 |
| 3 | 7 | 7 | 7 | 3 |
| 3 | 7 | 7 | 7 | 3 |
| 3 | 7 | 7 | 7 | 3 |
| 3 | 3 | 3 | 3 | 3 |

→

| | | | | |
|---|---|---|---|---|
| 0 | 4 | 4 | 4 | 0 |
| 4 | 0 | 0 | 4 | 0 |
| 4 | 0 | 0 | 4 | 0 |
| 4 | 4 | 4 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 |

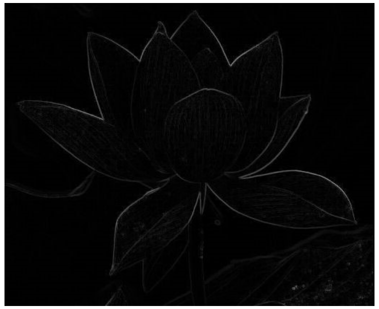
（4）例程

```
Image=im2double(rgb2gray(imread('lotus.jpg')));
subplot(131),imshow(Image),title('原图像');
[h,w]=size(Image);    edgeImage=zeros(h,w);
for x=1:w-1
    for y=1:h-1
        edgeImage(y,x)=abs(Image(y,x+1)-Image(y,x))
                            +abs(Image(y+1,x)-Image(y,x));
    end
end
subplot(132),imshow(edgeImage),title('梯度图像');
sharpImage=Image+edgeImage;
subplot(133),imshow(sharpImage),title('锐化图像');
```

# 7.2.1 梯度算子

## （4）例程

原图



梯度图像

锐化图像

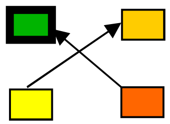
原图

梯度图像

# 7.2.2 Robert算子



## （1）定义



- 交叉求微分

$$g(x,y)=\left|f(x,y)-f(x+1,y+1)\right|+\left|f(x+1,y)-f(x,y+1)\right|$$

- 用模板表示为

$$H_1=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \qquad H_2=\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}$$

# 7.2.2 Robert算子

（2）示例

$$\begin{bmatrix} \begin{bmatrix} 3 & 3 \\ 3 & 7 \end{bmatrix} & \begin{matrix} 3 & 3 & 3 \\ 7 & 7 & 3 \end{matrix} \\ \begin{matrix} 3 & 7 \\ 3 & 7 \\ 3 & 3 \end{matrix} & \begin{matrix} 7 & 7 & 3 \\ 7 & 7 & 3 \\ 3 & 3 & 3 \end{matrix} \end{bmatrix}$$

$$g(0,0)$$
$$=\left|f(0,0)-f(1,1)\right|+\left|f(1,0)-f(0,1)\right|$$
$$=|3-7|+|3-3|=4$$

$$得\ g=\begin{bmatrix} 4 & 8 & 8 & 4 & 0 \\ 8 & 0 & 0 & 8 & 0 \\ 8 & 0 & 0 & 8 & 0 \\ 4 & 8 & 8 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

## 7.2.2 Robert算子



### （3）例程

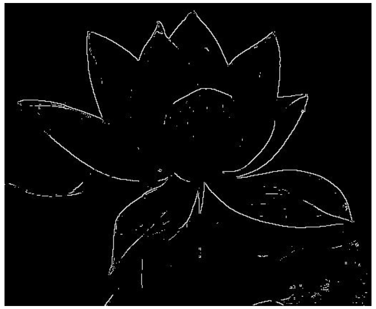
- 函数：**BW=edge(I,TYPE,PARAMETERS)**
- 程序

```
Image=im2double(rgb2gray(imread('lotus.jpg')));
BW= edge(Image,'roberts');
figure,imshow(BW),title('边缘检测');
H1=[1 0; 0 -1];                    H2=[0 1;-1 0];
R1=imfilter(Image,H1);     R2=imfilter(Image,H2);
edgeImage=abs(R1)+abs(R2);
figure,imshow(edgeImage),title('Robert梯度图像');
sharpImage=Image+edgeImage;
figure,imshow(sharpImage),title('Robert锐化图像');
```

# 7.2.2 Robert算子

## （3）例程

梯度图像



边缘检测

锐化图像

原图

梯度图像

# 7.2.3 Sobel算子

（1）定义

$$S_x = \left| f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1) \right| - \left| f(x-1,y-1) + 2f(x,y-1) + f(x+1,y-1) \right|$$

$$S_y = \left| f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1) \right| - \left| f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1) \right|$$

$$g = \left| S_x \right| + \left| S_y \right|$$

$$H_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad H_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

# 7.2.3 Sobel算子



（2）示例

$$\begin{bmatrix} 3 & 3 & 3 & 3 & 3 \\ 3 & 7 & 7 & 7 & 3 \\ 3 & 7 & 7 & 7 & 3 \\ 3 & 7 & 7 & 7 & 3 \\ 3 & 3 & 3 & 3 & 3 \end{bmatrix} \qquad g = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 24 & 16 & 24 & 0 \\ 0 & 16 & 0 & 16 & 0 \\ 0 & 24 & 16 & 24 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- 效果分析
  - 引入平均因素，对图像中随机噪声有一定的平滑作用
  - 相隔两行或两列求差分，故边缘两侧的元素得到了增强，边缘显得粗而亮

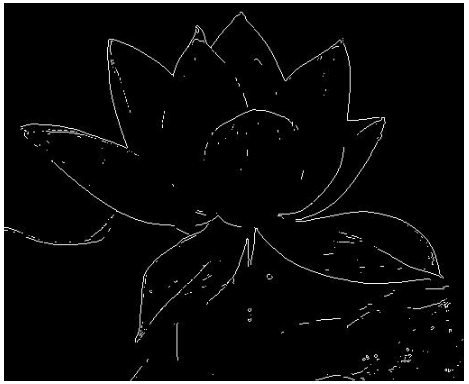
## 7.2.3 Sobel算子

（3）例程

```
Image=im2double(rgb2gray(imread('lotus.jpg')));
figure,imshow(Image),title('原图像');
BW= edge(Image,'sobel');
figure,imshow(BW),title('边缘检测');
H1=[-1 -2 -1;0 0 0;1 2 1];   H2=[-1 0 1;-2 0 2;-1 0 1];
R1=imfilter(Image,H1);    R2=imfilter(Image,H2);
edgeImage=abs(R1)+abs(R2);
figure,imshow(edgeImage),title('Sobel梯度图像');
sharpImage=Image+edgeImage;
figure,imshow(sharpImage),title('Sobel锐化图像');
```

# 7.2.3 Sobel算子

## （3）例程


Sobel边缘检测


Sobel梯度图像


Sobel锐化图像

## 7.2.3 Sobel算子

（4）扩展算子

$$H_1=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}\quad H_2=\begin{bmatrix}0 & -1 & -2\\ 1 & 0 & -1\\ 2 & 1 & 0\end{bmatrix}\quad H_3=\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}$$

$$H_4=\begin{bmatrix}2 & 1 & 0\\ 1 & 0 & -1\\ 0 & -1 & -2\end{bmatrix}\quad H_5=\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}\quad H_6=\begin{bmatrix}0 & 1 & 2\\ -1 & 0 & 1\\ -2 & -1 & 0\end{bmatrix}$$

$$H_7=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}\quad H_8=\begin{bmatrix}-2 & -1 & 0\\ -1 & 0 & 1\\ 0 & 1 & 2\end{bmatrix}\quad g=\max_i H_i f$$

# 7.2.3 Sobel算子

## （4）扩展算子

两个模板梯度图像

八个模板梯度图像

两种算子视觉效果区别不大，扩展算子检测的边缘信息丰富，在需要边缘方向信息的情况下，扩展算子应用更广

# 7.2.4 Prewitt算子



（1）定义

$$S_x = \left|f(x-1,y+1)+f(x,y+1)+f(x+1,y+1)\right| - \left|f(x-1,y-1)+f(x,y-1)+f(x+1,y-1)\right|$$

$$S_y = \left|f(x+1,y-1)+f(x+1,y)+f(x+1,y+1)\right| - \left|f(x-1,y-1)+f(x-1,y)+f(x-1,y+1)\right|$$

$$g = |S_x| + |S_y|$$

$$H_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad H_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

# 7.2.4 Prewitt算子

（2）示例

$$\begin{bmatrix} \boxed{\begin{matrix} 3 & 3 & 3 \\ 3 & 7 & 7 \\ 3 & 7 & 7 \end{matrix}} & \begin{matrix} 3 & 3 \\ 7 & 3 \\ 7 & 3 \end{matrix} \\ \begin{matrix} 3 & 7 & 7 \\ 3 & 3 & 3 \end{matrix} & \begin{matrix} 7 & 3 \\ 3 & 3 \end{matrix} \end{bmatrix}$$

$$g = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 16 & 12 & 16 & 0 \\ 0 & 12 & 0 & 12 & 0 \\ 0 & 16 & 12 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

# 7.2.4 Prewitt算子



## （3）扩展算子

$$H_1 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad H_2 = \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix} \quad H_3 = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$H_4 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix} \quad H_5 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad H_6 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

$$H_7 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad H_8 = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad g = \max_i H_i f$$

# 7.2.4 Prewitt算子

（4）例程

```
Image=im2double(rgb2gray(imread('lotus.jpg')));
H1=[-1 -1 -1;0 0 0;1 1 1];      H2=[0 -1 -1;1 0 -1;1 1 0];
H3=[1 0 -1;1 0 -1;1 0 -1];      H4=[1 1 0;1 0 -1;0 -1 -1];
H5=[1 1 1;0 0 0;-1 -1 -1];      H6=[0 1 1;-1 0 1;-1 -1 0];
H7=[-1 0 1;-1 0 1;-1 0 1];      H8=[-1 -1 0;-1 0 1;0 1 1];
R1=imfilter(Image,H1);          R2=imfilter(Image,H2);
R3=imfilter(Image,H3);          R4=imfilter(Image,H4);
R5=imfilter(Image,H5);          R6=imfilter(Image,H6);
R7=imfilter(Image,H7);          R8=imfilter(Image,H8);
```

## 7.2.4 Prewitt算子



### （4）例程

```
edgeImage1=abs(R1)+abs(R7);
sharpImage1=edgeImage1+Image;
f1=max(max(R1,R2),max(R3,R4));
f2=max(max(R5,R6),max(R7,R8));
edgeImage2=max(f1,f2);
sharpImage2=edgeImage2+Image;
subplot(221),imshow(edgeImage1),title('两个模板梯度图像');
subplot(222),imshow(edgeImage2),title('八个模板梯度图像');
subplot(223),imshow(sharpImage1),title('两个模板锐化图像');
subplot(224),imshow(sharpImage2),title('八个模板锐化图像');
```

# 7.2.4 Prewitt算子

## （4）例程

两个模板梯度

八个模板梯度

八个模板锐化

## 7.3 二阶微分算子

### （1）定义

拉普拉斯算子： $\nabla^2 f = \dfrac{\partial^2 f}{\partial x^2} + \dfrac{\partial^2 f}{\partial y^2}$

$$\frac{\partial^2 f}{\partial x^2} = \Delta_x f(x+1,y) - \Delta_x f(x,y)$$
$$= \left[f(x+1,y) - f(x,y)\right] - \left[f(x,y) - f(x-1,y)\right]$$
$$= f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\frac{\partial^2 f}{\partial y^2} = \Delta_y f(x,y+1) - \Delta_y f(x,y)$$
$$= \left[f(x,y+1) - f(x,y)\right] - \left[f(x,y) - f(x,y-1)\right]$$
$$= f(x,y+1) + f(x,y-1) - 2f(x,y)$$

# 7.3 二阶微分算子

## （1）定义

$$\therefore \nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

$$H_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

或

$$H_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

锐化模板：$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

## （2）示例

$$\begin{bmatrix} 3 & 3 & 3 & 3 & 3 \\ 3 & 7 & 7 & 7 & 3 \\ 3 & 7 & 7 & 7 & 3 \\ 3 & 7 & 7 & 7 & 3 \\ 3 & 3 & 3 & 3 & 3 \end{bmatrix}$$

$$g = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -8 & -4 & -8 & 0 \\ 0 & -4 & 0 & -4 & 0 \\ 0 & -8 & -4 & -8 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

## 7.3 二阶微分算子

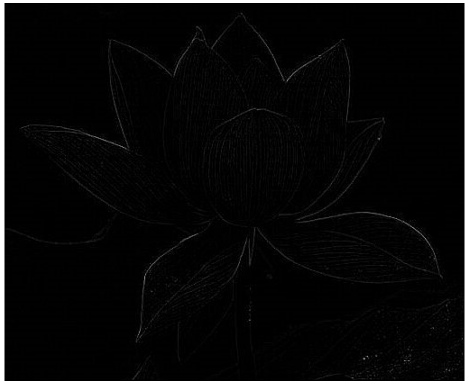
### （3）例程

```
Image=im2double(rgb2gray(imread('lotus.jpg')));
figure,imshow(Image),title('原图像');
H=fspecial('laplacian',0);
R=imfilter(Image,H);
edgeImage=abs(R);
figure,imshow(edgeImage),title('Laplacian梯度图像');
H1=[0 -1 0;-1 5 -1;0 -1 0];
sharpImage=imfilter(Image,H1);
figure,imshow(sharpImage),title('Laplacian锐化图像');
```

# 7.3 二阶微分算子

## （3）例程

原图



滤波图像

锐化图像

# 7.3 二阶微分算子

## （3）例程

原图

滤波图像

锐化图像

# 7.4 高斯滤波与边缘检测

7.4.1 高斯函数

7.4.2 LOG算子

7.4.3 Canny算子

# 7.4.1 高斯函数

## （1）定义

- 高斯函数

$$g(x)=\frac{1}{\sqrt{2\pi\sigma}}e^{\left(-\frac{x^2}{2\sigma^2}\right)}$$

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

- 高斯函数一阶导数

$$\nabla g(x)=\frac{-1}{\sqrt{2\pi\sigma^3}}xe^{\left(\frac{-x^2}{2\sigma^2}\right)}=-\frac{x}{\sigma^2}g(x)$$

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

# 7.4.1 高斯函数

## （1）定义

- 高斯函数二阶导数

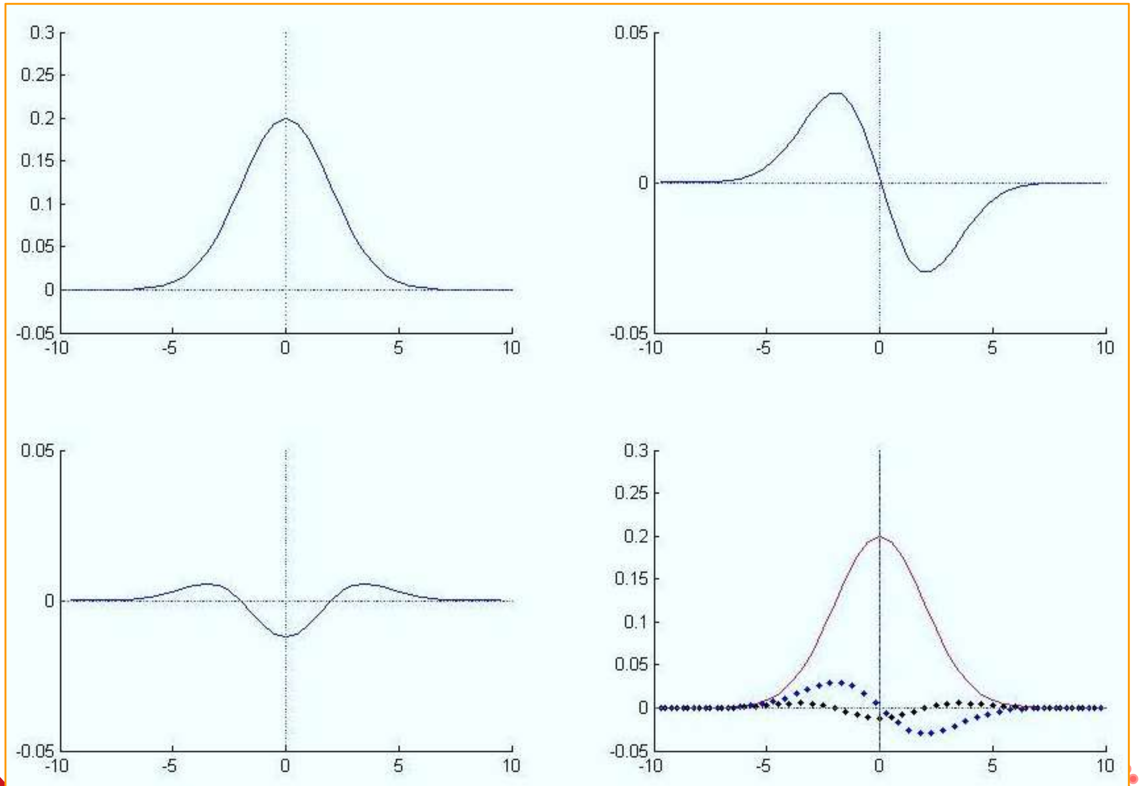
$$\nabla^2 g(x)=\left(\frac{x^2}{\sqrt{2\pi}\sigma^5}-\frac{1}{\sqrt{2\pi}\sigma^3}\right)e^{\left(\frac{-x^2}{2\sigma^2}\right)}=\left(\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2}\right)g(x)$$

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

# 7.4.1 高斯函数



（2）特点



# 7.4.1 高斯函数

## （2）特点

- 随着远离原点，权值逐渐减小到零，表明离中心较近的图像值比远处的更重要；标准差$\sigma$决定邻域范围，总权值的95%包含在$2\sigma$的中间范围内
- 二阶导数具有光滑的中间突出部分，函数值为负；两个光滑的侧边突出部分，值为正。零交叉位于$-\sigma$和$+\sigma$处，与$g(x)$的拐点和$g'(x)$的极值点对应
- 1D形式绕垂直轴旋转得各向同性的2D函数形式（在任意过原点的切面上具有相同的1D高斯截面），其二阶导数形式好像一个宽边帽或称为墨西哥草帽
- 从数学推导上，帽子的空腔口沿$z=g(x,y)$轴向上，在显示和滤波应用中空腔口一般朝下，即中间突起的部分为正，帽边为负。

# 7.4.2 LOG算子



## （1）定义

- **Marr**用高斯函数先对图像作平滑，然后用拉普拉斯算子检测边缘，简称**LOG**滤波器

二元高斯函数：$$g(x,y)=\frac{1}{2\pi\sigma^2}e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}$$

将$g$与图像函数$f$卷积，得到一个平滑的图像函数，对该函数做拉普拉斯运算，提取边缘

## 7.4.2 LOG算子

### （1）定义

可以证明 $\nabla^2\left[f(x,y)*g(x,y)\right]=f(x,y)*\nabla^2 g(x,y)$

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

$\nabla^2 g(x,y)$为LOG滤波器，也称为Marr-Hildrech算子

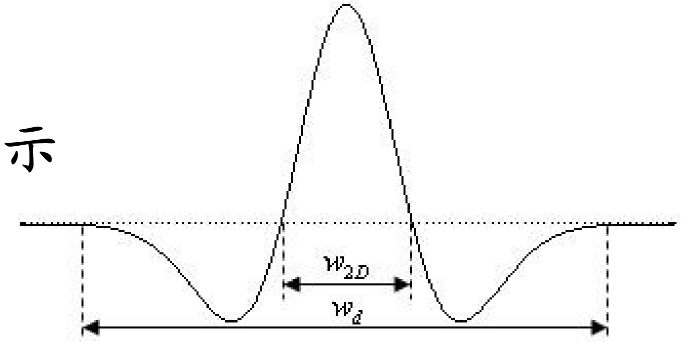
$\sigma$称为尺度因子，大的值可用来检测模糊的边缘，小的值可用来检测聚焦良好的图像细节。

# 7.4.2 LOG算子

## （1）定义

- **LOG算子的形状如图所示，常称为墨西哥草帽**

$$w_{2D} = 2\sigma$$




$\nabla^2 g$的横截面

滤波器的大小由$\sigma$的数值或等价地由$w_{2D}$的数值来确定。为了不使函数被过分地截短，应在足够大的窗口内作计算，窗口宽度通常取为：$w_d \geq 3.6w_{2D}$

# 7.4.2 LOG算子

## （1）定义

- 模板形式

$\sigma$取不同的值，对应不同大小的模板

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

## 7.4.2 LOG算子

（1）定义

$$
\begin{bmatrix}
0 & 0 & 0 & -1 & -1 & -2 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & -2 & -4 & -8 & -9 & -8 & -4 & -2 & 0 & 0 \\
0 & -2 & -7 & -15 & -22 & -23 & -22 & 15 & -7 & -2 & 0 \\
-1 & -4 & -15 & -24 & -14 & -1 & -14 & -24 & -15 & -4 & -1 \\
-1 & -8 & -22 & -14 & 52 & 103 & 52 & -14 & -22 & -8 & -1 \\
-2 & -9 & -23 & -1 & 103 & 178 & 103 & -1 & -23 & -9 & -2 \\
-1 & -8 & -22 & -14 & 52 & 103 & 52 & -14 & -22 & -8 & -1 \\
-1 & -4 & -15 & -24 & -14 & -1 & -14 & -24 & -15 & -4 & -1 \\
0 & -2 & -7 & -15 & -22 & -23 & -22 & 15 & -7 & -2 & 0 \\
0 & 0 & -2 & -4 & -8 & -9 & -8 & -4 & -2 & 0 & 0 \\
0 & 0 & 0 & -1 & -1 & -2 & -1 & -1 & 0 & 0 & 0
\end{bmatrix}
$$

**11×11的LOG近似模板，$\sigma^2=2$**

# 7.4.2 LOG算子



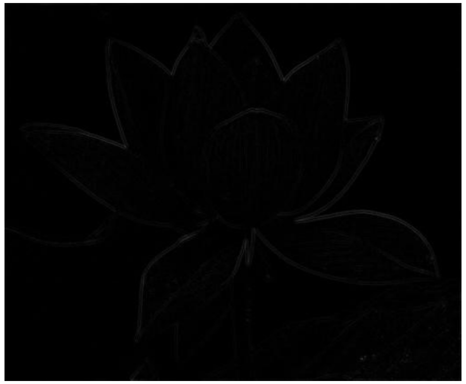
（2）例程

```
Image=im2double(rgb2gray(imread('lotus.jpg')));
figure,imshow(Image),title('原图像');
BW= edge(Image,'log');
figure,imshow(BW),title('Log边缘检测');
H=fspecial('log',7,1);
R=imfilter(Image,H);
edgeImage=abs(R);
figure,imshow(edgeImage),title('Log滤波图像');
sharpImage=Image+edgeImage;
figure,imshow(sharpImage),title('Log锐化图像');
```

# 7.4.2 LOG算子

（2）例程

边缘检测



滤波图像

锐化图像

# 7.4.2 LOG算子

（2）例程

## 7.4.3 Canny算子



### （1）最优边缘检测

- John F. Canny于1986年开发出来的一个多级边缘检测算法，被很多人认为是边缘检测的最优算法
- 最优边缘检测的三个主要评价标准
  - 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。
  - 对边缘的定位准确。标识出的边缘要与图像中的实际边缘尽可能接近。
  - 最小响应。图像中的边缘最好只标识一次，并且可能存在的图像噪声部分不应标识为边缘。

# 7.4.3 Canny算子

## （2）步骤

- 使用高斯平滑滤波器卷积降噪
- 计算平滑图像的梯度幅值和方向，可采用不同的梯度算子
- 对梯度幅值应用非极大抑制，即找出图像梯度中的局部极大值点，其他非局部极大值点置零
- 使用双阈值检测和连接边缘
  - 高阈值用来找到每一条线段
  - 低阈值用来确定线段上的点

## 7.4.3 Canny算子

（3）例程

```
Image=im2double(rgb2gray(imread('lotus.jpg')));
figure,imshow(Image),title('原图像');
BW= edge(Image,'canny');
figure,imshow(BW),title('Canny边缘检测');
```

原图

边缘检测

# 7.5 频域高通滤波

原理：

$$f(x,y) \xrightarrow{DFT} F(u,v) \xrightarrow{H(u,v)} G(u,v) \xrightarrow{IDFT} g(x,y)$$

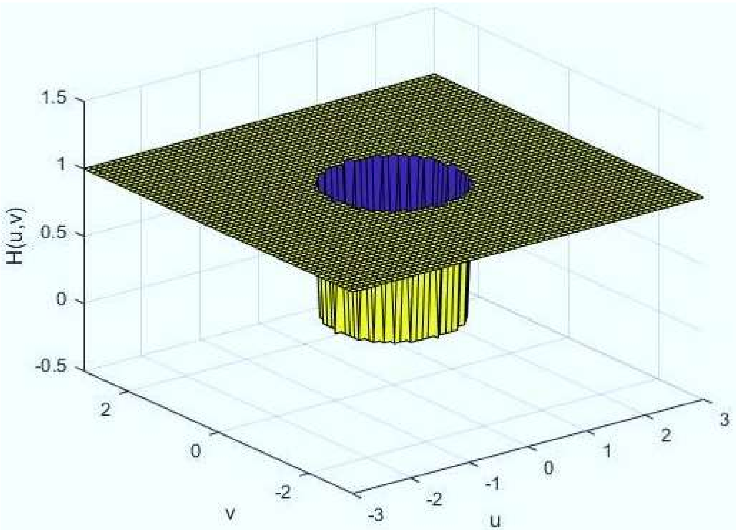
关键技术：

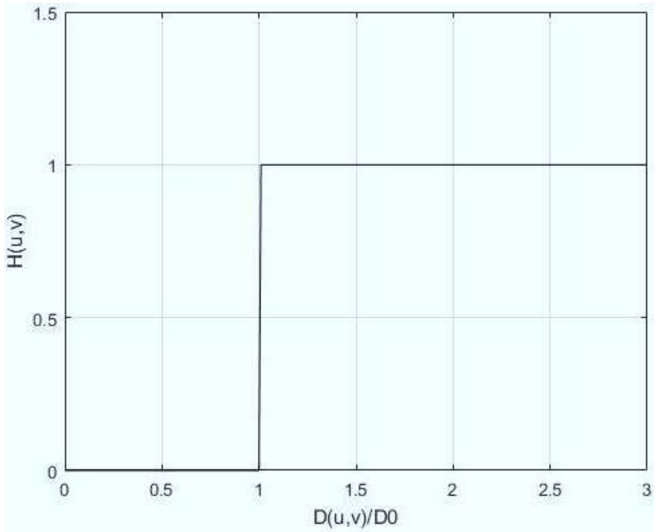
高通滤波器的设计

# 7.5 频域高通滤波

## （1）理想高通滤波器

$$H(u,v)=\begin{cases}0 & D(u,v)\le D_0\\ 1 & D(u,v)>D_0\end{cases}$$

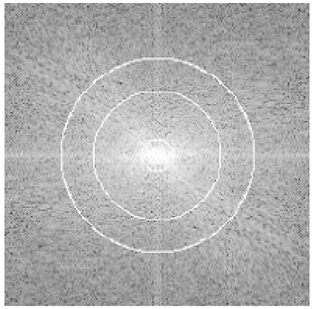


# 7.5 频域高通滤波

## （1）理想高通滤波器

lena图像



傅里叶频谱图

$D_0=5$

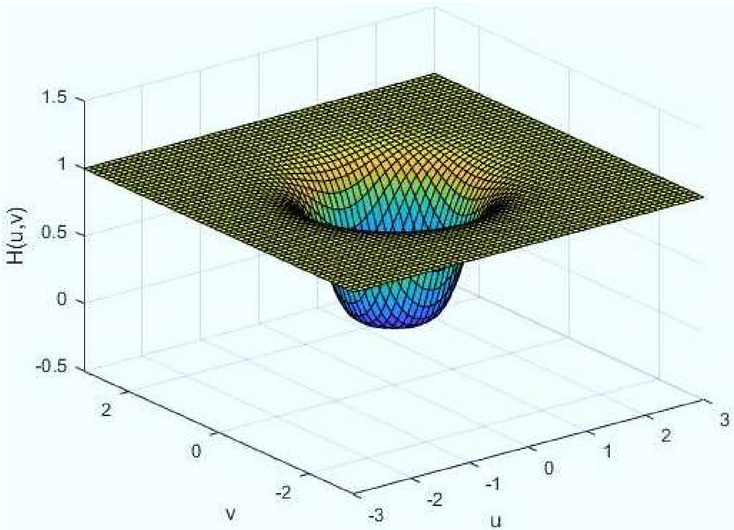
$D_0=11$

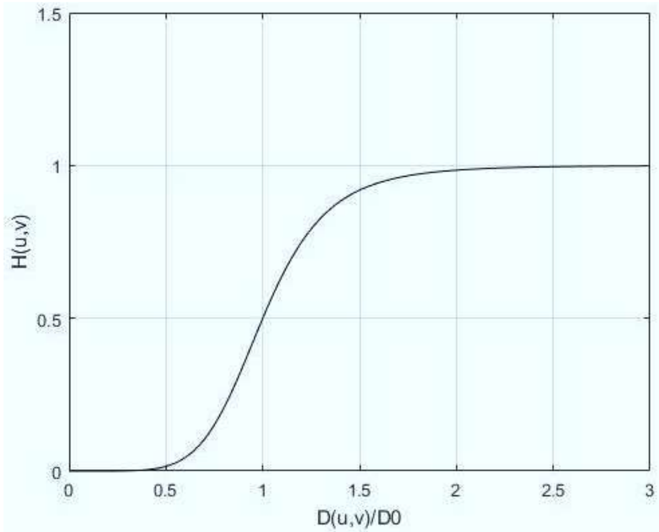
$D_0=45$

$D_0=68$

## （2）巴特沃斯高通滤波器

$$H(u,v)=\frac{1}{1+\left[D_0/D(u,v)\right]^{2n}}$$



# 7.5 频域高通滤波

## （2）巴特沃斯高通滤波器

原图

高通滤波

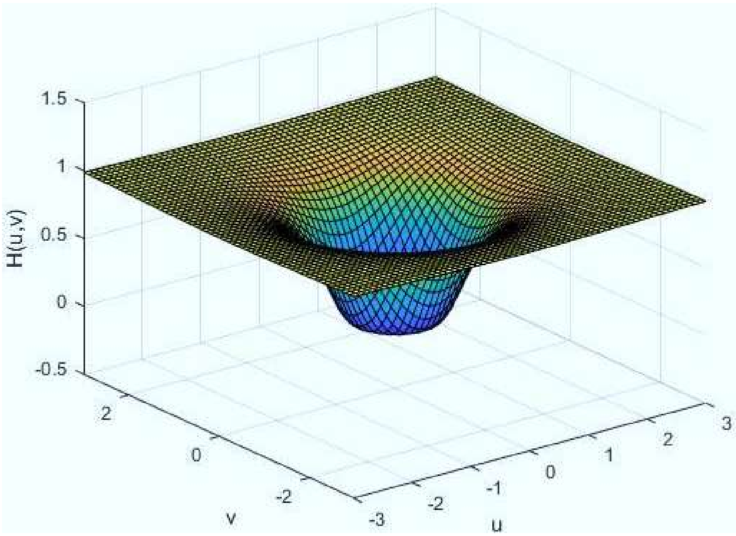
锐化

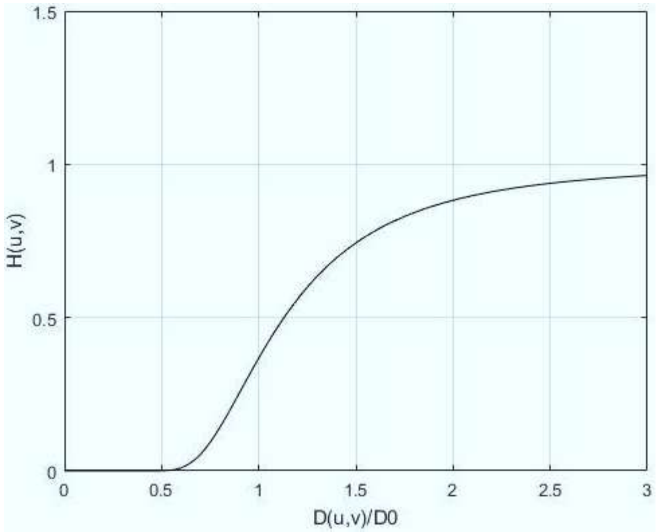
$$\mathbf{D_0=35;\ \mathit{n}=3;}$$

# 7.5 频域高通滤波

## （3）指数高通滤波器

$$H(u,v)=exp\left\{-\left[\frac{D_0}{D(u,v)}\right]^n\right\}$$



## 7.5 频域高通滤波

### （3）指数高通滤波器


原图

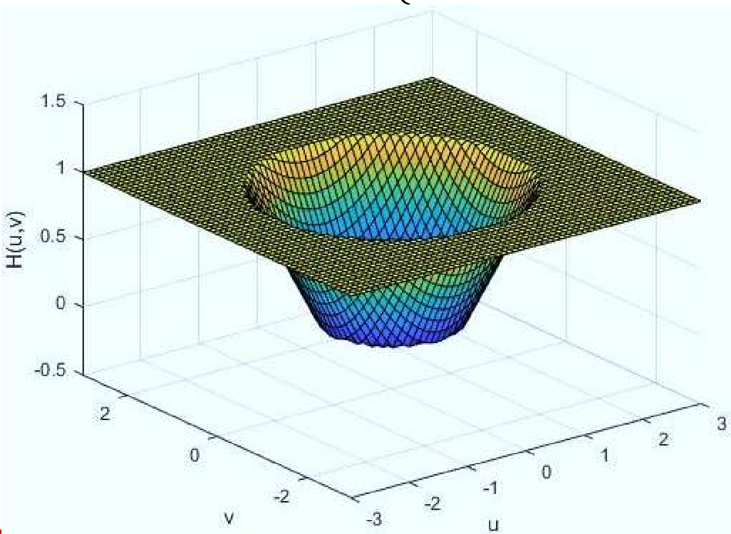

高通滤波

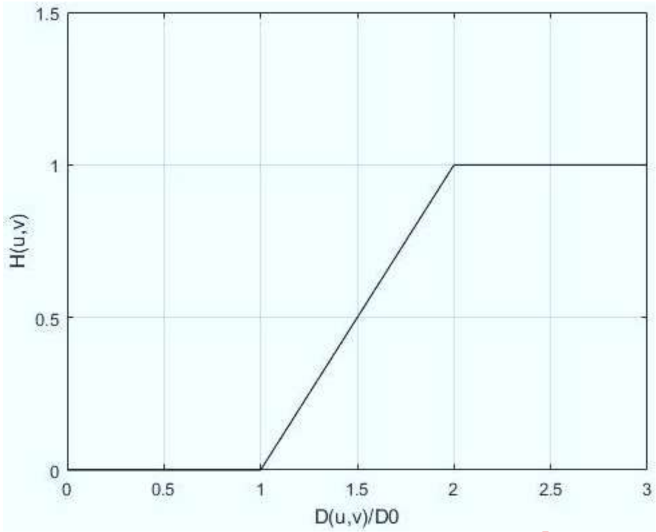

锐化

**$D_0=35$; $n=3$;**

（4）梯形高通滤波器

$$H(u,v)=\begin{cases}0 & D(u,v)<D_0\\ \dfrac{1}{D_1-D_0}\left[D(u,v)-D_0\right] & D_0\le D(u,v)\le D_1\\ 1 & D(u,v)>D_1\end{cases}$$



（4）梯形高通滤波器

原图

高通滤波

锐化

**$D_0$=35; $D_1$=75; $n$=3;**

# 7.6 基于小波变换的边缘检测

## （1）原理

- 对图像进行小波变换，将高频信息逆变换，实现边缘检测。如第4章的例子，检测效果有限。
- 基于小波变换模极大值的边缘检测方法
  - 先平滑图像，再检测边缘
  - 卷积运算的微分性质：

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

  - 平滑函数满足一定要求时，其一阶偏导数为小波函数，因此，对图像进行小波变换，再检测局部极值，确定边缘

# 7.6 基于小波变换的边缘检测

## （2）步骤

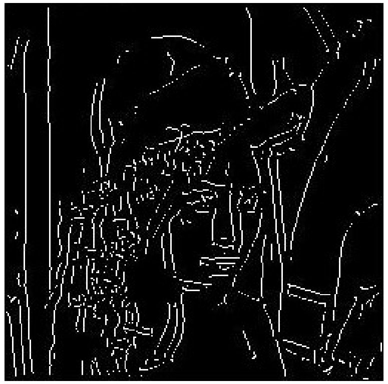
- 选取平滑函数，求一阶偏导确定小波函数
- 进行小波变换
- 计算图像的二进小波变换矢量模值和相角
- 寻找梯度方向上取极大值的点
- 去除噪声点，确定边缘图像

## 7.6 基于小波变换的边缘检测

### （3）例程

原图

小波变换模图像



模极大提取的边缘图像