**实验六 图像分割（一）实验报告**

**实验目的：**

1. 了解图像分割的基本理论和方法；
2. 掌握对图像进行点、线和边缘检测的方法；
3. 掌握阈值分割的方法和阈值的选择；

**实验内容：**

1. 对图像mask.tif分别用水平、＋45度、垂直和－45度Kirsch算子进行处理，观察效果图，理解不同方向的模板对图像处理的作用。

下面是四个算子

function kirschTEST(p);

f=imread(p);

g=double(f);

h1=[5 5 5;-3 0 -3;-3 -3 -3];%水平

h5=[-3 -3 -3;-3 0 -3;5 5 5];

g1=imfilter(g,h1);

subplot(121),imshow(g1);

title('水平(1)')；

g11=imfilter(g,h5);

subplot(122),imshow(g11);

title('水平(2)')

function kirschTEST1(p);

f=imread(p);

g=double(f);

h2=[-3 5 5;-3 0 5;-3 -3 -3];%-45度

h6=[-3 -3 -3;-3 0 -3;5 5 5];

g2=imfilter(g,h2);

subplot(121),imshow(g2);

title('-45度(1)');

g22=imfilter(g,h6);

subplot(122),imshow(g22);

title('-45度(2)');

function kirschTEST2(p);

f=imread(p);

g=double(f);

h3=[-3 -3 5;-3 0 5;-3 -3 5];%垂直

h7=[5 -3 -3;5 0 -3;5 -3 -3];

g3=imfilter(g,h3);

subplot(121),imshow(g3);

title('垂直(1)');

g33=imfilter(g,h7);

subplot(122),imshow(g33);

title('垂直(2)');

function kirschTEST3(p);

f=imread(p);

g=double(f);

h4=[-3 -3 -3;-3 0 5;-3 5 5];%+45度

h8=[5 5 -3;5 0 -3;-3 3 -3];

g4=imfilter(g,h4);

subplot(121),imshow(g4);

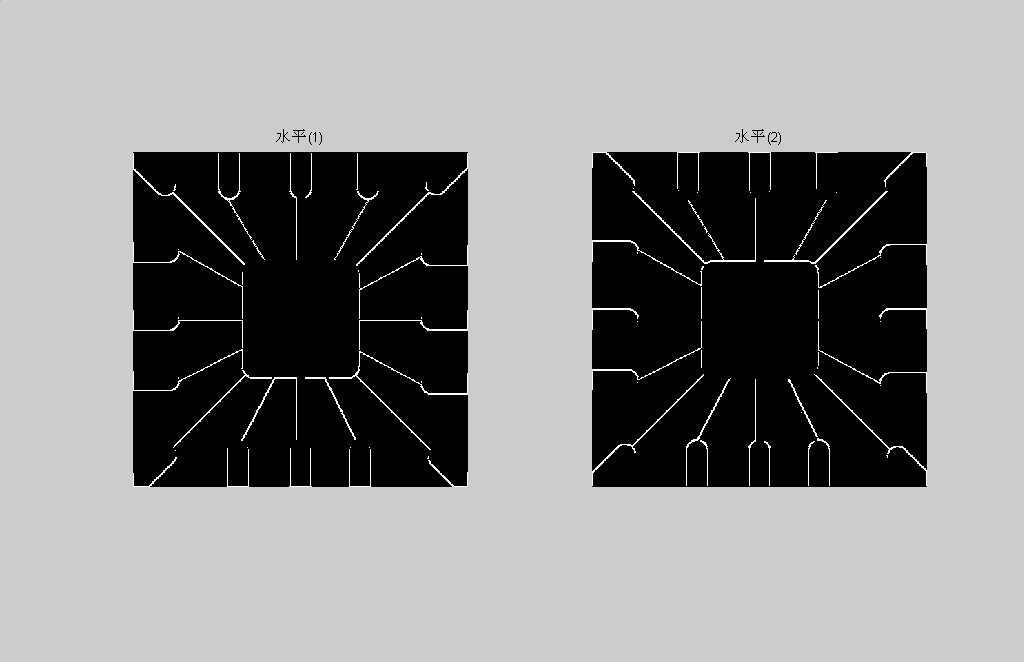
title('+45度(1)');

g44=imfilter(g,h8);

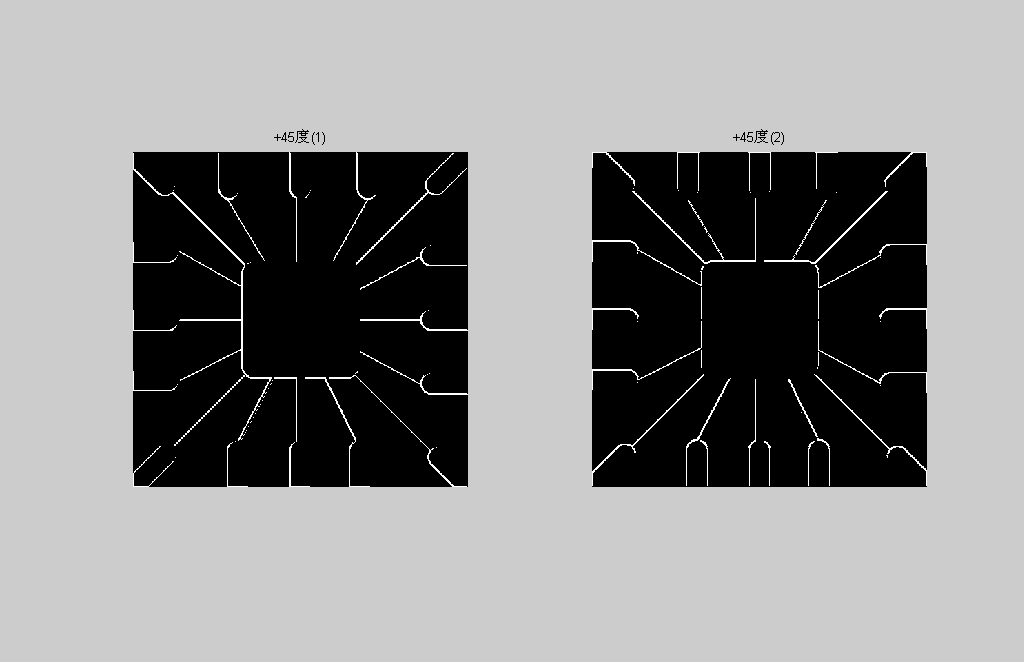
subplot(122),imshow(g44);

title('+45度(2)');

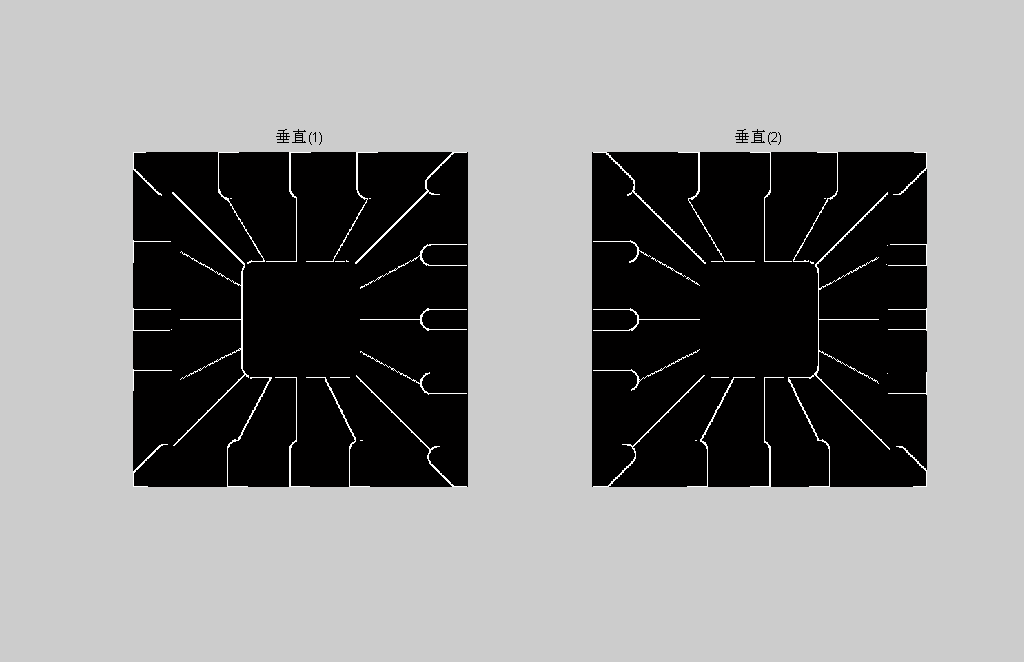
>> kirschTEST('mask.tif');



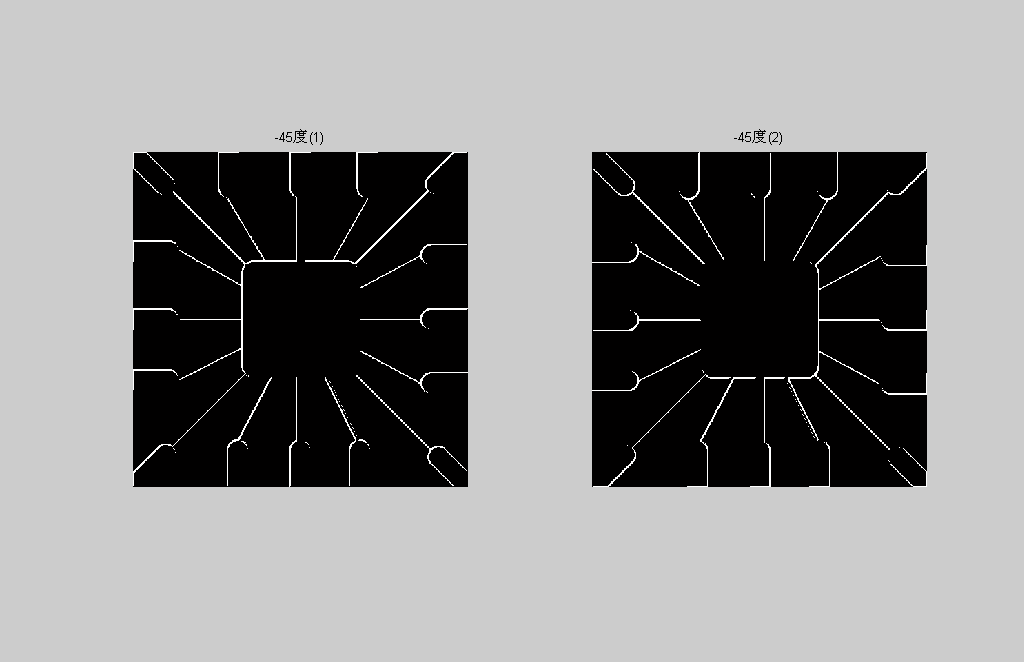
>> kirschTEST1('mask.tif');



>> kirschTEST2('mask.tif');



>> kirschTEST3('mask.tif');



对于以上四个方向的掩膜，可以看到水平掩膜可最佳响应图片中水平的线；+45°掩膜可最佳响应图片中+45°的线；垂直掩膜可最佳响应图片中垂直的线；-45°掩膜可最佳响应图片中-45°的线

而对于每一组的掩膜，虽然它们是同一个方向上的掩膜，但是对图片的影响是有点不一样的，每一组的第二幅图片都可以有第一幅图片翻转90°得到，这是因为每一组算子的权值刚好是旋转90度对称的，例如第一组（使用了水平算子的）图片中的水平方向上的线明显比较清晰明显。

1. 掌握图像处理Sobel、Prewitt和Roberts模板，然后分别调用edge函数对图片building.tif进行处理，分析处理效果图和模板之间的关系。

>> edgeTEST('building.tif');

>> edgeTESTp('building.tif');

function edgeTEST(p);

g=imread(p);

subplot(121),imshow(g);

title('原图');

g1=edge(g,'sobel');

subplot(122),imshow(g1);

title('sobel');

function edgeTESTp(p);

g=imread(p);

g2=edge(g,'Prewitt');

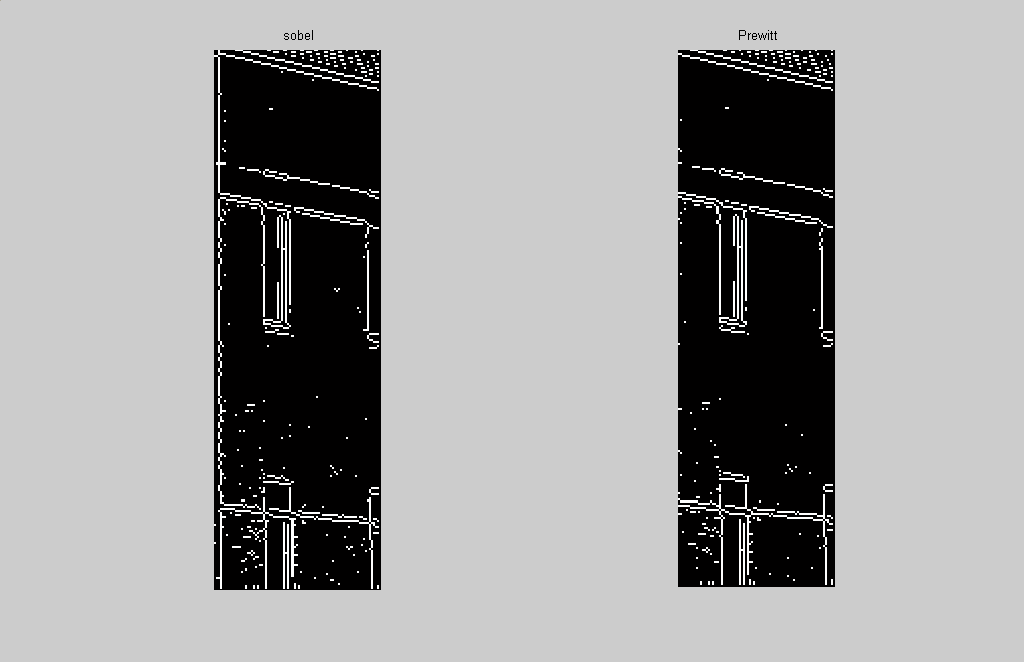
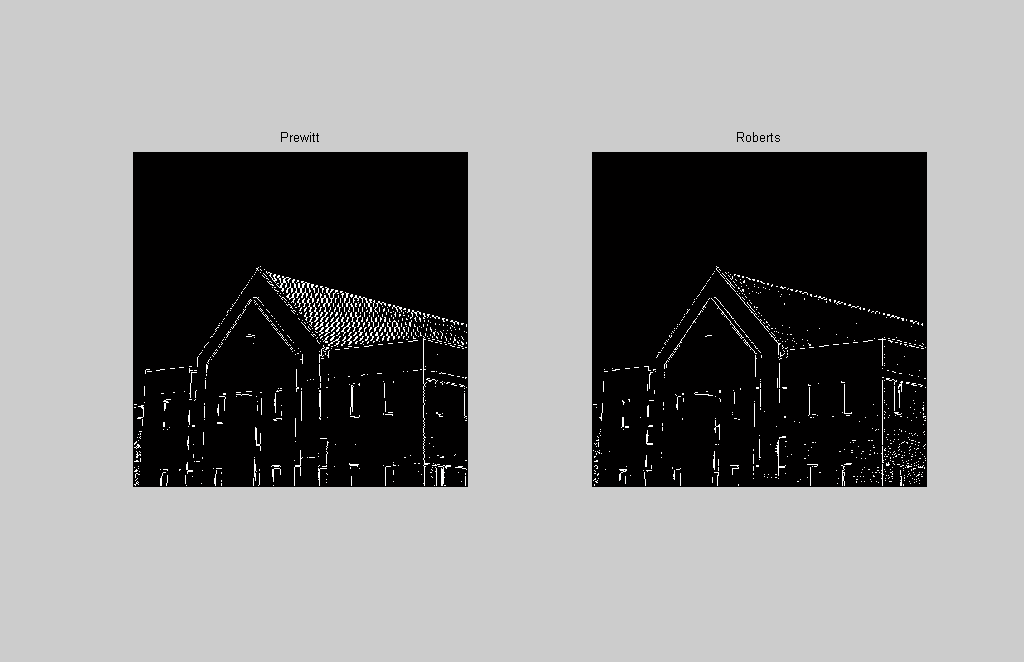
subplot(121),imshow(g2);

title('Prewitt');

g3=edge(g,'Roberts');

subplot(122),imshow(g3);

title('Roberts');



sobel算子和prewitt算子对building.Tif这幅图片大体上是没有太大的影响的，但是当放大图片之后，我们发现应用了sobel算子后图片的噪点会比用prewitt算子的要多一点。（原因是：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，边缘定位精度不够高。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。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。）所以使用sobel算子和prewitt算子对图像进行处理后，sobel算子处理后的图像的清晰度（定位精度）要比用prewitt算子高。

Roberts模板是一个2\*2的模板，由于它的非对称性，因而它不能检测诸如45°倍数的边缘，于是，我们看到的房子的屋顶的瓦片没有前面两个算子的清晰。

1. 利用canny算子对图片building.tif进行处理，深刻理解该算子的作用，并和题2比较。P294

Canny边缘检测算法

(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) 对梯度幅值进行非极大值抑制。

(4) 用双阈值算法检测和连接边缘。

**Canny算子检测边缘的方法是寻找图像梯度的局部极大值，梯度是用高斯滤波器的导数计算的。Canny 方法使用两个阈值来分别检测强边缘和弱边缘，而且仅当弱边缘与强边缘相连时，弱边缘才会包含在输出中。因此，此方法不容易受噪声的干扰，能够检测到真正的弱边缘。**

BW=edge(I, 'canny')

BW=edge (I, 'canny',thresh)

BW=edge (I, 'canny', thresh, sigma)

[BW, thresh]=edge (I, 'canny'…)

BW=edge(I, 'canny') 自动选择阈值用Canny算子进行边缘检测。

BW=edge(I, 'canny',thresh)根据所指定的敏感度阈值thresh，用Canny算子进行边缘检测，thresh是一个含两个元素的矢量，第一个元素是低阈值，第二个元素是高阈值；如果只给thresh指定一个值，则此值作为高阈值，而0.4×thresh作为低阈值；当thresh为空时，自动选择低阈值和高阈值。

BW=edge (I, 'canny', thresh, sigma)根据所指定的敏感度阈值thresh和标准偏差sigma，用Canny算子进行边缘检测，默认时sigma等于1，滤波器的尺寸sigma自动选择。

[BW, thresh]=edge (I, 'canny'…) 返回含两个元素的阈值矢量。

>> [g\_s,ts]=edge(i,'sobel');

>> ts

ts =

0.0738

>> [g\_c,tc]=edge(i,'canny');

>> tc

tc =

0.0188 0.0469

下面是sobel算子得出的图像

>> imshow(g\_s);



下面是canny算子得出的图像

>> imshow(g\_c);



这里挑了用sobel算子和canny算子处理过的进行对比，可以发现canny算子因其**使用两个阈值来分别检测强边缘和弱边缘，所以能精确地将房子的所有线条细节都显示出来（包括墙面上的砖块的线条），就如上图所示，但是这样没有完全达到目的，因为**使用算子目的是通过提取房屋的主要边缘特征并去掉不重要的细节，从而产生一个干净的边缘“映射”，但是我们发现直接用canny算子的默认阈值显然得出来的图像并不是我们想要的结果，除了sobel图像外，通过其他算子（默认阈值）得出的图像都与想象的相差甚远。所以要修改参数已达到较好的效果。

>> g\_sobel=edge(f,'sobel',0.05);



>> g\_canny=edge(f,'canny',[0.04 0.10],1.5);



但是当canny的阈值取到【0.08 0.4】的时候线条就更少了



为什么改变阈值的范围会出现这么大的变化呢？这是由于双阈值算法检测的原理：凡是大于高阈值的一定是边缘，凡是小于低阈值的一定不是边缘。当实验时将阈值从[0.04 0.10]改变到【0.08 0.4】时，这就将低阈值提高了，原来被认为是边缘的在改变双阈值之后已经变成是属于不是边缘的范围，这样修改阈值后显示的图像的线条更加少了，这样反而远离了我们要达到的效果，所以在使用canny算子对图像进行处理的时候要注意选择两个阈值的大小，才能得到较好的效果。

1. 对building.tif图像进行阈值分割，先用直方图人工选择其阈值进行分割，然后用graythresh函数计算其阈值进行处理。最好在自己编写m文件，完成全局阈值的自动获取（P179）。

采用阈值确定边界的最简单方法就是在整个图像中将灰度阈值的值设置为常数。如果背景的灰度值在整个图像中可合理地看做恒定，而且所有物体与背景都具有几乎相同的对比度。那么，只要选择正确的阈值，使用一个固定的全局阈值一般会有好的效果。一般运用全局阈值法的时候就会想到最佳阈值法，通过选择最佳的阈值进行分割图像，而不是利用最简单的im2bw语句来实现。最佳阈值的选择需要根据具体问题来确定，一般通过实验来确定。对于给定的图像，可以通过分析直方图的方法确定最佳的阈值。例如，当直方图明显呈现双峰情况时，可以选择两个峰值的中点作为最佳阈值。

在最佳阈值分割的情况下，取阈值分割后的图像可表示为，若，，*k* (,,…,为一系列分割阈值)。

下面是一种通过迭代求图像最佳分割阈值的方法。迭代法是基于逼近的思想：

(1) 求出图像中的最小和最大灰度值和的阈值初值。

(2) 根据阈值*Tk* 将图像分割成目标和背景两部分，求出两部分的平均灰度值和。其中*Z* ( *i*, *j*)是图像上( *i*, *j*)点的灰度值，*N* ( *i*, *j*)是( *i*, *j*)点的权重系数，*N* ( *i*, *j*)取( *i*, *j*)点灰度的概率。

(3) 求出新的阈值。

(4) 若结束, 否则*k* + 1→*k*转第二步。

(5) 第四步结束后*Tk*即为最佳阈值。

下面是使用了直方图人工选择其阈值进行分割

>> f=imread('building.tif');

>> figure,imhist(f);

>> t=165;%根据得出的直方图，本次测试选取了t=165为阈值

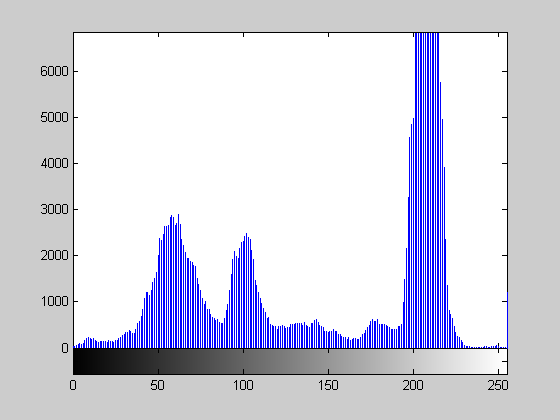
>> t=t/255

t =

0.6471

>> g1=im2bw(f,t);

>> figure,imshow(g1);



下面是使用了graythresh函数计算其阈值进行处理的图像

>> f=imread('building.tif');

>> tt=graythresh(f);

>> g2=im2bw(f,tt);

>> figure,imshow(g2);

>> tt

tt =

0.5490



下面是一个使用迭代方式计算全局阈值的函数

function [g,T]=iterating(f)

T=0.5\*(double(min(f(:)))+double(max(f(:))));

done=false;

while ~done

g=f>=T;

Tnext=0.5\*(mean(f(g))+mean(f(~g)));

done=abs(T-Tnext)<0.5;

T=Tnext;

end

T=T/255;

g=im2bw(f,T);

>> [g,t3]=iterating(f);

>> imshow(g);

>> t3

t3 =

0.5514



下面这个是graythresh函数的源文件，为了了解该函数的实质，所以挑了出来

function level = graythresh(I)

%GRAYTHRESH Compute global image threshold using Otsu's method.

% LEVEL = GRAYTHRESH(I) computes a global threshold (LEVEL) that can be

% used to convert an intensity image to a binary image with IM2BW. LEVEL

% is a normalized intensity value that lies in the range [0, 1].

% GRAYTHRESH uses Otsu's method, which chooses the threshold to minimize

% the intraclass variance of the thresholded black and white pixels.

%

% Class Support

% -------------

% The input image I can be of class uint8, uint16, or double and it

% must be nonsparse. LEVEL is a double scalar.

%

% Example

% -------

% I = imread('blood1.tif');

% level = graythresh(I);

% BW = im2bw(I,level);

% imshow(BW)

%

% See also IM2BW.

% Copyright 1993-2002 The MathWorks, Inc.

% $Revision: 1.9 $ $Date: 2002/03/15 15:27:29 $

% Reference:

% N. Otsu, "A Threshold Selection Method from Gray-Level Histograms,"

% IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1,

% pp. 62-66, 1979.

% One input argument required.

checknargin(1,1,nargin,mfilename);

checkinput(I,'uint8 uint16 double','nonsparse',mfilename,'I',1);

% Convert all N-D arrays into a single column. Convert to uint8 for

% fastest histogram computation.

I = im2uint8(I(:));

num\_bins = 256;

counts = imhist(I,num\_bins);

% Variables names are chosen to be similar to the formulas in

% the Otsu paper.

p = counts / sum(counts);

omega = cumsum(p);

mu = cumsum(p .\* (1:num\_bins)');

mu\_t = mu(end);

% Save the warning state and disable warnings to prevent divide-by-zero

% warnings.

state = warning;

warning off;

sigma\_b\_squared = (mu\_t \* omega - mu).^2 ./ (omega .\* (1 - omega));

% Restore the warning state.

warning(state);

% Find the location of the maximum value of sigma\_b\_squared.

% The maximum may extend over several bins, so average together the

% locations. If maxval is NaN, meaning that sigma\_b\_squared is all NaN,

% then return 0.

maxval = max(sigma\_b\_squared);

if isfinite(maxval)

idx = mean(find(sigma\_b\_squared == maxval));

% Normalize the threshold to the range [0, 1].

level = (idx - 1) / (num\_bins - 1);

else

level = 0.0;

end

1. 附加题（选做）：编写m文件实现基本自适应阈值分割，参数为图片、子图像的大小，输出为分割后的图片。

**实验报告：**

1. **实验目的**
2. **实验内容（部分可略写）每题分开写，并标明题号**
3. **实验小结和体会**