数字图像处理Project 1

曾烈康 14348162

zenglk3@mail2.sysu.edu.cn

2017-03-26

2017

目 录

[1 Technical discussion 2](#_Toc478320416)

[1.1 Project 3.2 2](#_Toc478320417)

[1.2 Project 3.5 3](#_Toc478320418)

[1.3 Project 3.6 3](#_Toc478320419)

[2 Discussion of results 5](#_Toc478320420)

[2.1 Project 3.2 5](#_Toc478320421)

[2.2 Project 3.5 6](#_Toc478320422)

[2.3 Project 3.6 8](#_Toc478320423)

[3 Program listings 12](#_Toc478320424)

[3.1 Project 3.2 12](#_Toc478320425)

[3.2 Project 3.5 13](#_Toc478320426)

[3.3 Project 3.6 14](#_Toc478320427)

# Technical discussion

## Project 3.2

1. 项目内容

本题目标可简化为如下三点：

1. 计算一张图片的直方图
2. 实现直方图均衡化技术
3. 对给定图像应用直方图均衡化
4. 实现原理

直方图均衡化的作用是图像增强。其实现关键是利用累积分布函数使像素值均匀分布。

一幅图像的灰度级可以看成是区间[0,L-1]内的随机变量，随机变量的基本描绘子是其概率密度函数PDF。实际应用中直方图是概率密度函数的近似，在处理过程中不能产生新的灰度级，原来像素值的大小关系也需要保持不变。累积分布函数是一个单调不减函数，且值域必然处于[0,1]，所以选择使用累积分布函数来实现均衡化的过程。

1. 实现步骤
2. 加载图像
3. 统计图像中各灰度值的像素个数，由此计算各灰度值在图像中的出现概率，得到原始图像的灰度值序列，绘制原始图像的直方图
4. 对每个灰度值计算均衡化函数，具体公式是
5. 对各灰度值经均衡化函数后计算的值四舍五入，得到对应的新灰度值
6. 对新产生的灰度值序列，统计各灰度值的出现概率，绘制均衡化后的直方图
7. 将原始灰度值序列中的灰度值替换成对应的均衡化后的灰度值，得到均衡化后的图像
8. 保存均衡化后的图像

## Project 3.5

1. 项目内容

本题目标可简化为如下两点：

1. 实现拉普拉斯增强技术
2. 对给定图像应用拉普拉斯增强技术
3. 实现原理
   1. 拉普拉斯是一种微分算子，其应用强调的是图像中灰度的突变，而不强调灰度级缓慢变化的区域，这将产生把浅灰色边线和突变点叠加到暗色背景中的图像。将原图像和拉普拉斯滤波后得到的图像加在一起，可以复原背景特性并保持拉普拉斯锐化处理的效果。
   2. 拉普拉斯变换的算子是一个线性算子，可以有多种滤波器模板。例如下面几种拉普拉斯滤波器矩阵。如果使用的滤波器矩阵具有负的中心系数，那么需要将原始图像减去经拉普拉斯变换后的图像而不是加上它，才能得到锐化结果。根据题意，我们使用中心系数为8的滤波器模板。

* 1. 拉普拉斯滤波器模板矩阵对原始图像滤波过程本质上是二维卷积，在实现过程中，我使用了BlurFilter函数进行专门处理。设mask为滤波器模板矩阵，im为原始图像矩阵，imLF是卷积后的矩阵，于是计算公式为

1. 实现步骤
2. 加载图像
3. 应用给定的拉普拉斯滤波器对原始图像进行滤波处理，得到图像的细节
4. 将图像的细节与原始图像相加，得到拉普拉斯增强后的图像
5. 保存拉普拉斯增强后的图像

## Project 3.6

1. 项目内容

本题目标可简化为如下两点：

1. 实现高提升滤波技术
2. 对给定图像应用高提升滤波技术
3. 实现原理
   1. 非锐化掩模工作原理可借用课本上的图示说明，如图 Figure 1 非锐化掩模工作原理

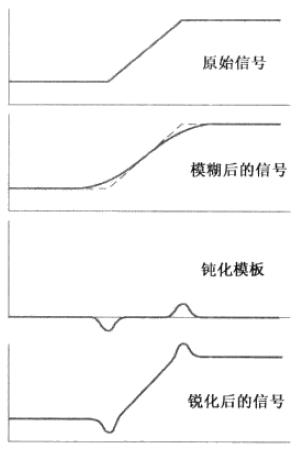


Figure 非锐化掩模工作原理

* 1. 掩模矩阵对原始图像的滤波处理本质是二维卷积，与Project 3-5中的拉普拉斯滤波处理相同，因此可以共用滤波函数，只需修改掩模矩阵即可。

1. 实现步骤
2. 加载图像
3. 应用给定的掩模矩阵对原始图像进行滤波处理，由原图像减去模糊图像得到模板图像
4. 将原始图像乘系数(A-1)后与模板图像相加，得到高提升图像
5. 保存保存高提升图像

# Discussion of results

## Project 3.2

实验结果如下。

Figure 原始图像 Figure 均衡化后的图像

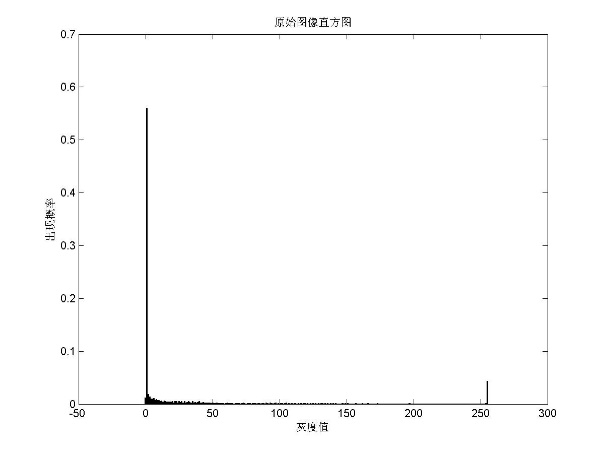
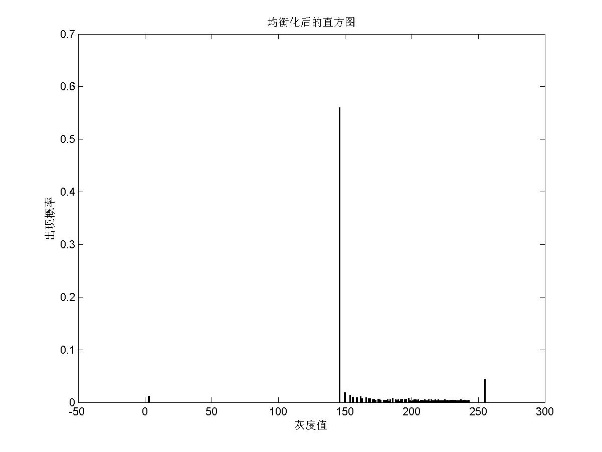
 

Figure 原始直方图 Figure 均衡化后的直方图

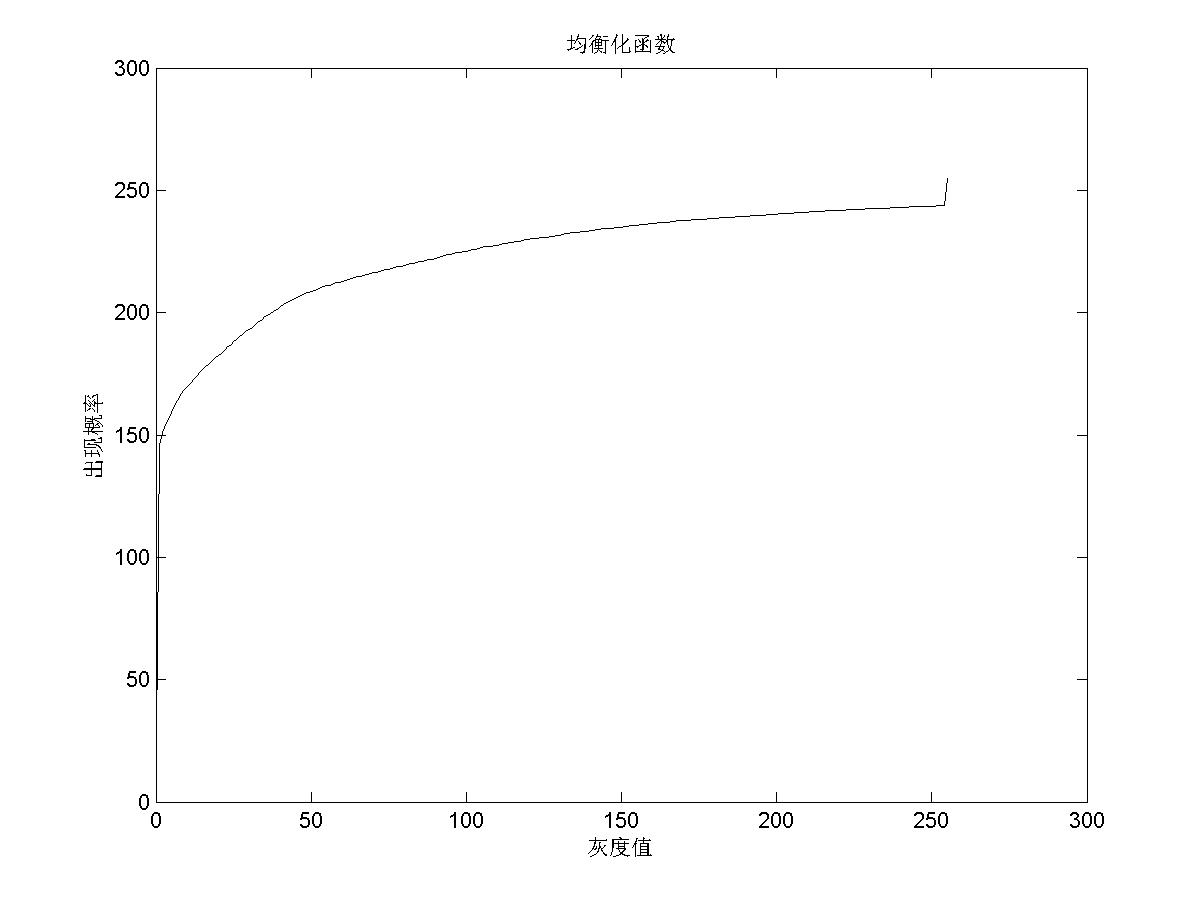


Figure 均衡化函数

可以看出，直方图均衡化应用于该图效果并不好。由原始图像的直方图可以发现，原始图像的灰度分布比较极端，由此导致均衡化后的直方图灰度分布同样不理想，所以均衡化后的图像效果也不好。

将均衡化前后两张直方图相比可知，均衡化使图像的灰度分布整体右移，所以图像整体的灰度变高。原始图像中两边的黑色部分几乎全部变换为灰色。但是直方图中明暗灰度的相对分布位置并未改变，低灰度的部分仍然占据了非常高的直方图成分。

此图体现了直方图均衡化的不足，对于特殊的直方图可以进一步使用直方图匹配的方法来处理。

## Project 3.5

实验结果如下。

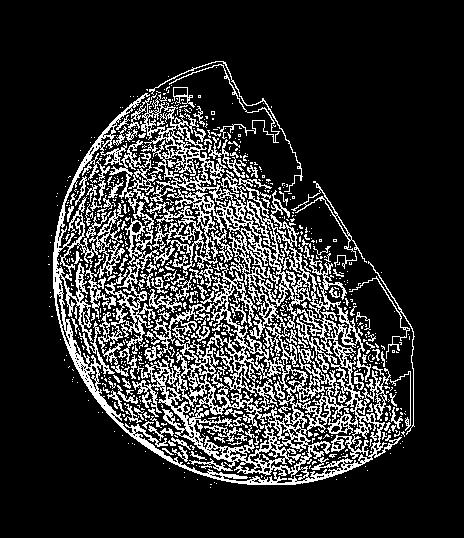
 

Figure 原始图像 Figure 拉普拉斯滤波后的图像



Figure 拉普拉斯增强后的图像

本题中采用的滤波器矩阵为

经该滤波器矩阵处理得到的图像细节如Figure 7 原始图像 Figure 8拉普拉斯滤波后的图像所示，可以看出该图将原始图像大量细节保留了下来。

将Figure 7 原始图像 Figure 8拉普拉斯滤波后的图像与Figure 9 拉普拉斯增强后的图像相加得到拉普拉斯增强后的图像。显然，该图比原始图像包含了更多的细节，显得更为清晰。

## Project 3.6

实验结果如下。

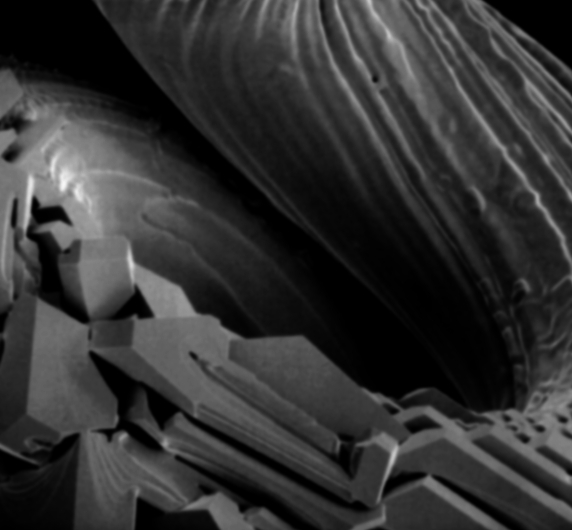


Figure 原始图像

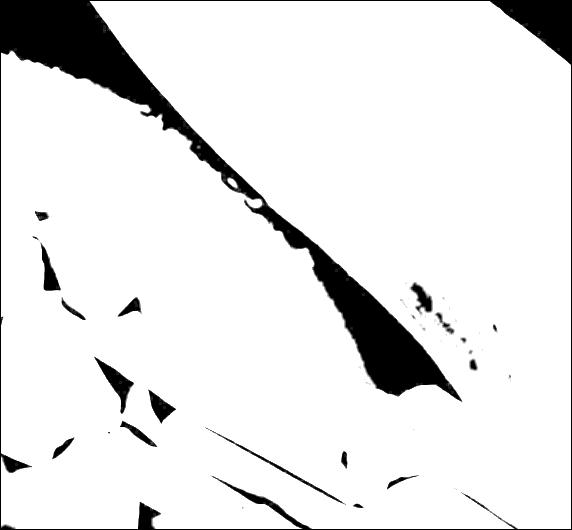
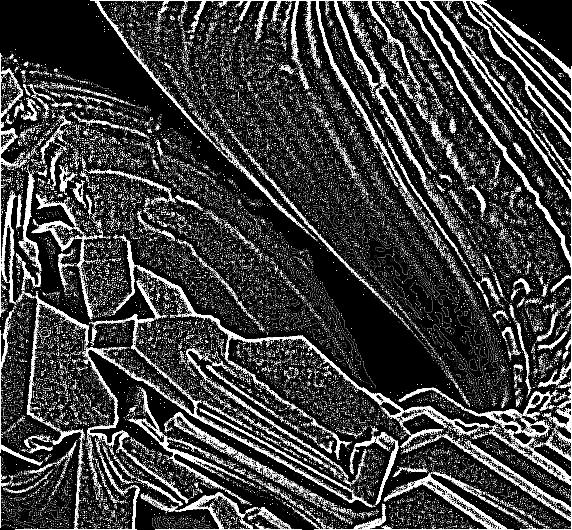
 

Figure 模糊图像 Figure 模板图像

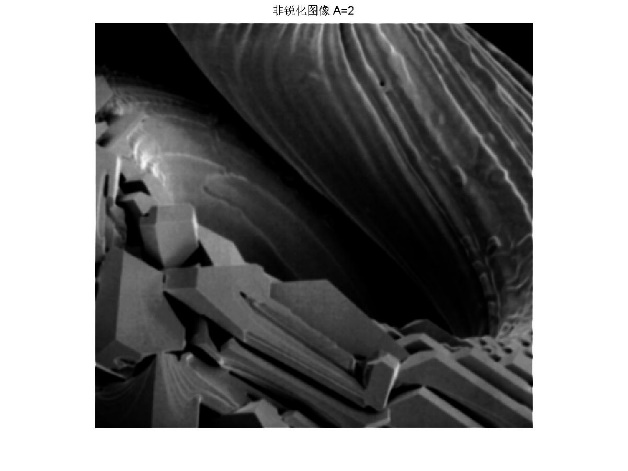


Figure 非锐化图像 A=1 Figure 高提升图像 A=2

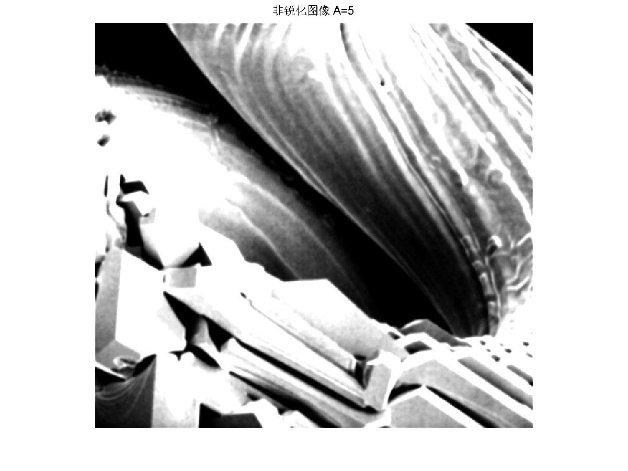
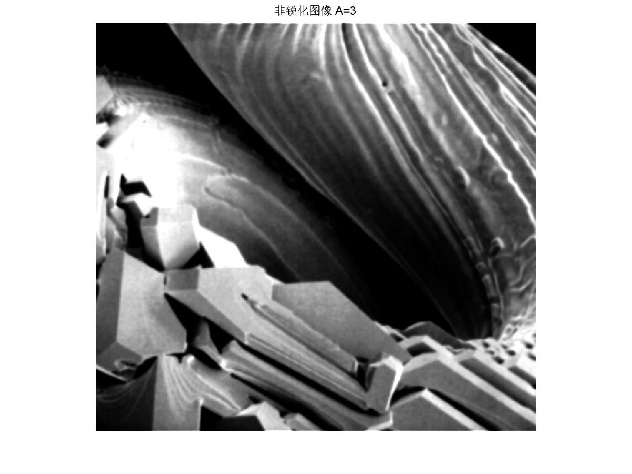


Figure 高提升图像A=3 Figure 高提升图像 A=5



Figure 高提升图像 A=10 Figure 高提升图像 A=20



Figure 高提升图像A=30 Figure 高提升图像 A=50

模糊图像由均值滤波器矩阵处理得到，模板图像由原图像减去模糊图像得到。而非锐化图像的计算，在课本第二版与第三版中公式是不一样的。课本第三版中，高提升图像由原图像加上乘系数k的模板图像得到，计算公式为

而课本第二版中，高提升图像由原图像乘系数A再减去模糊图像得到，或者写为原图像乘系数（A-1）再加上模板图像得到，计算公式为

本实验中采用课本第二版的计算公式。

非锐化图像中分别设置A=1,2,3,5,10,20,30,50来处理图像。可以发现，随着A的增加，图像亮度逐渐增加。然而随着A的增大，图像亮度会出现过大的情况，图像质量反而降低，可见A的设置是有上限的，且存在一个合适的值使图像提升效果达到最佳。本题随机设置的一系列A值中，个人认为A=2时效果最佳。为得到更精确的最优值，缩小步长为0.1，测试得到A=2.7附近时效果最佳，A=2.7时的高提升图像如图Figure 21 高提升图像 A=2.7（见下页）。若需要更高精度则需要进一步缩小精度进行测试。

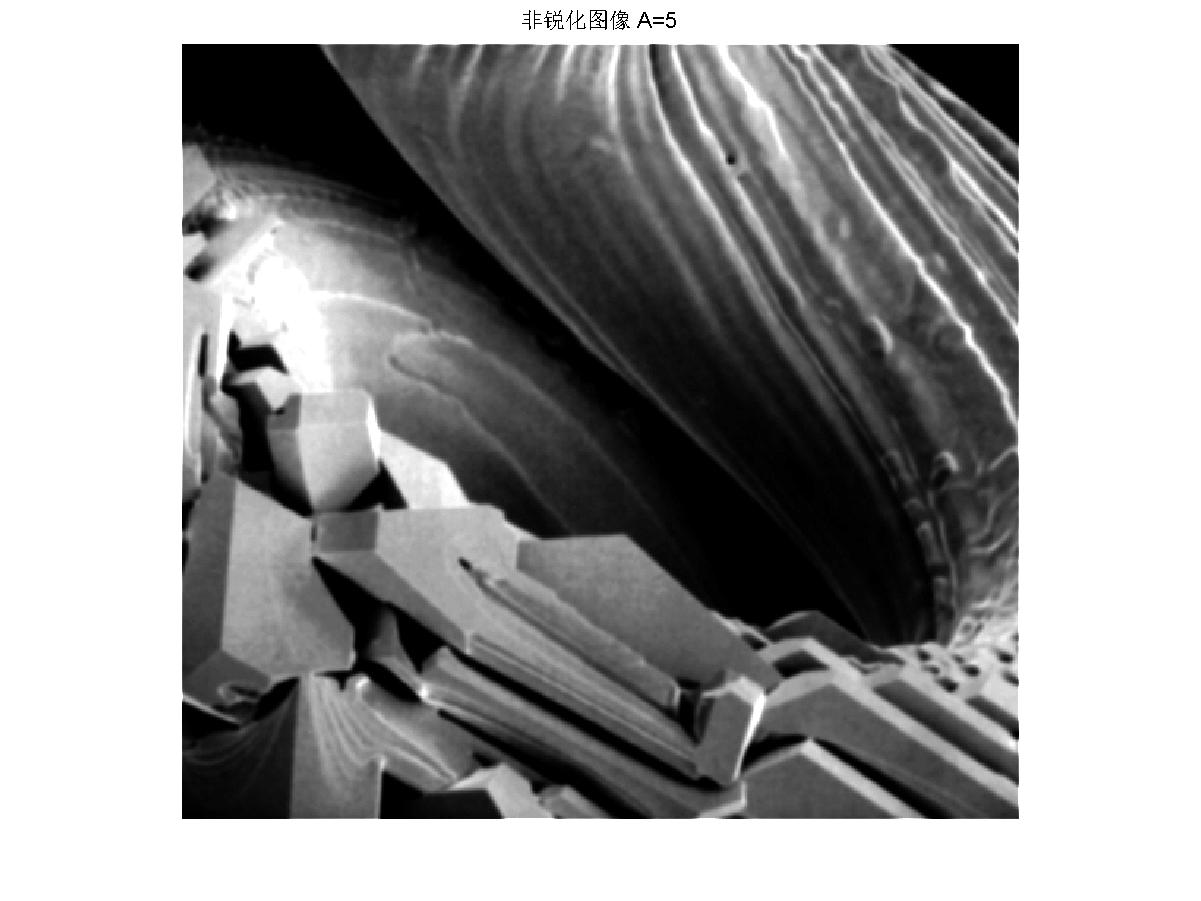


Figure 高提升图像 A=2.7

# Program listings

## Project 3.2

|  |
| --- |
| function Histogram\_Equalization()  close all  fi=imread('Fig3.08(a).jpg');  figure,imshow(fi),title('原始图像')    [m,n]=size(fi);  Pr=zeros(1,256);  for k=0:255  Pr(k+1)=length(find(fi==k))/(m\*n); %统计灰度值为k的像素的出现概率  end    figure(1),bar(0:255,Pr,'k'),title('原始图像直方图')  xlabel('灰度值'),ylabel('出现概率'); %打印原始图像直方图  saveas(1,'oringin\_hist.jpg');    sumPr=zeros(1,256);  for i=1:256  for j=1:i  sumPr(i)=Pr(j)\*255+sumPr(i); %均衡化函数：对每个灰度i求和灰度0~i的概率  end  end    figure(2),plot(0:255,sumPr,'k'),title('均衡化函数')  xlabel('灰度值'),ylabel('出现概率'); %打印均衡化函数  saveas(2,'eq\_plot.jpg');    newGray=round((sumPr)+0.5); %对每个灰度的概率求对应的新的灰度值并四舍五入  for i=1:256  Ps(i)=sum(Pr(newGray==i)); %ps(i)表示灰度为i的像素出现的概率  end    figure(3),bar(0:255,Ps,'k'),title('均衡化后的直方图')  xlabel('灰度值'),ylabel('出现概率'); %打印均衡化后的直方图  saveas(3,'new\_hist.jpg');    fiEq=fi;  for i=0:255  fiEq(fi==i)=newGray(i+1); %对fi中灰度为i的像素俊替换为newGray中的新灰度  end  figure,imshow(fiEq),title('均衡化后图像');  imwrite(fiEq,'Fig3.08(a)Eq.jpg'); |

## Project 3.5

|  |
| --- |
| function Laplacian\_Enhancement()  close all  fi=imread('Fig3.40(a).jpg');  figure,imshow(fi),title('原始图像'); %打印原始图像  mask = [-1,-1,-1;-1,8,-1;-1,-1,-1]; %mask表示掩模矩阵  imLF = BlurFilter(fi,mask);  figure,imshow(imLF),title('拉普拉斯滤波后图像');  imwrite(imLF,'Fig3.40(a)mask.jpg');    imLE = double(fi)+double(imLF);  imLE = uint8(imLE);  figure,imshow(imLE),title('拉普拉斯增强后图像');  imwrite(imLE,'Fig3.40(a)Lap.jpg');    function imLF = BlurFilter(fi,mask) %计算滤波后的矩阵  im = double(fi);  [m,n] = size(fi);  imLF = zeros(m,n);  for x = 2:m-1 %对每个像素点，按掩模矩阵加权求和周围的像素值  for y = 2:n-1  imLF(x,y)=im(x-1,y-1)\*mask(1,1)+im(x-1,y)\*mask(1,2)+im(x-1,y+1)\*mask(1,3)...  +im(x,y-1)\*mask(2,1)+im(x,y)\*mask(2,2)+im(x,y+1)\*mask(2,3)...  +im(x+1,y-1)\*mask(3,1)+im(x+1,y)\*mask(3,2)+im(x+1,y+1)\*mask(3,3);  end  end |

## Project 3.6

|  |
| --- |
| function Unsharp\_Masking()  close all  fi=imread('Fig3.43(a).jpg');  figure,imshow(fi),title('原始图像');  im = double(fi);  mask = [1/9 1/9 1/9;1/9 1/9 1/9;1/9 1/9 1/9];%mask表示掩模矩阵  %c=conv2(fi,w,'same'); %可用系统自带的卷积函数实现  c = BlurFilter(im,mask);  c = BlurFilter(im,mask);  figure,imshow(c),title('模糊图像');  imwrite(c,'Fig3.43(a)blur.jpg');  gmask = im-c;  figure,imshow(gmask),title('模板图像');  imwrite(gmask,'Fig3.43(a)mask.jpg');    A=[1,2,3,5,10,20,30,50]; %A中是随机设置的测试常数  for i = 1:length(A)  g = im\*(A(i)-1)+gmask;  g = uint8(g); %每次均打印出图像  figure(i),imshow(g),title(['非锐化图像 A=',num2str(A(i))]);  saveas(i,['Fig3.43(a)\_A',num2str(A(i)),'.jpg']);  end    function imGau = BlurFilter(fi,mask) %计算滤波后的矩阵  im = double(fi);  [m,n] = size(fi);  imGau = zeros(m,n);  for x = 2:m-1 %对每个像素点，按掩模矩阵加权求和周围的像素值  for y = 2:n-1  imGau(x,y)=im(x-1,y-1)\*mask(1,1)+im(x-1,y)\*mask(1,2)+im(x-1,y+1)\*mask(1,3)...  +im(x,y-1)\*mask(2,1)+im(x,y)\*mask(2,2)+im(x,y+1)\*mask(2,3)...  +im(x+1,y-1)\*mask(3,1)+im(x+1,y)\*mask(3,2)+im(x+1,y+1)\*mask(3,3);  end  end |