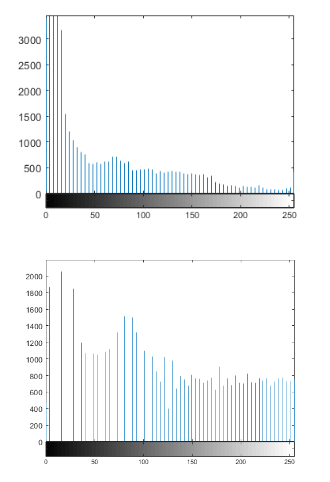
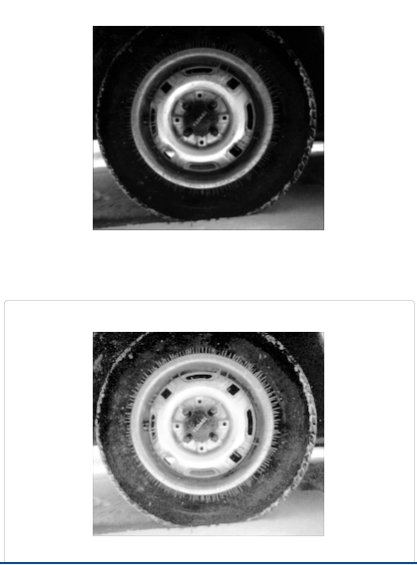
**集美大学计算机工程学院实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称： 数字信号与图像处理** | **班级：计算2114** | **实验成绩：** |
| **实验编号：004** | **姓名： 庄佳强** | **实验日期：12.1** |
| **实验名称：**图像处理基础与图像变换 | **学号：202121331104** | **实验地点：陆大206** |

1. **目的**（本次实验所涉及并要求掌握的知识点）

通过本实验加深对数字图像增强操作的理解，熟悉MATLAB中的有关函数；了解直方图均衡化和卷积滤波的原理；熟悉低通和高通滤波模板的构造方法。

1. **实验内容与设计思想**（设计思路、主要代码结构、主要



**原始图像 vs. 直方图均衡化后的图像：**

直方图均衡化可以增强图像的对比度。对比度提高后，图像中的细节更为突出。

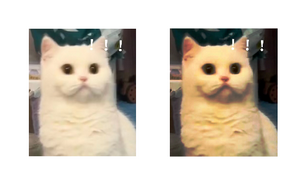
**原始图像的直方图 vs. 直方图均衡化后图像的直方图：**

直方图均衡化会使得直方图更均匀分布，使得图像的灰度级别更广泛地覆盖整个亮度范围。

输入代码，比较I、J两幅图像的差别，写出其原因

将程序中的图像换成其他图像，验证你的说法是否正确

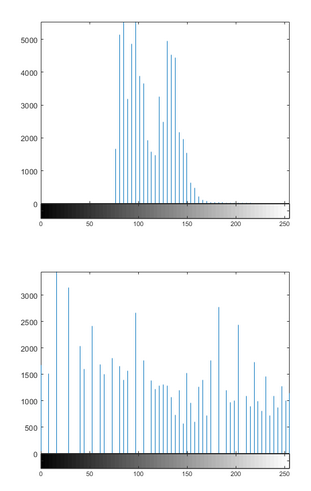
画出I、J的直方图，写出两者的差别



**视觉效果：** 直方图均衡化通常增强了图像的对比度，使得细节更为明显。

**原因：** 直方图均衡化通过重新分配图像的灰度级别，使得图像的亮度范围更加均匀分布，从而提高了图像的对比度。

通过猫咪图也验证了这个原因，猫咪的对比度被提高了。



通过比较直方图，你可以看到直方图均衡化后的图像在灰度级别上更加均匀，这是提高对比度的一个表现。

窗体底端



**模糊效果：** 平均滤波器的作用是对图像进行平滑处理，因此经过处理后的图像在视觉上会显得更加模糊。

**细节损失：** 平均滤波器的使用导致图像中的细节被模糊掉，特别是那些小尺度的细节。

将h分别修改下列的值，观察实验结果，写出产生此现象的原因

h=ones(3,3)/9;



由于h是较小的平均滤波器，图像经过处理后仍然保留了一些细节，但整体进行了轻微的平滑。

a=[1 1 1

1 2 1

1 1 1]

h=a/10;



由于h是一个权重滤波器，对图像进行了更强烈的平滑处理。这样的滤波器在中心像素周围引入了更大的权重，使得图像更加模糊。

构造一个低通滤波模板，并加以实现

**原理：**

使用函数fft2将空间图像进行二维傅里叶变换后，

转化到频域上，我们可以得到这个图像每个像素的相位和幅度。

而幅度值则主要代表着能量的大小，也就是每一个频率上的能量大小。

由于本题中我们要构造低通滤波器，即去除图像的高频部分，

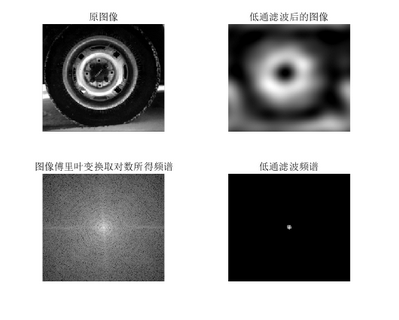
公式如下：

*H*(*u*,*v*)= 1    *D*(*u*,*v*)≤*D*0,

0    *D*(*u*,*v*)>*D*0

其中D0为自己定义的低频滤波范围

完成滤波之后进行反傅里叶变化还原。



构造2—3个高通滤波模板，并加以实现，和低通滤波的效果加以对比，写出高通滤波和低通滤波的差别，并写出低通滤波模板、高通滤波模板构造的规律。

第一：

在高频滤波中，只需要把低频滤波截至频率换成高频滤波，公式换成如下：

*H*(*u*,*v*)= 1    *D*(*u*,*v*)>*D*0,

0    *D*(*u*,*v*)≤*D*0

其中D0为自己定义的高频滤波范围

就可以变成高频滤波模板

第二:

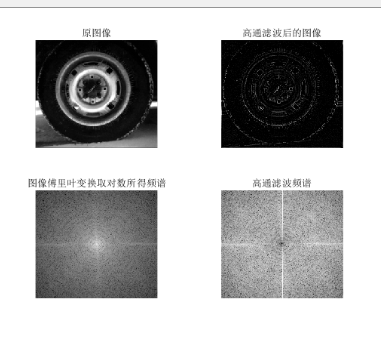
*H(u,v)=*

网上找到比较复杂的高通滤波器

对应的函数修改为对应的就行。

d = ((u-center\_u)^2+(v-center\_v)^2);

H(u,v) = 1 - exp(-(d)/(2\*(D0^2)));



1. 低通滤波:

效果： 允许低频信号通过，削弱高频信号。应用： 常用于去除图像中的高频噪声，平滑图像，或者在图像处理中的下采样操作。

低通滤波模板构造规律：

圆形模板： 低通滤波器通常采用圆形的频域模板，中心部分为高振幅的低频成分，边缘逐渐减小。这可以通过定义一个中心点，并根据距离中心的距离来调整振幅。

2. 高通滤波:

效果： 允许高频信号通过，抑制低频信号。

高通滤波模板构造规律：

中心被抑制： 高通滤波器的频域模板通常将低频部分置零，而高频部分保留。这可以通过定义一个中心点，并根据距离中心的距离来调整振幅，但是这次是低频部分振幅置零。

1. 使用C／C++编程实现直方图均衡化函数（histeq）和卷积滤波函数（imfilter）

histeq函数：

先计算原图片的直方图，遍历图片，对使用的像素值进行统计。接着归一化(/rows\*lows)直方图以获得概率，计算直方图的累积分布函数（CDF），对图像应用直方图均衡。

imfilter函数：

获取图像通道数。遍历输入图像的内部像素（排除边缘），以应用3x3均值滤波，内部循环遍历当前像素的通道值，应用3x3均值滤波器的公式。

\*output++ = saturate\_cast<uchar>

(floor((current[i] + current[i - nChannels] + current[i + nChannels]

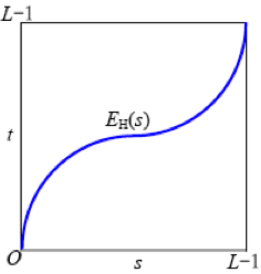
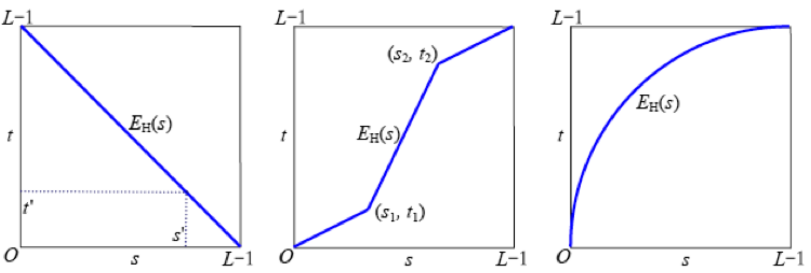
+ precious[i] + precious[i - nChannels] + precious[i + nChannels]

+ next[i] + next[i - nChannels] + next[i + nChannels]) / 9.0));

设置Result矩阵的边缘像素为零，以确保输出图像的边缘不受滤波器影响。



（4）使用C／C++编程实现灰度映射



灰度映射是指对图像的灰度值进行转换的过程。在灰度映射中，每个像素的原始灰度值通过一个特定的映射函数进行转换，得到新的灰度值。

先遍历图像像素，获得当前像素的灰度值pixelValue ，使用灰度映射公式函数alpha \* pixelValue + beta，再更新函数就行了。



**三、实验使用环境**（本次实验所使用的平台和相关软件）

Matlab,C++。

1. **源代码附件**（完整的程序源代码，**注意排版紧凑，如果代码较长，字体、字间距选择小一点的**）

clear;

I = imread('tire.tif');

J = histeq(I);

imshow(I)

figure, imshow(J)

imhist(I,64)

figure; imhist(J,64)

I=imread('pout.tif');

J=histeq(I);

subplot(1,2,1);imshow(I);

subplot(1,2,2);imshow(J);

figure;

imhist(I,64)

figure; imhist(J,64)

I=imread('demo.jpg')

J=histeq(I);

subplot(1,2,1);imshow(I);

subplot(1,2,2);imshow(J);

clear;

I=imread('demo.jpg');

%I=rgb2gray(I);

h=ones(5,5)/25;

J=imfilter(I,h);

figure(1);

imshow(I);

figure(2);

imshow(J);

h=ones(3,3)/9;

J=imfilter(I,h);

figure(3);

imshow(I);

figure(4);

imshow(J);

a=[1 1 1

1 2 1

1 1 1]

h=a/10;

J=imfilter(I,h);

figure(5);

imshow(I);

figure(6);

imshow(J);

% Parameters:

% H: 低通滤波器

% u\_max: 图像第一维的长度

% v\_max: 图像第二维的长度

% s:

%D0: 低频滤波截止频率

function [H] = LowFil(u\_max, v\_max, s,D0)

% 计算图像中心点坐标以及截止频率D0

center\_u = floor(u\_max/2);

center\_v = floor(v\_max/2);

% 初始化

H=zeros(u\_max, v\_max);

for u=1:u\_max

for v=1:v\_max

% (u, v)和中心点间的距离

dist = (u - center\_u)^2 + (v - center\_v)^2;

% 判断是否要截止

if(dist<=D0)

h = 1;

else

h = 0;

end

H(u, v) = h \* s(u, v);

end

end

% Parameters:

% H: 高通滤波器

% u\_max: 图像第一维的长度

% v\_max: 图像第二维的长度

% s:

function [H] = highFil(u\_max, v\_max, s)

% 计算图像中心点坐标以及截止频率D0

center\_u = floor(u\_max/2);

center\_v = floor(v\_max/2);

D0 = min(center\_u,center\_v)/12;

D0 = D0^2;

% 初始化

H=zeros(u\_max, v\_max);

for u=1:u\_max

for v=1:v\_max

d = ((u-center\_u)^2+(v-center\_v)^2);

H(u,v) = 1 - exp(-(d)/(2\*(D0^2)));

end

end

H = H .\* s;

clc;

clear;

img = imread('tire.tif');

subplot(2,2,1),imshow(img);

title('原图像');

% 将低频移动到图像的中心

s = fftshift(fft2(img)); % 对图像进行傅里叶变换并移动低频到中心

subplot(2,2,3),imshow(log(abs(s)),[]);

title('图像傅里叶变换取对数所得频谱');

%低频滤波截止频率

D0=20

[a,b] = size(img); % 获取图像的大小

low\_filter = LowFil(a,b,s,D0); % 使用自定义的低通滤波器函数LowFil进行滤波

% low\_filter就是低通滤波器

subplot(2,2,4),imshow(log(abs(low\_filter)),[]);

title('低通滤波频谱');

new\_img = uint8(real(ifft2(ifftshift(low\_filter)))); % 对滤波后的频谱进行逆傅里叶变换，并取实部得到图像

subplot(2,2,2),imshow(new\_img,[]); %

title('低通滤波后的图像');

clc;

clear;

img = imread('tire.tif');

subplot(2,2,1),imshow(img);

title('原图像');

% 将高频移动到图像的中心

s = fftshift(fft2(img)); % 对图像进行傅里叶变换并移动高频到中心

subplot(2,2,3),imshow(log(abs(s)),[]);

title('图像傅里叶变换取对数所得频谱');

[a,b] = size(img); % 获取图像的大小

high\_filter = highFil(a,b,s); % 使用自定义的高通滤波器函数highFil进行滤波

% high\_filter就是通滤波器

subplot(2,2,4),imshow(log(abs(high\_filter)),[]);

title('高通滤波频谱');

new\_img = uint8(real(ifft2(ifftshift(high\_filter)))); % 对滤波后的频谱进行逆傅里叶变换，并取实部得到图像

subplot(2,2,2),imshow(new\_img,[]); %

title('高通滤波后的图像');

}