数字图像处理

课程设计

实习报告

|  |  |
| --- | --- |
| 题 目： | 数字图像处理方法的MATLAB实现 |
| 指导老师： | 刘耀林 |
| 姓 名： | 康雨豪 |
| 学 号： | 2014301130041 |
| 专 业： | 地理信息科学 |

2017年1月10日

# 摘要

**摘要：**数字图像处理是一门集计算机科学、光学、数学、物理学等多学科的综合科学。随着计算机科学的发展，数字图像处理技术取得了巨大的进展，呈现出强大的生命力，已经在多种领域取得了大量的应用，推动了社会的发展。其中，遥感领域中，对于影像数据的处理均基于数字图像处理的技术。而遥感影像数据作为地理信息科学的重要数据源，如何从中获取有用的信息，是地理信息数据处理中重要的内容。本文是笔者在学习完理论知识后对于数字图像处理技术的一次实践体验。MATLAB作为数学领域应用最广泛的一种软件，集成了对于图片处理的函数和功能，成为了处理数字图像问题的佼佼者。其出众的计算能力和简便的绘图能力可以有效进行数字图像的变换和操作。本文根据所学理论概念知识，探究了MATLAB(R2014a)软件下数字图像处理部分简单内容方法的操作实现，包括图像变换、图像增强、图像分割、二值图像处理与形状分析等，作为理论学习后的一种实践补充，加深了笔者对于图像处理模式和细节的理解。由于时间所迫和水平不足，本文中可能会出现一些问题，希望能够指出。

**关键词**：数字图像处理；MATLAB；图像变换；图像增强

**备注：本文中所有的代码m文件均保存在了code文件夹下，图片均保存在了image文件夹下，可供参阅。**

目录

[摘要 2](#_Toc471842175)

[1 数字图像处理基础 2](#_Toc471842176)

[1.1 MATLAB图像处理基本操作 2](#_Toc471842177)

[1.2 图像数字化 3](#_Toc471842178)

[1.3 直方图 4](#_Toc471842179)

[1.4 图像噪声的添加 5](#_Toc471842180)

[2 图像变换 7](#_Toc471842181)

[2.1 傅立叶变换 7](#_Toc471842182)

[2.2 哈达玛变换 12](#_Toc471842183)

[2.3 离散余弦变换 13](#_Toc471842184)

[3 图像增强 14](#_Toc471842185)

[3.1 直方图均衡化 14](#_Toc471842186)

[3.2 灰度变换 16](#_Toc471842187)

[3.3 空间域平滑 18](#_Toc471842188)

[3.4 空间域锐化 22](#_Toc471842189)

[3.5 频率域图像增强 28](#_Toc471842190)

[4 图像分割 34](#_Toc471842191)

[4.1 边缘检测算子 34](#_Toc471842192)

[5 二值图像处理与形状分析 38](#_Toc471842193)

[5.1 二值化 38](#_Toc471842194)

[5.2 膨胀 39](#_Toc471842195)

[5.3 腐蚀 40](#_Toc471842196)

[5.4 开闭运算 41](#_Toc471842197)

[6 总结 43](#_Toc471842198)

1. 数字图像处理基础

本章所有代码均保存在kyh\_One.m文件中。

* 1. MATLAB图像处理基本操作

本文中对于大多数的操作，是对数字图像处理领域中最为著名的“lena”图片进行操作的。原图如下（Figure 1）：



Figure 1

首先，在MATLAB中显示这幅图片：

I=imread('lena.jpg');

imfinfo('lena.jpg')

imshow(I);

其中，imread()可以读取图片，imfinfo()可以获取图片的信息（Figure 2），imshow()可以显示图片。

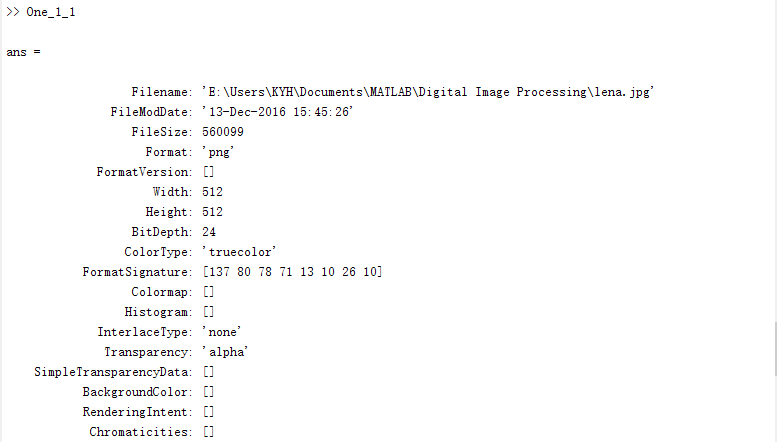


Figure 2

从Figure 2中可以看出，该图片格式为png，长宽均为512px，颜色类型为真彩色。

处理后的图片还需要进行保存：

imwrite(I,'lenaSave.jpg');

* 1. 图像数字化

使用imread()读取图像后，可以看到，读入的图片I是以一个512\*512\*3的矩阵进行保存的，即分别是RGB颜色的数字图片。为了方便起见，在本文中，笔者将lena图进行了转换，转换为灰度图，即保存为一个512\*512\*1的矩阵。

为了完成这样一个步骤，MATLAB中提供了相应的函数：

I1=rgb2gray(I);

imshow(I1);

I1以512\*512\*1的矩阵形式进行存储。效果如下图（Figure 3）：



Figure 3

* 1. 直方图

为了显示图像灰度的分布情况，还需要绘制灰度直方图。可以使用如下代码：

figure,imhist(I1);

效果如下（Figure 4）：

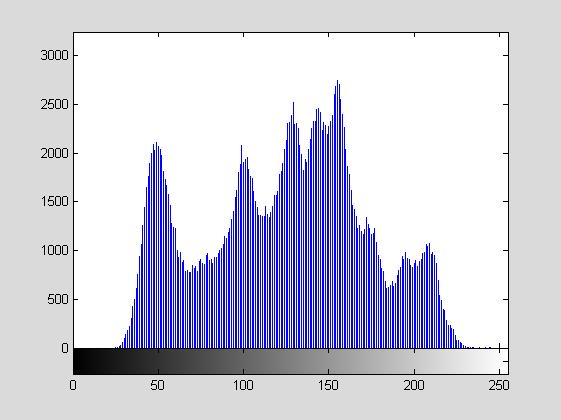


Figure 4

* 1. 图像噪声的添加

为了完成多种图像处理的操作和试验，还可以对图片添加噪声。所用函数为imnoise (I, type)，该函数中的type可以为5种噪声参数，分别为：'gaussian'(高斯白噪声)，'localvar'(与图象灰度值有关的零均值高斯白噪声)，'poisson'(泊松噪声)，'salt & pepper'(椒盐噪声)和'speckle'(斑点噪声)。

以下为高斯噪声（Figure 5）和椒盐噪声（Figure 6）的代码和效果：

I2=imnoise(I1,'gaussian');

figure,imshow(I2);



Figure 5

I3=imnoise(I1,'salt & pepper');

figure,imshow(I3);



Figure 6

1. 图像变换

本章第1小节的代码可以在kyh\_Two1.m中找到，第2-4小节的代码可以在kyh\_Two2.m中找到。

* 1. 傅立叶变换

傅立叶变换可以将图像从空间域转换到频率域，然后再进行相应的处理。MATLAB中有傅立叶变换的函数。在这里，笔者没有使用lena图作为例子，而是采用了教科书中的一个例子。具体如下：

原图：

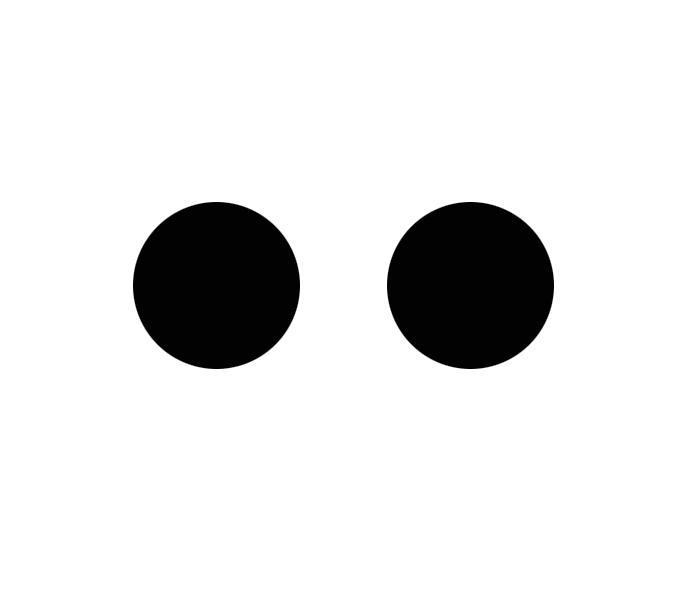


Figure 7

傅立叶变换：

F=fft2(I1);

S=abs(F);

figure,imshow(S,[]);

效果：



Figure 8

看上去是一片黑，事实上在全图的左上角可以看出有一个白点（即左上角缺失了）。

平移：

Fc=fftshift(F);

figure,imshow(abs(Fc),[]);

效果：

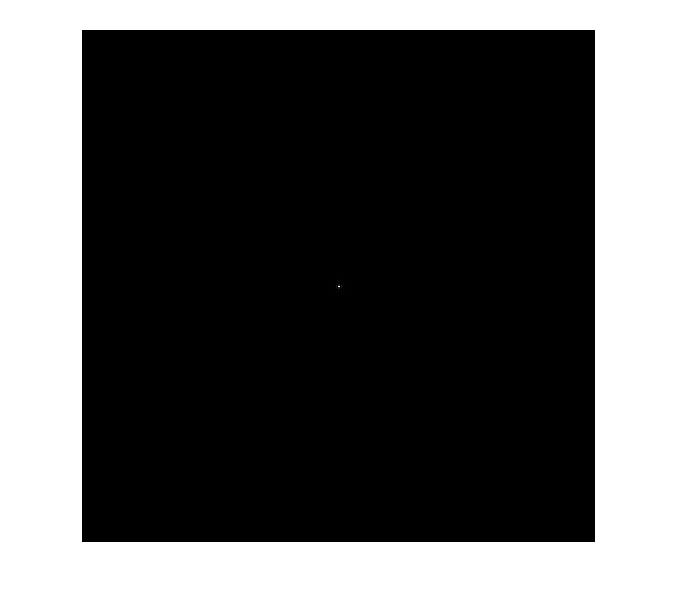


Figure 9

将频率平移到中间。

频谱图：

figure,imshow(uint8(abs(Fc/256)));

效果：

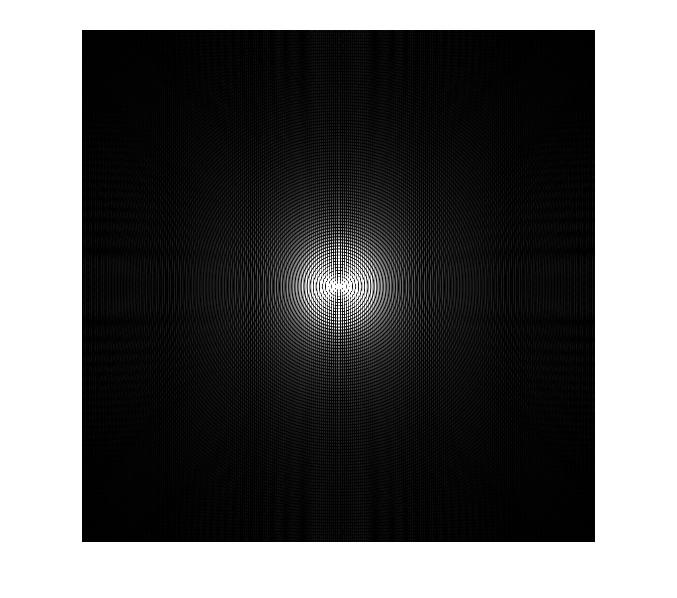


Figure 10

之所以在本例中没有使用lena的图片，是因为该图傅立叶频谱图并无特点。而本例的图与教科书上显示的效果类似。

傅立叶逆变换：

f=real(ifft2(F)/255);

figure,imshow(f);

效果：

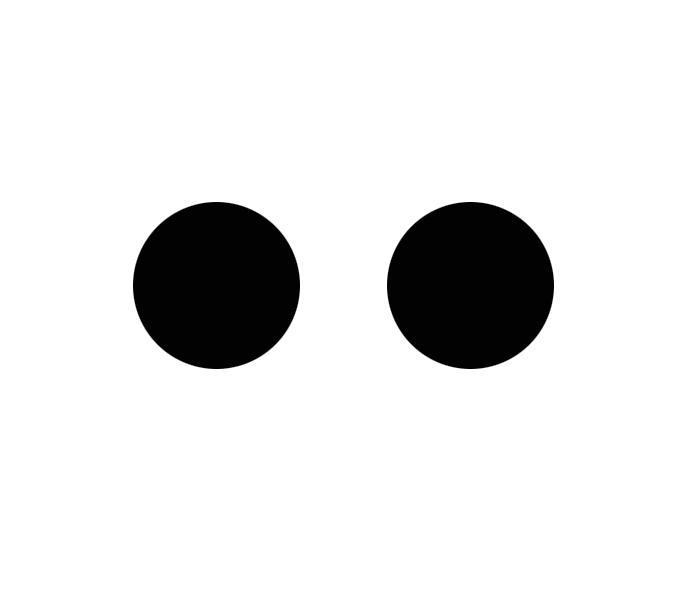


Figure 11

经过逆变换后，仍然为原图。

* 1. 哈达玛变换

离散哈达玛变换的原理在于在图像矩阵左右分别乘以一个正交矩阵。MATLAB中可以直接使用hadamard函数产生哈达玛变换矩阵。

代码：

Idouble=im2double(I1);

h1=size(I1,1);

h2=size(I1,2);

H1=hadamard(h1);

H2=hadamard(h2);

I2=H1\*Idouble\*H2/sqrt(h1\*h2);

figure,imshow(I2);

效果：

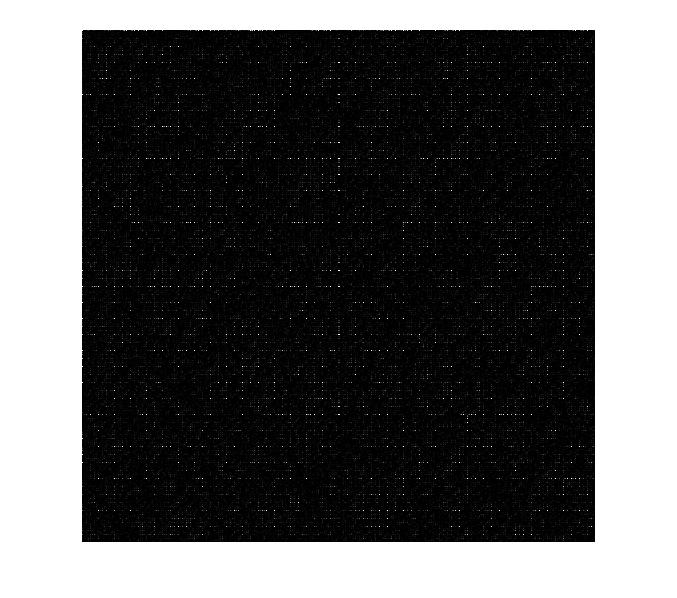


Figure 12

* 1. 离散余弦变换

在MATLAB中，可以使用dct2函数进行离散余弦变换。

代码：

I3=dct2(I1);

figure,imshow(log(abs(I3)),[]);

效果：

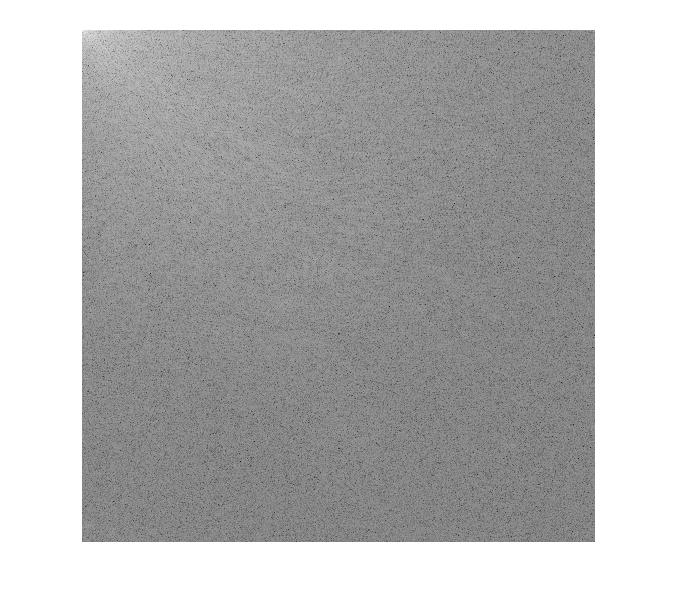


Figure 13

1. 图像增强

本章1-2小节代码保存在kyh\_Three1.m文件中，第3小节代码保存在kyh\_Three2.m文件中，第4小节代码保存在kyh\_Three3.m文件中，第5小节代码保存在kyh\_Three4.m文件中。

* 1. 直方图均衡化

对于灰度图像，可以使用直方图均衡化的方法使得原图像的灰度直方图修正为均匀的直方图。

代码如下：

I2=histeq(I1);

figure,imshow(I2);

figure,imhist(I2);

原图像为Figure 3，经过直方图均衡化后的图片为Figure 14。



Figure 14

原图像的灰度直方图为Figure 4，而经过修正后的灰度直方图为Figure 15：

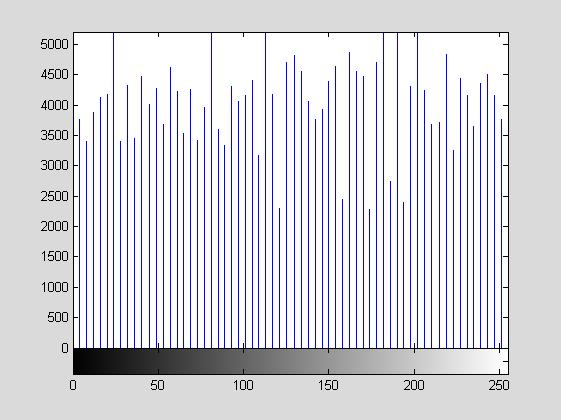


Figure 15

可以看出，与原图对比，灰度分布更加均匀。

* 1. 灰度变换

在MATLAB中，可以使用imadjust函数对图像进行线性变换。常用的为线性变换。使用的函数为imadjust(I, [low\_in, high\_in], [low\_out high\_out],gamma)，其中，in为输入的图片变化的灰度范围，out为输出的图片变化的灰度范围，均为[0 1]之间。也就是将low\_in和high\_in之间的值映射到low\_out和high\_out之间。gamma为线性值，若为1则为线性变换。

代码和效果如下：

I3=imadjust(I1,[0.3 0.7],[0.1 0.9],1);

figure,imshow(I3);

figure,imhist(I3);



Figure 16

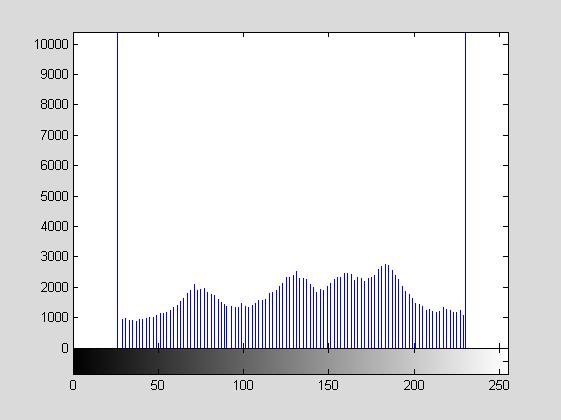


Figure 17

其中，突出了原图0.3-0.7之间的灰度元素。

* 1. 空间域平滑

图像平滑是为了抑制图像噪声，改善图像质量而进行的图片处理。在这里，笔者将首先介绍MATLAB中自带的均值滤波和中值滤波实现，之后设计自己的模版算子， 再次分别实现了均值滤波和中值滤波。

MATLAB中自带的均值滤波可以采用fspecial(‘params’,n)函数实现，其中，params代表的是滤波的种类，常用的有average（均值滤波），还有可以在空间域锐化中用到的高斯、拉普拉斯、prewitt、sobel算子等。n代表的是模版的在这里仅仅介绍均值滤波。

MATLAB中，均值滤波代码如下：

avgModel=fspecial('average',3);

Iavg=filter2(avgModel,I2)/255;

figure,imshow(Iavg);

由于均值滤波对于高斯噪声效果较好，故对高斯噪声的图（Figure 5）进行了图像平滑。效果如下（Figure 18）：



Figure 18

而笔者根据均值滤波原理，自己写了一段代码，如下：

n=3;

model(1:n,1:n)=1;

Iavg=I2;

for i=2:length(Iavg)-1

for j=2:length(Iavg(2,:))-1

x=I2(i-(n-1)/2:i+(n-1)/2,j-(n-1)/2:j+(n-1)/2);

xuint=uint8(model).\*uint8(x);

x\_avg=mean(xuint(:));

Iavg(i,j)=x\_avg;

end

end

figure,imshow(Iavg);

效果（Figure 19）：



Figure 19

可以发现，除了运行速度稍慢，效果与MATLAB自带的函数相同。

MATLAB中自带的中值滤波可以采用medfilt2(I,[n n])函数实现，其中，I代表的是需要处理的图片，n代表的是模版的大小。

MATLAB中，中值滤波代码如下：

Imid=medfilt2(I3,[3,3]);

figure,imshow(Imid);

由于中值滤波对于椒盐噪声效果较好，故对椒盐噪声的图（Figure 6）进行了图像平滑。效果如下（Figure 20）：



Figure 20

而笔者根据中值滤波原理，自己写了一段代码，如下：

n=3;

Imid=I3;

for i=2:length(I1)-1

for j=2:length(I1(2,:))-1

x=I3(i-(n-1)/2:i+(n-1)/2,j-(n-1)/2:j+(n-1)/2);%3\*3

x\_mid=median(x(:));

Imid(i,j)=x\_mid;

end

end

figure,imshow(Imid);

效果（Figure 21）：



Figure 21

可以发现，除了运行速度稍慢，效果与MATLAB自带的函数相同。

* 1. 空间域锐化

图像锐化有多种方法，最常用的锐化算子包括梯度算子、Roberts算子、Prewitt算子、Sobel算子以计算梯度的梯度锐化法，Laplacian算子等。接下来，笔者将一一介绍相关的算子。

笔者根据梯度算子的计算原理，编写了如下代码：

Igrad=I1;

for i=1:length(Igrad)-1

for j=1:length(Igrad(2,:))-1

x=I1(i,j+1)-I1(i,j);

y=I1(i+1,j)-I1(i,j);

grad=max(abs(x),abs(y));

Igrad(i,j)=grad;%¸³Öµ

end

end

figure,imshow(Igrad);

效果（Figure 22）：



Figure 22

MATLAB中自带了Prewitt算子和Sobel算子的函数，代码及效果如下：

model=fspecial('prewitt');

Iprewitt=filter2(model,I1);

figure,imshow(Iprewitt);



Figure 23

model=fspecial('sobel');

Isobel=filter2(model,I1);

figure,imshow(Isobel);



Figure 24

事实上，可以发现这几种算子之间不同的地方只是在于模版，因此笔者编写了一个可以自定义各种模版的代码，如下：

n=2

modelx=[-1 0;0 1];

modely=[0 -1;1 0];

Iend=I1;

Idouble=double(I1);

for i=1:length(Idouble)-5+n

for j=1:length(Idouble(2,:))-5+n

area=Idouble(i:i+n-1,j:j+n-1);

x=area.\*modelx;

y=area.\*modely;

grad=max(abs(sum(x(:))),abs((sum(y(:)))));

Iend(i,j)=grad;%¸³Öµ

end

end

figure,imshow(Iend);

可以显示出Roberts算子的效果：



Figure 25

与Prewitt算子、Sobel算子经过对比后发现效果相同，说明有效。

对于Laplacian算子，同样可以使用MATLAB中自带的函数，也可以进行自定义。自带函数使用方法如下：

model=fspecial('laplacian');

Ilaplacian=filter2(model,I1);

figure,imshow(Ilaplacian,[]);

效果：

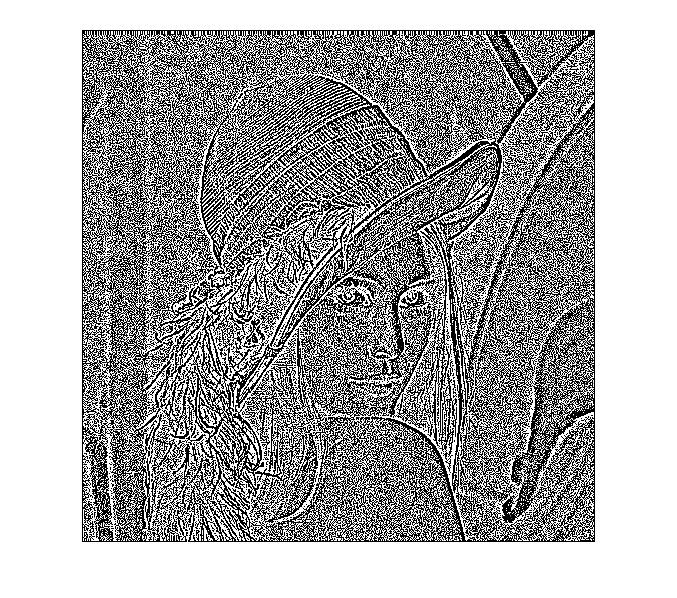


Figure 26

套用前面自定义模版后的代码如下：

n=3;

model=[0 -1 0;-1 5 -1;0 -1 0];

Iend=I1;

Idouble=double(I1);

for i=1:length(Idouble)-5+n

for j=1:length(Idouble(2,:))-5+n

area=Idouble(i:i+n-1,j:j+n-1);

x=area.\*model;

grad=sum(x(:));

Iend(i,j)=grad;

end

end

figure,imshow(Iend);

效果：



Figure 27

与教科书上的效果相同。

* 1. 频率域图像增强

频率域增强包括高通滤波器和低通滤波器，主要是使用傅立叶变换和逆变换对图像进行处理。在MATLAB中没有直接的滤波器函数，但是可以结合原理与MATLAB中的数学公式，构造各种滤波器。在本文中，笔者分别构造了理想、Butterworth、指数的高通和低通滤波器，并一一进行了实验。

首先需要提前设定截止频率D0和阶数n。之后就可以套用具体的公式进行计算了。部分代码如下：

n=6;%滤波器的阶数

D0=80;%滤波器的截止频率

[u,v]=meshgrid(-M/2:(M/2-1),-N/2:(N/2-1));%产生离散数据

D=sqrt(u.^2+v.^2);

%各种不同的滤波器

H1=double(D<=D0);%理想低通滤波器

H2=1./(1+(D/D0).^(2\*n));%Butterworth低通滤波器

H3=exp(-(D/D0).^n);%指数低通滤波器

H4=double(D>=D0);%理想高通滤波器

H5=1./(1+(D0./D).^(2\*n));%Butterworth高通滤波器

H6=exp(-(D0./D).^n);%指数高通滤波器

之后，可以使用各种不同的滤波器对图片进行处理。效果如下：



Figure 28



Figure 29



Figure 30



Figure 31



Figure 32



Figure 33

通过以上一系列的图片可以看出，低通滤波器可以抑制图像噪声，改善图像质量，高通滤波器可以突出图像的边界。

1. 图像分割

本章所有代码均保存在kyh\_Four.m文件中。

* 1. 边缘检测算子

对于边缘检测，与空间锐化的模版相似，在这里就不再自定义模版，而是直接使用MATLAB中已经提供了的edge系列函数。

Edge函数的使用方法如下：edge(I, params)，其中params可以为’sobel’, ‘prewitt’, ‘roberts’, ‘canny’。

代码和效果分别如下：

Sobel算子：

figure,imshow(edge(I1,'sobel'));

效果：



Figure 34

Prewitt算子：

figure,imshow(edge(I1,'prewitt'));

效果：



Figure 35

Prewitt和Sobel算子效果区别不大。

Roberts算子：

figure,imshow(edge(I1,'roberts'));

效果：



Figure 36

可以看出Roberts较Prewitt和Sobel效果有所提升，但是边缘仍然有损失。

Canny算子：

figure,imshow(edge(I1,'canny'));

效果：



Figure 37

可以看出Canny提取出了最多的边缘，但可能会造成一定的冗余。

1. 二值图像处理与形状分析

本章所有代码均保存在kyh\_Five.m文件中。

* 1. 二值化

如果一个图片中的像素只有0和1两个值，则这个图片为二值化图片。在MATLAB中可以通过已有的函数自动生成阈值，对图像进行二值化处理。代码如下：

level = graythresh(I1);

I2 = im2bw(I1,level);

imshow(I2)

效果：



Figure 38

可以看出，全图仅有黑白两种颜色，其中，黑为0，白为1.

* 1. 膨胀

在MATLAB中，有现成的函数可以调用，进行二值图像膨胀的操作。使用的函数为imdilate(I, shape)，其中shape是膨胀的形状（结构元），可以使用矩阵进行定义。代码如下：

model=[0 1 0;1 1 1;0 1 0];

Idilate=imdilate(I2,model);

figure,imshow(Idilate);

效果：



Figure 39

注意原二值化的图与膨胀后图在帽子附近细线的区别，可以很明显地看出，细线变粗；全图变白，这是因为经过膨胀后，值为1（白色）的区域增多了。

* 1. 腐蚀

与膨胀相类似，MATLAB中有现成的函数imerode(I, shape)，其中shape为腐蚀的形状（结构元）。

代码如下：

Ierode=imerode(I2,model);

figure,imshow(Ierode);

这里使用与膨胀相同的模版进行腐蚀操作。

效果：



Figure 40

可以看到与原图相比，黑色区域增大，帽子附近白色条纹消失。

* 1. 开闭运算

在图像处理应用中，膨胀和腐蚀更多地以各种组合来应用。先腐蚀再膨胀称为开运算，先膨胀再腐蚀成为闭运算。MATLAB中对于两种操作均有函数可以进行。

开运算可以使用imopen(I, shape)进行操作，闭运算可以使用imclose(I, shape)进行操作，其使用方式与膨胀和腐蚀类似。代码和效果分别如下：

开运算：

Iopen=imopen(I2,model);

figure,imshow(Iopen);



Figure 41

闭运算：

Iclose=imclose(I2,model);

figure,imshow(Iclose);



Figure 42

1. 总结

本文总结了多种数字图像处理方法在MATLAB中的实现。从最基本的概念讲起，包括了MATLAB中对于图像的基本操作，图像变换，图像增强，图像分割，二值图像处理与形状分析等。这是对于笔者课上概念学习的一种补充和实践。通过代码的实现、图片前后效果的对比，加深了一些概念的理解，也找到了一些原来没有注意到的细节。

为了完成本文中各种方法的实现，笔者也上网参阅了一些技术博客，查阅了一些相关书籍，以教科书为蓝本，结合《数字图像处理的MATLAB实现（第2版）》（冈萨雷斯著），《MATLAB数字图像处理实战》（赵小川著），《MATLAB R2015a数字图像处理》（丁伟雄著），以及MATLAB自带的帮助进行了学习。一些函数MATLAB中有现成的，笔者学会了调用的方法；部分函数虽然有现成函数可以进行调用，但是为了加深对于细节的理解，笔者又重新进行了实现并和MATLAB中已有的函数进行了对比；一些图像处理的方法在MATLAB中也没有实现的现成函数，笔者就根据原理进行了实现。

虽然完成了众多图像处理方法的实现，但仍然有一些不足。首先，由于时间所迫，并没能完成所有教科书中的方法实现，只是选择了一些较为基本和容易实现的方法进行了实践。其次，对于一些函数仍然不能完全掌握全部用法，只能学会基本的用法。最后，由于时间不足，本文中对于各种图像处理的方法仅仅介绍了实现的代码，对于结果也仅仅是放上了效果图供参考，并没有进行具体的分析，总体而言叙述较少，对于结果的分析有所欠缺。

总之，通过这样一次实践，加深了笔者对于数字图像处理的理解，也锻炼了动手实践能力，不失为对于课程的一个很好的补充。