西安交通大学

数字图像处理作业报告

作业 4：空域滤波与边缘检测

摘要：本次报告首先简要介绍了空域滤波与边缘检测的基本概念及原理，然后展示了通过 Matlab 编程实现以下功能的具体过程：（1）利用固定方差 sigma=1.5 产生高斯滤波器；（2）用不同模板大小的中值滤波器与高斯滤波器平滑测试图像；（3）用不同的高通滤波器滤波测试图像，并分析其优缺点；

关键词：高斯滤波器，中值滤波器，高通滤波器，MATLAB

姓　名　：胡　　欣　　盈

班　级　：自　动　化　9 4

学　号　：2 1 9 4 3 2 3 1 7 6

提交日期：2022 年 3 月 13 日

目录

一、基本概念及原理

（1）掩模 Mask

掩模是数字图像滤波中最基本的操作，就是用一个窗口在图像上进行滑动，并对窗口内的像素点进行各种运算，这个窗口就是掩模（mask）。常用的掩模是 d×d 的正方形（为保证有中心点，d 一般为奇数）。

（2）中值滤波 Median Filtering

中值滤波是一种非线性低通滤波。所谓中值滤波，就是对于每个像素点，找到其掩模覆盖范围内像素点灰度值的中位数，作为这一点的灰度值。这样处理可以使得图像更加平滑，且效果较为良好，是实际中广泛使用的一种平滑滤波方式。

（3）高斯滤波 Gaussian Filtering

高斯滤波是一种线性低通滤波，相当于在每个像素点的掩模中，按照二维高斯分布的方式增加了权重，将掩模中的点按照权重相加即得到中心点的值。

二维高斯概率密度函数如下

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

将掩模中心点坐标设为（0,0），其他点坐标分别在此基础上加减即可得到，可以看出掩模中所有点坐标相加为 0

由此得到掩模中每个位置的权重：

$$w(i,j)=\frac{1}{2\pi\sigma^2}e^{-\frac{(i-d-1)^2+(j-d-1)^2}{2\sigma^2}}$$

其中 d 是掩模半径（掩模大小为(2d+1) × (2d+1)，由此就可以得到高斯滤波模板 W

根据概率密度的性质，这样计算得到的权重之和为 1，恰好能使得对图像的灰度处理不超出限度且变换后的总体灰度等级没有改变。

数字滤波中常使用二维卷积来实现对每个点进行模板中加权去和的操作，此处即：A'=A*W，A、A′ 分别是滤波前和滤波后的图像灰度矩阵。

（4）反锐化掩模 Unsharp Masking

反锐化掩模是先将图像进行低通滤波（常用高斯滤波），然后将原图像与之作差得到保留高频成分的图像，再将这个图像与原图像相加，得到增强边缘的图像（因为图像边缘往往灰度值变化率较快，即频率较大）。反锐化掩模可以起到提高图像高频成分，增强图像边缘轮廓的作用。

其中，得到的保留高频成分的图像有时常乘以一个系数 k 再与原图像相加，当 k=1 时就是反锐化掩模，当 k＞1 时，称为高提升滤波（high boost filtering）。

（5）Sobel 边缘检测

通过 Sobel 算子与原图像卷积即可。通过这样的操作可以得到图像的边缘，且这种方法得到的边缘往往较为粗大。Sobel 算子用于得到图像的梯度信息，则这样处理后梯度较大的位置更亮，而梯度较大的位置往往是图像的边缘位置，故而通过这种方法可以实现图像的边缘检测和提取。

（6）Laplacian 边缘检测

一般来说，图像的边缘轮廓不仅具有较高的梯度，也具有较高的二阶导数。对一个二维函数求二阶导的变换就是 Laplacian 变换，所以我们容易想到通过构造一个 Laplace 算子来实现图像的边缘检测，即 Laplacian 边缘检测。

（7）Canny 算法

Canny 算法是对 Sobel 算法的更进一步操作。首先用高斯滤波降低噪音对图像的影响，再用 Sobel 算子计算出图像的梯度，包括幅值和方向。对每个像素点，判断其是否是其梯度方向上的极大值，如果不是则抑制（置 0）。最后用双阈值进行检测和连接，即大于高阈值的确定保留，小于低阈值的一定抑制，介于两者之间的像素点，如果它的 8-邻域中存在确定保留点，则保留，否则抑制。

其中判断梯度方向极大值时，需要使用插值法构建虚拟像素点，称为亚像素点。使用 Canny 算法得到的图像边缘很细，且对噪音的滤去效果较好。

二、空域滤波

（1）高斯滤波器的实现

本题要求利用固定方差 sigma=1.5 产生高斯滤波器，根据前述原理与公式，我们可以构造函数 GaussianFilter (Img, masksize, sigma)，以掩模中心点为原点，得到掩模内各点的坐标，再计算出每个点的高斯权重，得到高斯模板；

（2）平滑测试图像

本题要求分别用高斯滤波器和中值滤波器去平滑测试 test1.pgm 和 test2.tif 文件，模板大小分别选取 3x3 ， 5x5 ，7x7，并分析各其优缺点。

1、高斯滤波

由于已经构造出 GaussianFilter 得到高斯模板，我们只需将原图与高斯模板进行二维卷积得到处理后的图像就是所需的平滑图像。结果如图 2-1~2-2 所示。

2、中值滤波

实现中值滤波只需在设置掩模大小后使用 padarray()函数扩展图像，各向扩展大小等于掩模半径，对每个像素点，在掩模范围内找灰度值的中位数赋给它即可，结果如图 2-3~2-4 所示。

3、分析

通过对比分析，可以发现中值滤波的优点是计算简单迅速，能够有效的去除图像中的椒盐噪声；缺点则是对于一些高斯型噪声不能很好的滤除。

而高斯滤波的优点则在于对高斯型噪声有非常好的降噪效果，缺点则是计算相对复杂。同时，高斯滤波相当于对整幅图像进行了模糊化处理，这一点有利有弊，视情况而定。

对于这几幅图来看，采用中值滤波的效果略优于高斯滤波的效果。这两种滤波效果都与掩模大小密切相关，掩模越大，滤波效果越好，而且图像更加明亮

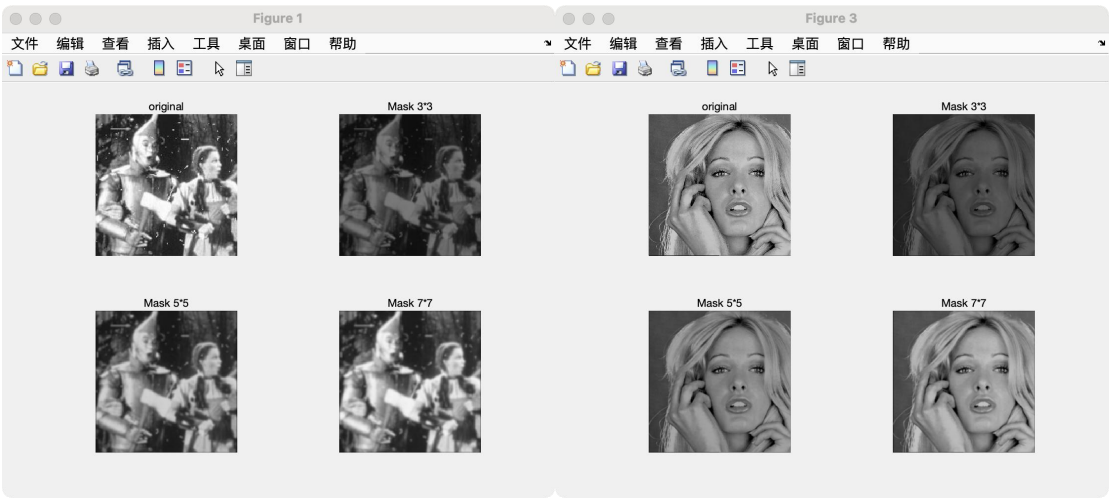

图 2-1~2-2 不同模板大小高斯滤波后的 test1、2 图像

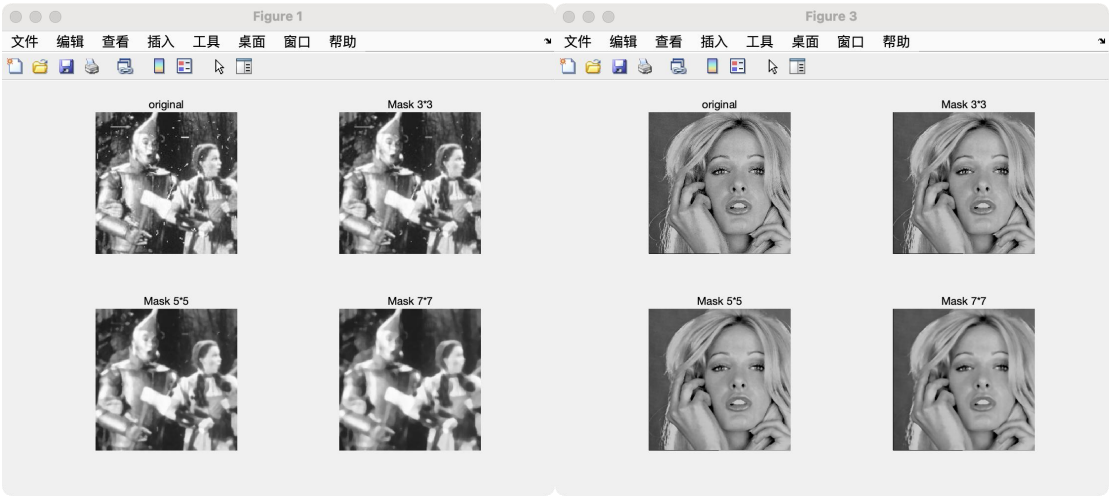

图 2-3~2-4 不同模板大小中值滤波后的 test1、2 图像

三、边缘检测

本题要求利用高通滤波器对 test3_corrupt.pgm 和 test4.tif 两个文件进行处理，需要使用 unsharp masking, Sobel edge detector, Laplace edge detection; Canny algorithm 分别处理，并分析其优缺点。具体处理过程如下：

（1）unsharp masking

如原理部分所述，反锐化掩模、高提升滤波算法可以使得图像的边缘轮廓和高频成分增强，其效果与 k 取值密切相关，k 越大对细节

的增强就越明显。由此，我们可以进行如下操作。

首先通过高斯滤波得到模糊化图像 Imgblur，用原图像 Img-Imgblur 可以得到反锐化掩模 UnsharpMask，再通过 UnsharpMask + Img 得到反锐化掩模处理后的图像，最后令 k*UnsharpMask + Img 可以得到高提升滤波后的图像（这里不妨选 k=3），结果如图 3-1~3-2 所示。

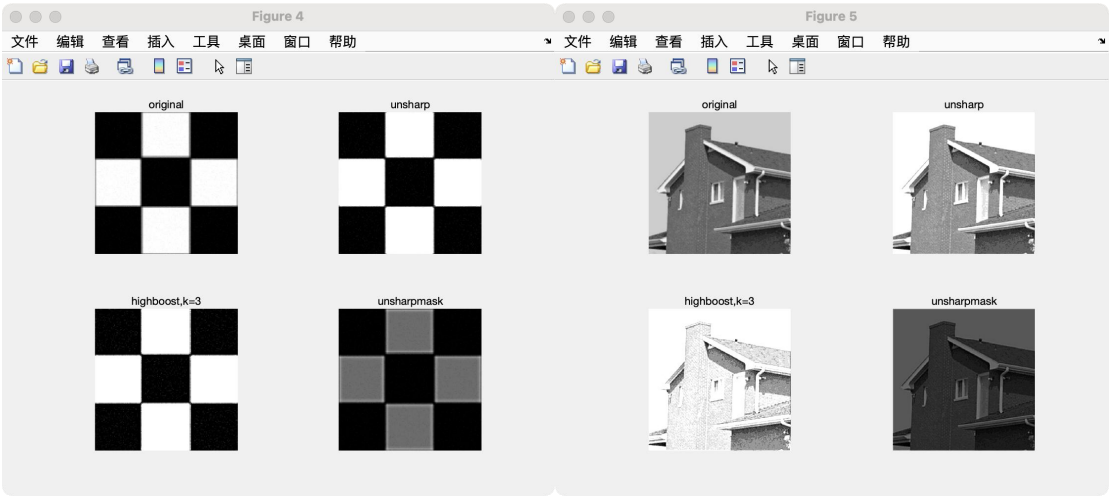

图 3-1~3-2 unsharp masking 处理后的 test3、4 图像

（2）Sobel edge detector

首先构造 x，y 方向的 Sobel 算子 Sx 和 Sy，再分别用原图与 Sx，Sy 进行二维卷积，得到两个方向的梯度矩阵，然后可以将两个方向梯度分别求绝对值并相加得到图像的梯度矩阵，最后将梯度矩阵转成 uint8 格式，即为处理后图像如图 3-3 所示。

由图可知，采用 Sobel 算法提取出的边缘范围较大，所以显示的边缘较为粗大，看起来更加清晰明了，且对噪声有不错的抑制效果，但是精确度不高。

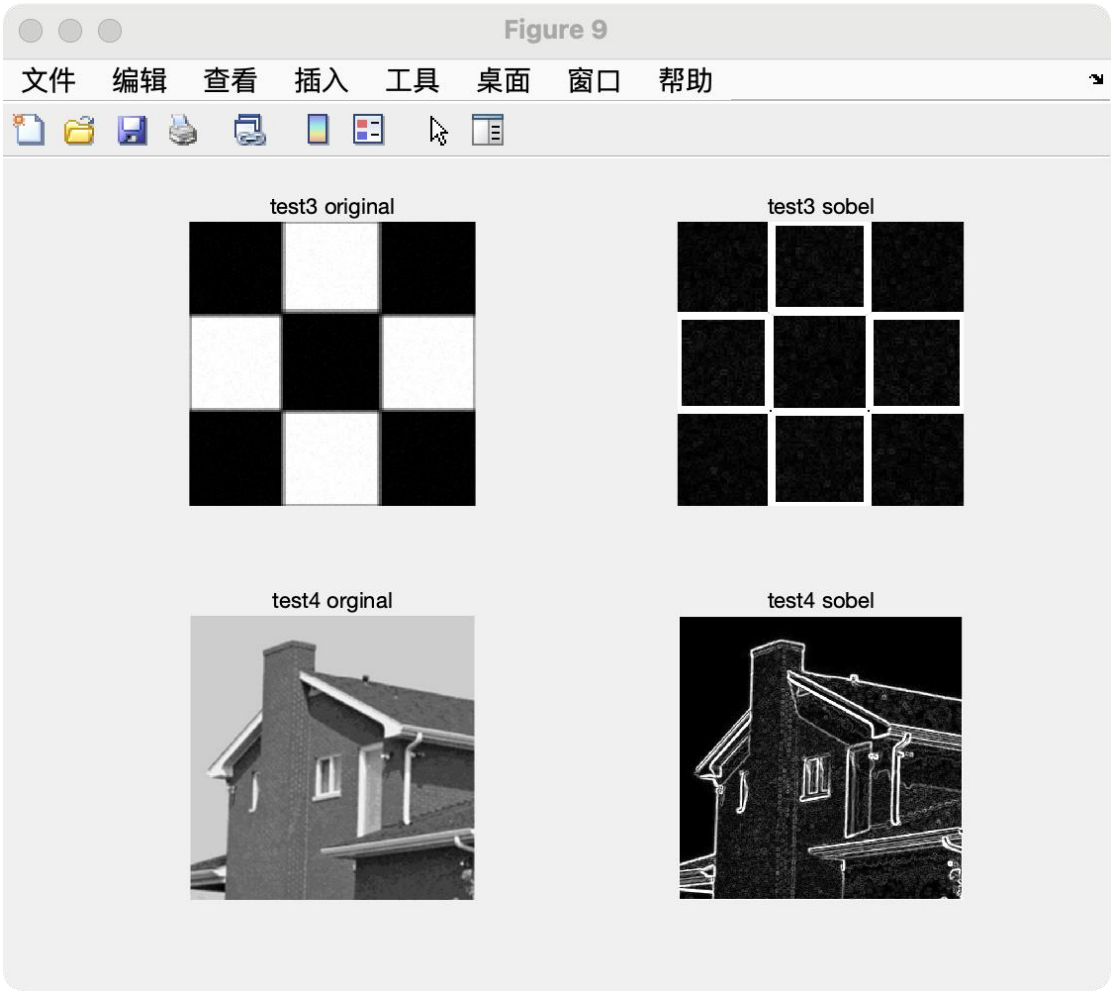

图 3-3 Sobel edge detector 处理后的 test3、4 图像

（3）Laplace edge detection

由原理可知，只需构造出扩展 Laplace 算子，再将原图与 Laplace 算子进行二维卷积，结果转为 uint8 格式即为处理后图像，如图 3-4 所示。

采用 Laplacian 边缘检测算法，可以看出当图像比较“干净”时有较好的边缘检测作用，且精度明显优于 Sobel 算法。但是 Laplacian 对噪声过于敏感，很容易受到噪音干扰，可以看到 test3 的检测结果仍然有很多椒盐噪声。

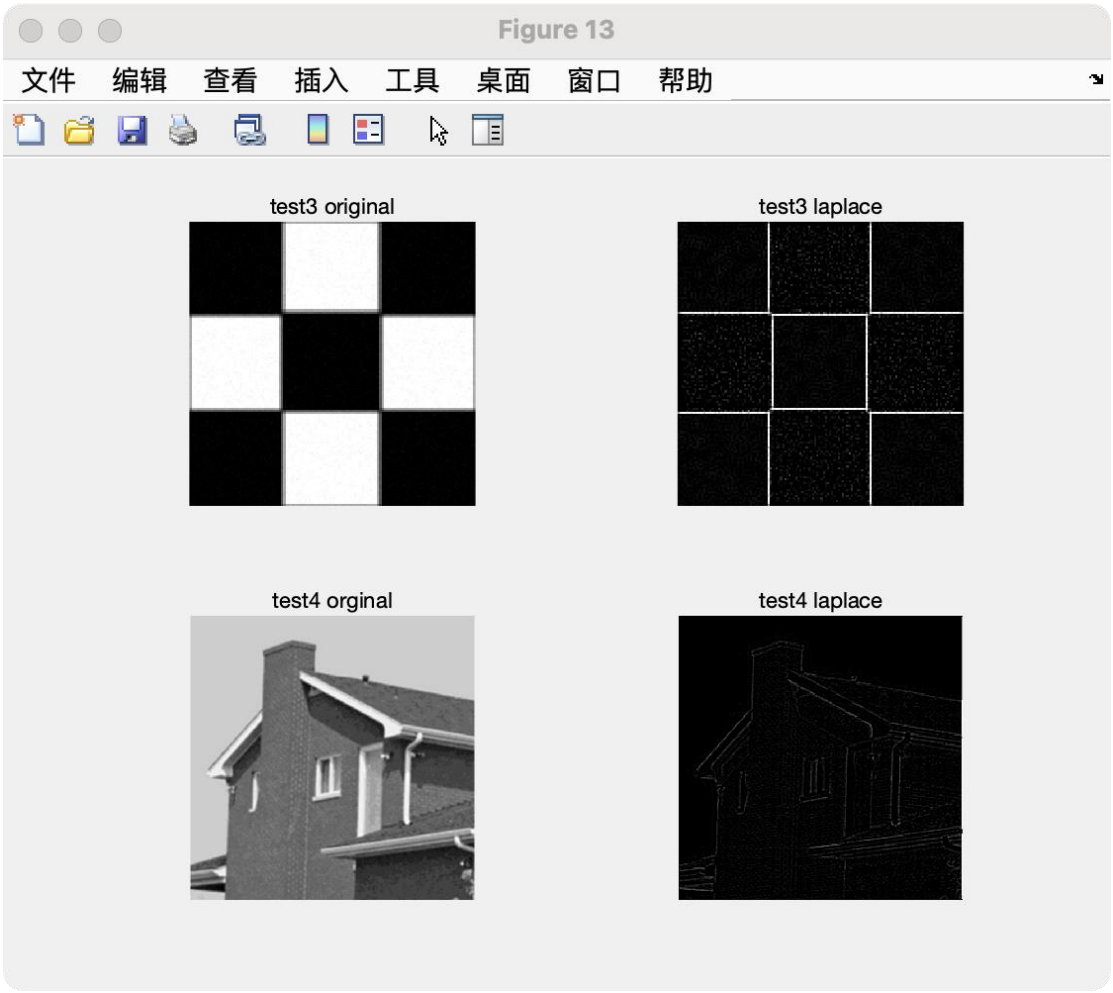

图 3-4 Laplace edge detection 处理后的 test3、4 图像

（4）Canny algorithm

根据原理，首先用高斯滤波将图像模糊化处理以降低噪音影响，再用 Sobel 算子得到 x，y 方向的梯度幅值和方向，对梯度方向非极大值进行抑制，然后将梯度方向从 0° 开始，每 45° 化为一个区域，共四个区域（正负视为同一方向）。

其次，对像素点所在区域找到区域两侧的正像素点，根据梯度方向进行加权平均，就可以插值得到两个亚像素点 g1，g2，分别处于梯度方向的两侧，再判断像素点是否大于 g1，g2，即判断在梯度方向上是否为极大值，若不是则置 0，并设置高低阈值进行双阈值检测，

大于高阈值 T2 的值一定保留，小于低阈值 T1 的一定置 0，介于中间的点，如果在其 8-邻域内有确定保留的点，则它保留，否则置 0。最后，考虑到 Canny 得到的边缘往往较细，可以对整幅图进行强化处理得到最终处理后的图像，如图 3-5 所示。

可以发现，Canny 算法几乎不受噪音干扰，且得到的边缘精度很高，但是算法较为复杂费时，并且处理效果与高斯滤波的两个参数（掩模大小、标准差 sigma），以及自身的两个参数（高低阈值）的选取密切相关，这几个参数的选取也根据图像有所不同。在本题中，为将两幅图一起处理输出，综合比较后选取高斯掩模大小为 3，sigma 为 2，高阈值为 30，低阈值为 5，得到的效果较为良好。

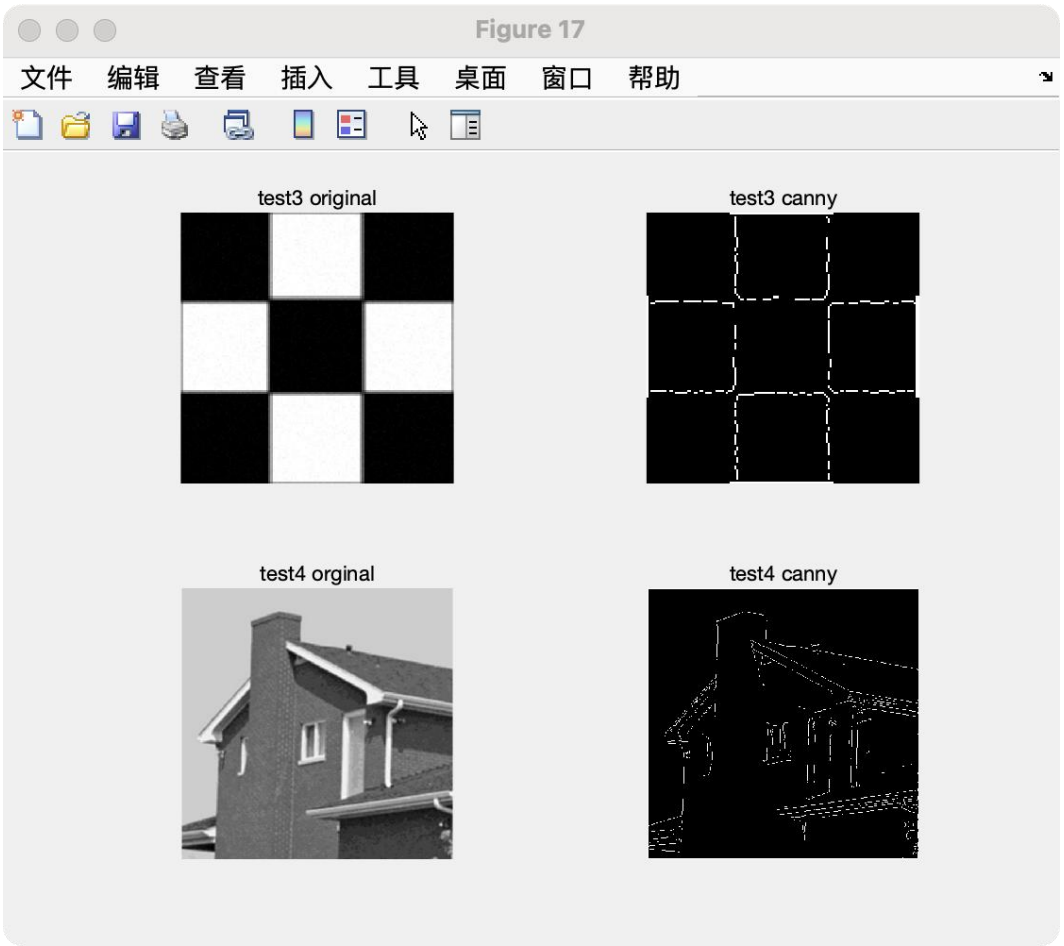

图 3-5 Canny algorithm 处理后的 test3、4 图像

附录

附录 1：参考文献

[1]阮秋琦,阮宇智等.数字图像处理(第三版)(美)冈萨雷斯[M],2003,电子工业出版社

[2]门敬文•数字图像处理 MATLAB 版[M].2007.2,国防工业出版社.

[3]数字图像处理——虚宇宸轩-CSDN 博客

附录 2：源代码

```
clear;
clc;

test1=imread('Users/macbookpro/Desktop/test1.pgm');
test2=imread('Users/macbookpro/Desktop/test2.tif');

test1Mask3=GaussianFilter(test1,3,1.5);
test1Mask5=GaussianFilter(test1,5,1.5);
test1Mask7=GaussianFilter(test1,7,1.5);

figure
subplot(2,2,1);imshow(test1);title('original');
subplot(2,2,2);imshow(test1Mask3);title('Mask 3*3');
subplot(2,2,3);imshow(test1Mask5);title('Mask 5*5');
subplot(2,2,4);imshow(test1Mask7);title('Mask 7*7');

test2Mask3=GaussianFilter(test2,3,1.5);
test2Mask5=GaussianFilter(test2,5,1.5);
test2Mask7=GaussianFilter(test2,7,1.5);

figure
subplot(2,2,1);imshow(test2);title('original');
subplot(2,2,2);imshow(test2Mask3);title('Mask 3*3');
subplot(2,2,3);imshow(test2Mask5);title('Mask 5*5');
subplot(2,2,4);imshow(test2Mask7);title('Mask 7*7');

function Img_out=MedianFilter(Img,masksize)
exsize=floor(masksize/2);   %各方向扩展大小
Imgex=padarray(Img,[exsize,exsize],'replicate','both'); %扩展图片
[m,n]=size(Img);
Img_out=Img;    %将 Img_out 准备为和 Img 相同的 size
for i=1:m
    for j=1:n
        neighbor=Imgex(i:i+masksize-1,j:j+masksize-1);  %截取邻域
        Img_out(i,j)=median(neighbor(:));   %中值滤波
    end
end
end

function Img_out=GaussianFilter(Img,masksize,sigma)
for i=1:masksize
```

```
    for j=1:masksize
        x=i-ceil(masksize/2);
        y=j-ceil(masksize/2);
        h(i,j)=exp(-(x^2+y^2)/(2*sigma^2))/(2*pi*sigma^2);
    end
end
Img_out=uint8(conv2(Img,h,'same'));
end


```

```
clear;
clc;

test3=imread('Users/macbookpro/Desktop/test3_corrupt.pgm');
test4=imread('Users/macbookpro/Desktop/test4.tif');

[test3_unsharp,test3_highboost,test3_unsharpmask]=Unsharp(test3);
[test4_unsharp,test4_highboost,test4_unsharpmask]=Unsharp(test4(:,:));

figure
subplot(2,2,1);imshow(test3);title('original');
subplot(2,2,2);imshow(test3_unsharp);title('unsharp');
subplot(2,2,3);imshow(test3_highboost);title('highboost,k=3');
subplot(2,2,4);imshow(test3_unsharpmask);title('unsharpmask');

figure
subplot(2,2,1);imshow(test4(:,1:512));title('original');
subplot(2,2,2);imshow(test4_unsharp(1:512,1:512));title('unsharp');
subplot(2,2,3);imshow(test4_highboost(1:512,1:512));title('highboost,k=3');
subplot(2,2,4);imshow(test4_unsharpmask(1:512,1:512));title('unsharpmask');

test3_sobel=Sobel(test3);
test4_sobel=Sobel(test4(:,:));

figure
subplot(2,2,1);imshow(test3);title('test3 original');
subplot(2,2,2);imshow(test3_sobel);title('test3 sobel');
subplot(2,2,3);imshow(test4(1:512,1:512));title('test4 orginal');
subplot(2,2,4);imshow(test4_sobel(1:512,1:512));title('test4 sobel');

test3_laplace=Laplace(test3);
test4_laplace=Laplace(test4(:,:));
```

```
33.
34.  figure
35.  subplot(2,2,1);imshow(test3);title('test3 original');
36.  subplot(2,2,2);imshow(test3_laplace);title('test3 laplace');
37.  subplot(2,2,3);imshow(test4(1:512,1:512));title('test4 orginal');
38.  subplot(2,2,4);imshow(test4_laplace(1:512,1:512));title('test4 laplace');
39.
40.  test3_canny=Canny(test3);
41.  test4_canny=Canny(test4(:,:));
42.
43.  figure
44.  subplot(2,2,1);imshow(test3);title('test3 original');
45.  subplot(2,2,2);imshow(test3_canny);title('test3 canny');
46.  subplot(2,2,3);imshow(test4(1:512,1:512));title('test4 orginal');
47.  subplot(2,2,4);imshow(test4_canny(1:512,1:512));title('test4 canny');
48.  function [Img_unsharp,Img_highboost,mask_unsharp]=Unsharp(Img)
49.
50.  Imgblur=GaussianFilter(Img,7,3);     %用高斯滤波得到虚化图像
51.  mask_unsharp=Img-Imgblur;    %得到反锐化掩模
52.  Img_unsharp=Img+mask_unsharp;    %得到反锐化掩模处理后的图像
53.  Img_highboost=Img+3*mask_unsharp;    %得到 highboost 滤波后的图像（k=3）
54.  end
55.
56.  function Img_out=GaussianFilter(Img,masksize,sigma)
57.  for i=1:masksize
58.      for j=1:masksize
59.          x=i-ceil(masksize/2);
60.          y=j-ceil(masksize/2);
61.          h(i,j)=exp(-(x^2+y^2)/(2*sigma^2))/(2*pi*sigma^2);
62.      end
63.  end
64.  Img_out=uint8(conv2(Img,h,'same'));
65.  end
66.
67.  function [Img_out,gx,gy]=Sobel(Img)
68.  sobely=[1,2,1;0,0,0;-1,-2,-1];  %构造 y 方向 Sobel 算子
69.  sobelx=[1,0,-1;2,0,-2;1,0,-1];  %构造 x 方向 Sobel 算子
70.  gx=conv2(Img,sobelx,'same');
71.  gy=conv2(Img,sobely,'same');
72.  Img_out=uint8(abs(gx)+abs(gy));
73.  end
74.
75.  function Img_out=Laplace(Img)
76.  laplace=[1,1,1;1,-8,1;1,1,1];   %构造拉普拉斯算子
```

```
Img_out=uint8(conv2(Img,laplace,'same'));
end

function Img_out=Canny(Img)
Imgblur=GaussianFilter(Img,3,2);     %用高斯滤波对图像进行模糊处理以降低噪声影响
[~,gx,gy]=Sobel(Imgblur);    %用 Sobel 得到 x，y 方向的梯度
g=uint8(sqrt(double(gx.*gx)+double(gy.*gy)));    %得到图像的梯度幅值矩阵
theta=atan2(double(gy),double(gx)); %得到图像的梯度方向矩阵
gcom=g; %准备一个存储空间 gcom
g=double(g);     %将 g 化为 double 型，为下面运算准备
%下面将按照方向分别判断:
for i=2:size(g,1)-1
    for j=2:size(g,2)-2
        if ((theta(i,j)>=0)&(theta(i,j)<pi/4))|((theta(i,j)>=-pi)&(theta(i,j)<-3*pi/4)
)
            g1=g(i-1,j+1)*(gy(i,j)/gx(i,j))+g(i,j+1)*(1-gy(i,j)/gx(i,j));    %用插值得到梯度方向上的一个亚像素点，下同
            g2=g(i+1,j-1)*(gy(i,j)/gx(i,j))+g(i,j-1)*(1-gy(i,j)/gx(i,j));    %用插值得到梯度方向上的另一个亚像素点，下同
            if (gcom(i,j)<=g1)|(gcom(i,j)<=g2)
                gcom(i,j)=0;     %若在梯度方向上不是极大值则置 0，下同
            end
        end
        if ((theta(i,j)>=pi/4)&(theta(i,j)<pi/2))|((theta(i,j)>=-3*pi/4)&(theta(i,j)<-
pi/2))
            g1=g(i-1,j+1)*(gx(i,j)/gy(i,j))+g(i-1,j)*(1-gx(i,j)/gy(i,j));
            g2=g(i+1,j-1)*(gx(i,j)/gy(i,j))+g(i+1,j)*(1-gx(i,j)/gy(i,j));
            if (gcom(i,j)<=g1)|(gcom(i,j)<=g2)
                gcom(i,j)=0;
            end
        end
        if ((theta(i,j)>=pi/2)&(theta(i,j)<3*pi/4))|((theta(i,j)>=-pi/2)&(theta(i,j)<-
pi/4))
            g1=g(i-1,j-1)*(-gx(i,j)/gy(i,j))+g(i-1,j)*(1+gx(i,j)/gy(i,j));
            g2=g(i+1,j+1)*(-gx(i,j)/gy(i,j))+g(i+1,j)*(1+gx(i,j)/gy(i,j));
            if (gcom(i,j)<=g1)|(gcom(i,j)<=g2)
                gcom(i,j)=0;
            end
        end
        if ((theta(i,j)>=3*pi/4)&(theta(i,j)<=pi))|((theta(i,j)>=-pi/4)&(theta(i,j)<0)
)
            g1=g(i-1,j-1)*(-gy(i,j)/gx(i,j))+g(i,j-1)*(1+gy(i,j)/gx(i,j));
            g2=g(i+1,j+1)*(-gy(i,j)/gx(i,j))+g(i,j+1)*(1+gy(i,j)/gx(i,j));
            if (gcom(i,j)<=g1)|(gcom(i,j)<=g2)
```

```
                gcom(i,j)=0;
            end
        end
    end
end
g=uint8(gcom); %将更新后的矩阵存回 g 中
high=30;low=5; %设置高低阈值
gunderlow=uint8(g>=low);    %低于低阈值的点
g1=g.*gunderlow;    %将低于低阈值的点置 0
gbetween=uint8(g1<=high);   %介于两阈值之间的点（待确定点）
gsure=uint8(g1>high);   %高于高阈值的点，即确定保留的点
[m,n]=size(gsure);z1=zeros(1,n);z2=zeros(m,1);  %z1，z2 分别是水平竖直的 0 向量
%以下操作将高于高阈值的点扩展到 8-邻域：
gsureex=gsure+[z2,gsure(:,1:n-1)]+[gsure(:,2:n),z2]+[z1;gsure(1:m-1,:)]+[gsure(2:m,:);
z1]...
    +[z2,[z1(1:n-1);gsure(1:m-1,1:n-1)]]+[z2,[gsure(2:m,1:n-1);z1(1:n-1)]]...
    +[[z1(1:n-1);gsure(1:m-1,2:n)],z2]+[[gsure(2:m,2:n);z1(1:n-1)],z2];
gsureex=uint8(gsureex>0);
gbetween1=gbetween.*gsureex;    %得到 8-邻域中有高于高阈值点的待确定点
Img_out=g1.*(gbetween1+gsure);  %两者合并即为最终保留的点
Img_out=255*Img_out;    %将最终保留的点强化
end
```