****

**实验报告**

**课程名称：数字图像处理**

**实验项目名称：三种低通滤波器GUI集成实验**

**指导教师：徐端全**

**专业班级：电信卓越2101班**

**学生学号：U202114202**

**学生姓名: 黄煜翀**

**成绩/等级（批阅教师）:**

**实验执行日期：2024年 4 月 7 日**

# 实验目的

本实验通过MatlabGUI界面对三种低通滤波器进行集成实现，以加深对于理想低通滤波器（ILPF）、布特沃茨低通滤波器（BLPF）、高斯低通滤波器的（GLPF）原理以及适用场景。同时，通过在不同参数设定下对滤波效果进行对比分析，体会不同滤波器的滤波效果以及参数设定的技巧与经验。进一步熟练掌握matlab编程工具中GUI界面的设计，通过图形化界面形式为用户提供方便快捷的使用体验。

# 三种低通滤波器的原理

## 图示 描述已自动生成频域滤波原理

由于空间滤波器仍在图像的像素级别上进行处理，具有较大的局限性。很多图像处理问题与频域有重要的关联，例如噪声问题，边缘问题等。由于频率成分与图像的外在表现具有一定的对应关系，我们引入频域滤波来处理与频率有关的图像处理问题。同时，频率域处理对于试验、迅速而全面地控制滤波器参数是一个理想工具。

标志上写着字

描述已自动生成 我们对频域滤波的一般原理进行解释。首先，我们将图像通过傅里叶变换变换到频域中，这是一个由各频率成分的幅值组成的幅度谱。我们在此频谱图中可以针对性的滤除或者保留所需要的频率成分，舍弃不需要的频率成分。由于傅里叶变换后低频成分被变换到四个角上，为了方便进行观察和处理，我们做频率中心化将低频分量集中在图像的中央，剩余的频率成分分布在图像的外围。此后，我们构造滤波函数H(u,v)，这也是决定滤波效果的关键函数。将滤波函数与原图像傅里叶变换相乘实现滤波，也即原图像与滤波函数的冲激响应在空域卷积。

得到结果后，做去中心化和傅里叶反变换即可获得最终滤波结果。

## 理想低通滤波器原理（ILPF）

卡通人物

中度可信度描述已自动生成图标

描述已自动生成

因为图像由一个个像素点组成，所以图像直方图均衡化是通过离散形式的累积分布函数求解的，直方图均衡化过程中，映射方法是：



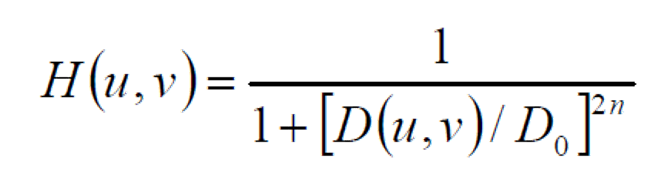
其中，指当前灰度级经过累积分布函数映射后的值，n是图像中像素的总和，是当前灰度级的像素个数，L是图像中的灰度级总数。此时为[0,1]之间的一组数据，我们将其进行范围的规范化，使其成为[0,L-1]的一组数据：



即获得了从原始灰度到变换后的灰度的最终映射关系。

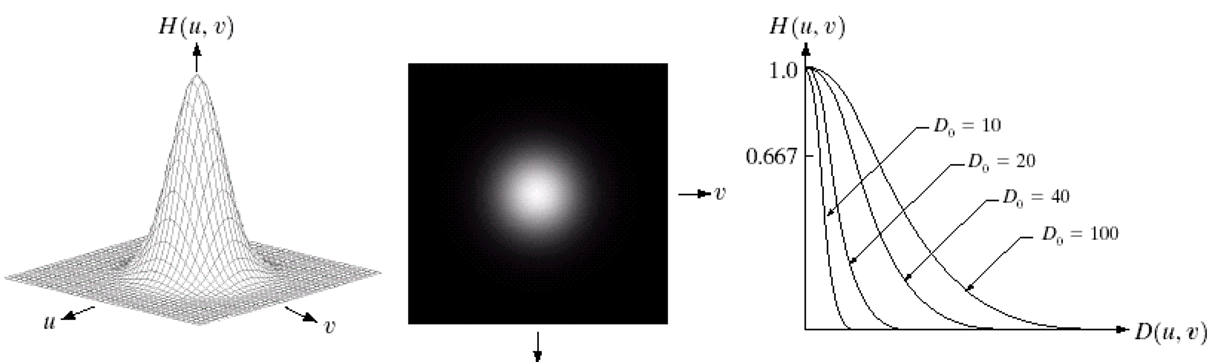
## 巴特沃茨低通滤波器原理（BLPF）

巴特沃茨（Butterworth）滤波器是一种具有最大平坦幅度响应低通滤波器。N阶巴特沃思低通滤波器定义如下：

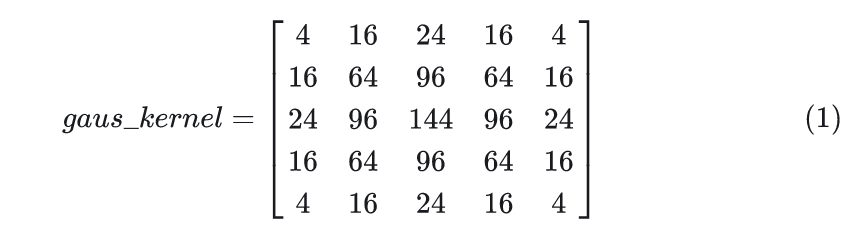


其中D0为截止频率距原点的距离，D(u,v)是点(u,v)距原点的距离。不同于ILPF，BLPF变换函数在通带与被滤除的频率之间没有明显的截断。

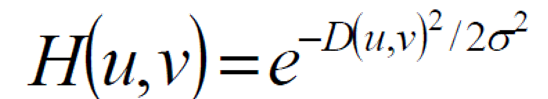
## 高斯低通滤波器原理（GLPF）



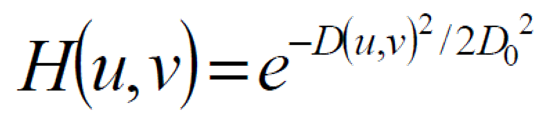
高斯滤波器的本质是一个二维低通系统，它通常是以一个矩阵的形式出现，比如式（1）这个5×5高斯核（又称为kernel），用它对图像做卷积，图像的高频分量会被打到低频，最终呈现的效果就是图像变模糊了。



二维高斯低通滤波器(GLPF)定义如下：



其中，D(u,v)是点(u,v)距中心点的距离，δ表示高斯曲线扩展的程度。用D0代替δ，将滤波器表示为：



# 编程环境介绍

MATLAB 2021

# 程序结构及流程讲解

## GUI界面控制（GUI.m）

clc

clear all;

picture=imread('huafen.bmp'); *%读取图像*

picture = rgb2gray(picture); *%将RGB彩色图像转换为灰度图*

figure

subplot(2,2,1)

imshow(picture)

title('原来的图象'); *%显示原始图象*

after = hist\_1(picture);

subplot(2,2,2)

imshow(after)

title('直方图均衡化后的图象'); *%显示均衡化后的图象*

subplot(2,2,3)

imhist(picture)

title('原始图像直方图')

subplot(2,2,4)

imhist(after)

title('均衡化后图像直方图')

主函数包含读取图像、将RGB彩色图像转变为灰度图像、调用自定义hist\_1直方图均衡化函数、展示原始图像和均衡化后图像、展示原始图像直方图和均衡化后图像直方图等操作。其中核心为hist\_1自定义函数的实现，将在后续部分中重点讲解。

## 三个低通滤波器的代码实现

* + 1. 理想低通滤波器实现

I = imread(pic);

I=rgb2gray(I);

% 函数fft2()用于计算二维傅立叶变换

% 函数fftshift()是对函数fft2()作傅里叶变换后得到的频谱进行平移,将变换后的图像频谱中心从矩阵的原点移到矩阵的中心

% 作二维傅里叶变换前一定要用函数im2double()把原始图像的数据类型由uint8转化为double类型

% 否则会因为unit8数据类型只能表示0-255的整数而出现数据截断,进而出现错误结果

s=fftshift(fft2(im2double(I)));

[a,b]=size(s);

a0=round(a/2);

b0=round(b/2);

for i=1:a %双重for循环计算频率点(i,j)与频域中心的距离D(i,j)=sqrt((i-round(a/2)^2+(j-round(b/2)^2))

for j=1:b

distance=sqrt((i-a0)^2+(j-b0)^2);

if distance<=d0 % 根据理想低通滤波器产生公式,当D(i,j)<=D0,置为1

h=1;

else

h=0; % 根据理想低通滤波器产生公式,当D(i,j)>D0,置为0

end

s(i,j)=h\*s(i,j);% 频域图像乘以滤波器的系数

end

end

% real函数取元素的实部

ProcessedPic=real(ifft2(ifftshift(s)));% 最后进行二维傅里叶反变换转换为时域图像

end

首先，将图片转换成灰度图像，并调用fft2()函数和fftshift()函数进行傅里叶变换，得到频域图像s。之后，使用双重循环对所有像素进行遍历，对于D(i,j)>D0的频域点全部置零，其余点则不变。对处理后的频域图取实部，做傅里叶反变换，即可得到理想低通图像滤波后的图片。

* + 1. 巴特沃茨低通滤波器实现

function ProcessedPic = BLPF(pic,d0,n)

I = imread(pic);

I=rgb2gray(I);

% 函数fft2()用于计算二维傅立叶变换

% 函数fftshift()是对函数fft2()作傅里叶变换后得到的频谱进行平移,将变换后的图像频谱中心从矩阵的原点移到矩阵的中心

% 作二维傅里叶变换前一定要用函数im2double()把原始图像的数据类型由uint8转化为double类型

% 否则会因为unit8数据类型只能表示0-255的整数，出现数据截断,进而出现错误结果

s=fftshift(fft2(im2double(I)));

[N1,N2]=size(s);%求二维傅里叶变换后图像大小

n1=round(N1/2);

n2=round(N2/2);

for i=1:N1 %双重for循环计算频(i,j)与频域中心的距离D(i,j)=sqrt((i-round(N1/2)^2+(j-round(N2/2)^2))

for j=1:N2

distance=sqrt((i-n1)^2+(j-n2)^2);

if distance==0

h=0;

else

h=1/(1+(distance/d0)^(2\*n));% 根据巴特沃斯低通滤波器公式为1/(1+[D(i,j)/D0]^2n)

end

s(i,j)=h\*s(i,j);% 频域图像乘以滤波器的系数

end

end

% real函数取元素的实部

ProcessedPic=real(ifft2(ifftshift(s)));% ????进行二维傅里叶反变换转换为时域图??end

将图片转换成灰度图像，并得到频域图像。再通过一个双重for循环遍历所有像素，得到它们距离频域中心的距离。之后使用巴特沃斯低通滤波器公式1/(1+[D(i,j)/D0]^2n)与原像素值相乘，可得处理后的频域图。最后进行傅里叶反变换即得处理后的图像。

* + 1. 高斯低通滤波器实现

function ProcessedPic = GLPF(pic,d0)

I = imread(pic);

I=rgb2gray(I);

% 函数fft2()用于计算二维傅立叶变换

% 函数fftshift()是对函数fft2()作傅里叶变换后得到的频谱进行平移,将变换后的图像频谱中心从矩阵的原点移到矩阵的中心

% 作二维傅里叶变换前一定要用函数im2double()把原始图像的数据类型由uint8转化为double类型

% 否则会因为unit8数据类型只能表示0-255的整数而出现数据截断,进而出现错误结果

s=fftshift(fft2(im2double(I)));

[a,b]=size(s);

a0=round(a/2);

b0=round(b/2);

for i=1:a

for j=1:b

distance=sqrt((i-a0)^2+(j-b0)^2); % 根据高斯低通滤波器公式H(u,v)=e^-[D^2(u,v)/2\*D0^2]

h=exp(-(distance\*distance)/(2\*(d0^2))); % exp表示以e为底的指数函数

s(i,j)=h\*s(i,j);% 频域图像乘以滤波器的系数

end

end

ProcessedPic=real(ifft2(ifftshift(s)));% 最后进行二维傅里叶反变换转换为时域图像

end

将图片转换成灰度图像，并得到频域图像。通过一个双重for循环遍历所有像素，得到它们距离频域中心的距离distance=sqrt((i-a0)^2+(j-b0)^2);再用公式求得滤波器系数h=exp(-(distance\*distance)/(2\*(d0^2)));与原像素值相乘可得处理后的像素值。将得到的频域图像进行傅里叶反变换，即可得到高斯低通滤波后的图片。

# 实验结果