实验五：频域滤波

学号：SA22225286 姓名：孟寅磊 日期：20221101

实验内容

> 1. 灰度图像的DFT和IDFT
>
>   具体内容：利用OpenCV提供的`dft`函数对图像进行DFT和IDFT变换
>
> 2. 利用理想高通和低通滤波器对灰度图像进行频域滤波
>
>   具体内容：利用`dft`函数实现DFT，在频域上利用理想高通和低通滤波器进行滤波，并把滤波后的图像显示在屏幕上(观察振铃现象)，要求截止频率可输入。
>
> 3. 利用布特沃斯高通和低通滤波器对灰度图像进行频域滤波
>
>   具体内容：利用`dft`函数实现DFT，在频域上利用布特沃斯高通和低通滤波器进行滤波，并把滤波后的图像显示在屏幕上(观察振铃现象)，要求截止频率可输入。

实验原理

使用如下的傅里叶变换可以将图像从空间域转换到频率域

$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)e^{-j2\pi(ux/M+vy/N)} \tag{5.1}$$

其中$f(x,y)$是大小为$M\times N$的输入图像。

频率域滤波的步骤是，首先修改一幅图像的傅里叶变换，然后计算其反变换，得到处理后的结果的空间域表示。所以频率域滤波的基本公式为

$$g(x,y)=Real\{\Im^{-1}[H(u,v)F(u,v)]\} \tag{5.2}$$

其中，$\Im^{-1}$是IDFT，$F(u,v)$是输入图像$f(x,y)$的DFT，$H(u,v)$是滤波器传递函数，$g(x,y)$是滤波后的输出图像。我们所使用的有理想低通、理想高通、布特沃斯低通、布特沃斯高通滤波器。

在以原点为中心的一个圆内无衰减地通过所有频率，而在这个圆外“截止”所有频率的二维低通滤波器，称为理想低通滤波器($ILPF$)，它由下面的传递函数规定：

$$H(u,v)=\begin{cases}1, & D(u,v)\le D_0\\ 0, & D(u,v)>D_0\end{cases} \tag{5.3}$$

其中，D_0称为截止频率，$D(u,v)$是频率域中点(u,v)到$P\times Q$频率矩形中心的距离。理想低通滤波器可以用来平滑图像。像在空间域中使用核那样，在频率域中用1减去低通滤波器传递函数，会得到对应的高通滤波器($IHPF$)传递函数：

$$H(u,v)=\begin{cases}0, & D(u,v)\le D_0\\ 1, & D(u,v)>D_0\end{cases} \tag{5.4}$$

截止频率位于距频率矩形中心D_0处的n阶布特沃斯低通滤波器($BLPF$)的传递函数定义为：

$$H(u,v)=\frac{1}{1+[D(u,v)/D_0]^{2n}} \tag{5.5}$$

这个函数可以用较高的n值来逼近$ILPF$的特性，且振铃效应要比$ILPF$小得多。由该式得到的布特沃斯高通滤波器($BHPF$)的传递函数为：

$$H(u,v) = \frac{1}{1 + [D_0/D(u,v)]^{2n}} \tag{5.6}$$

使用OpenCV实现的用于计算这4种传递函数的函数如下：

```
Mat transfer_func(const int P, const int Q,
                  int type, const int D0, const int n) {
    const int M = P / 2;
    const int N = Q / 2;
    const double D0_2 = D0 * D0;
    Mat H(Size(P, Q), CV_32F);
    if (type == 0) { // LP
        for (int i = 0; i < P; i++)
        for (int j = 0; j < Q; j++)
            H.at<float>(i, j) =
            (n == -1) ?
            (((i-M)*(i-M) + (j-N)*(j-N) <= D0_2) ? 1 : 0) :  // ILPF
            1.0/(1+pow(((i-M)*(i-M)+(j-N)*(j-N))/D0_2,2*n)); // BLPF
    } else {         // HP
        for (int i = 0; i < P; i++)
        for (int j = 0; j < Q; j++)
            H.at<float>(i, j) =
            (n == -1) ?
            (((i-M)*(i-M) + (j-N)*(j-N) <= D0_2) ? 0 : 1) :  // IHPF
            1.0/(1+pow(D0_2/((i-M)*(i-M)+(j-N)*(j-N)),2*n)); // BHPF
    }
    return H;
}
```

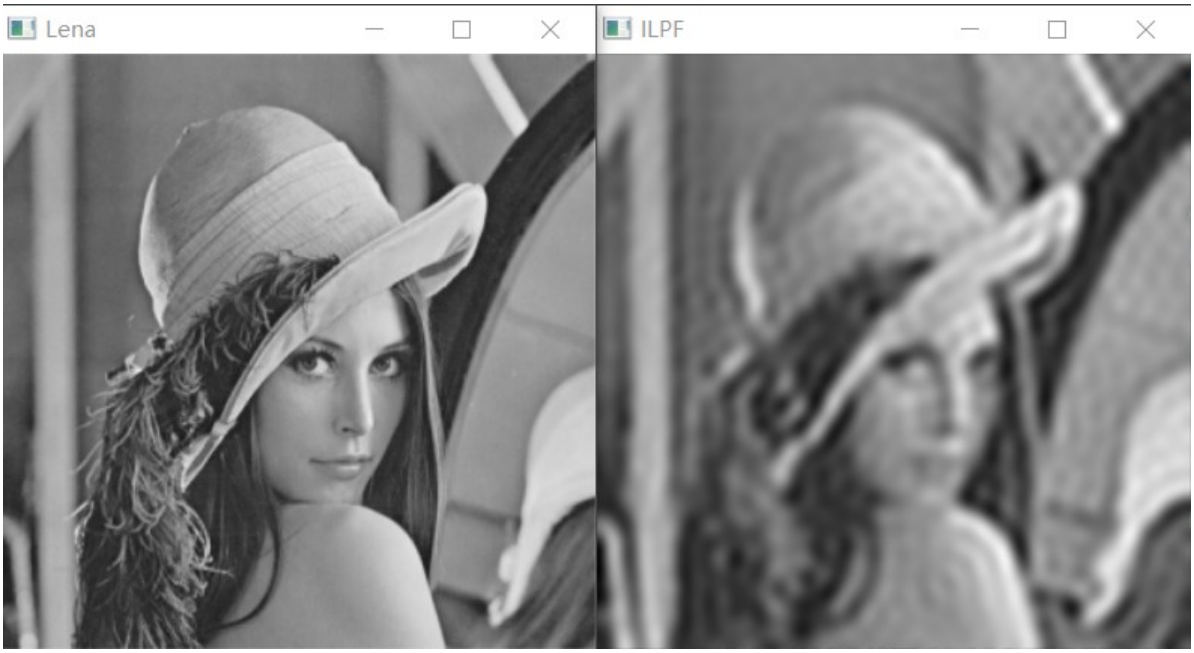
然后实现用于频率域滤波的函数：

```
/**
 * @brief
 * 使用《数字图像处理(第四版)》第182-183页所描述的算法实现的频率域滤波函数
 * @param    f 输入图像
 * @param type 0表示低通滤波器，非0表示高通滤波器
 * @param   D0 截止频率
 * @param    n 布特沃斯滤波器的阶数，如果要使用理想高通/低通滤波器，请将该值设为-1
 */
void frequency_domain_filter(Mat& f, const int type,
                             const int D0, const int n) {
    const int M = f.rows;
    const int N = f.cols;
    // 1. 计算填充后的图像的尺寸
    const int P = 2 * M;
    const int Q = 2 * N;
    // 2. 使用镜像填充形成大小为P * Q的填充后的图像fp(x,y)
    Mat fp;
    copyMakeBorder(f, fp, 0, M, 0, N, BORDER_REFLECT);
    // 3. 将fp(x,y)乘以(-1)^(x+y)，使傅里叶变换位于P * Q大小的频率矩形的中心
    for (int i = 0; i < P; i++)
        for (int j = 0; j < Q; j++)
            if ((i + j) & 1)
                fp.at<uchar>(i, j) = -fp.at<uchar>(i, j);
    // 4. 计算步骤3得到的图像的DFT
```

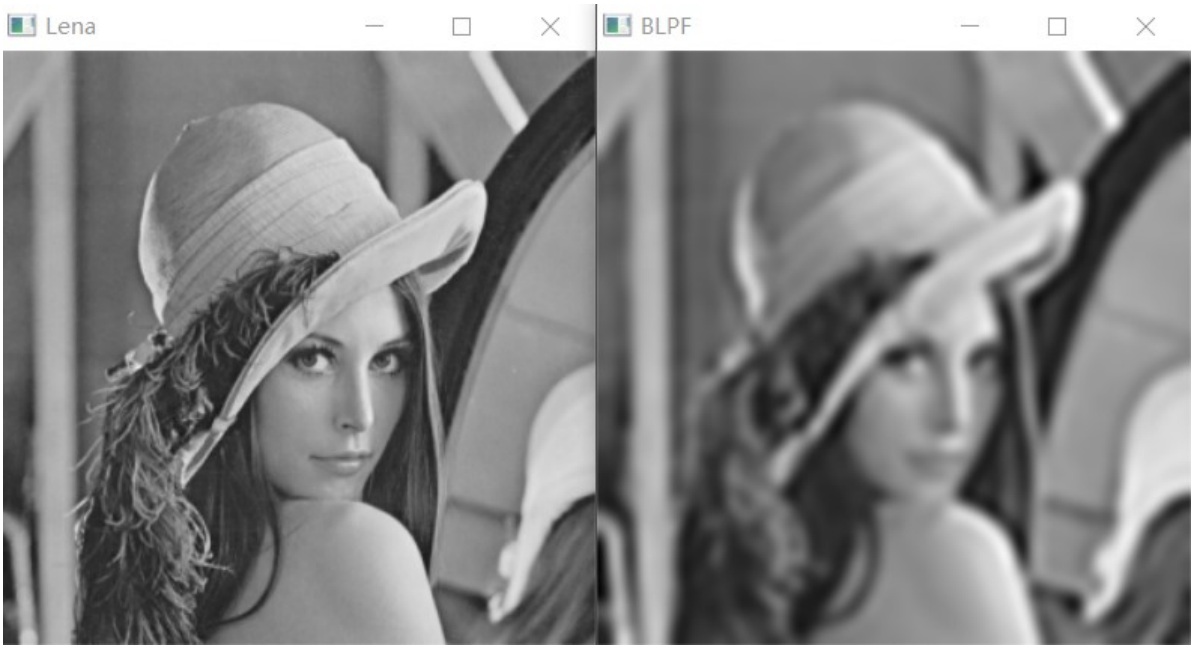
```
    Mat real_imag[] = {Mat_<float>(fp), Mat::zeros(fp.size(), CV_32F)};
    Mat F;
    merge(real_imag, 2, F);
    dft(F, F);
    split(F, real_imag);
    // 5. 构建滤波器传递函数
    Mat H = transfer_func(P, Q, type, D0, n);
    // 6. 采用对应像素相乘得到G
    for (int i = 0; i < P; i++) {
        for (int j = 0; j < Q; j++) {
            real_imag[0].at<float>(i, j) *= H.at<float>(i, j);
            real_imag[1].at<float>(i, j) *= H.at<float>(i, j);
        }
    }
    // 7. 计算G的IDFT
    Mat G;
    merge(real_imag, 2, G);
    idft(G, G);
    split(G, real_imag);
    // 8. 提取G的左上角部分的实部，去除寄生复数项
    Mat g(Size(M, N), CV_32F);
    for (int i = 0; i < M; i++)
        for (int j = 0; j < N; j++)
            g.at<float>(i, j) = real_imag[0].at<float>(i, j) * pow(-1, i+j);
    // 9. 归一化处理
    normalize(g, g, 0, 1, NORM_MINMAX);
    imshow("g", g);
}
```

实验结果

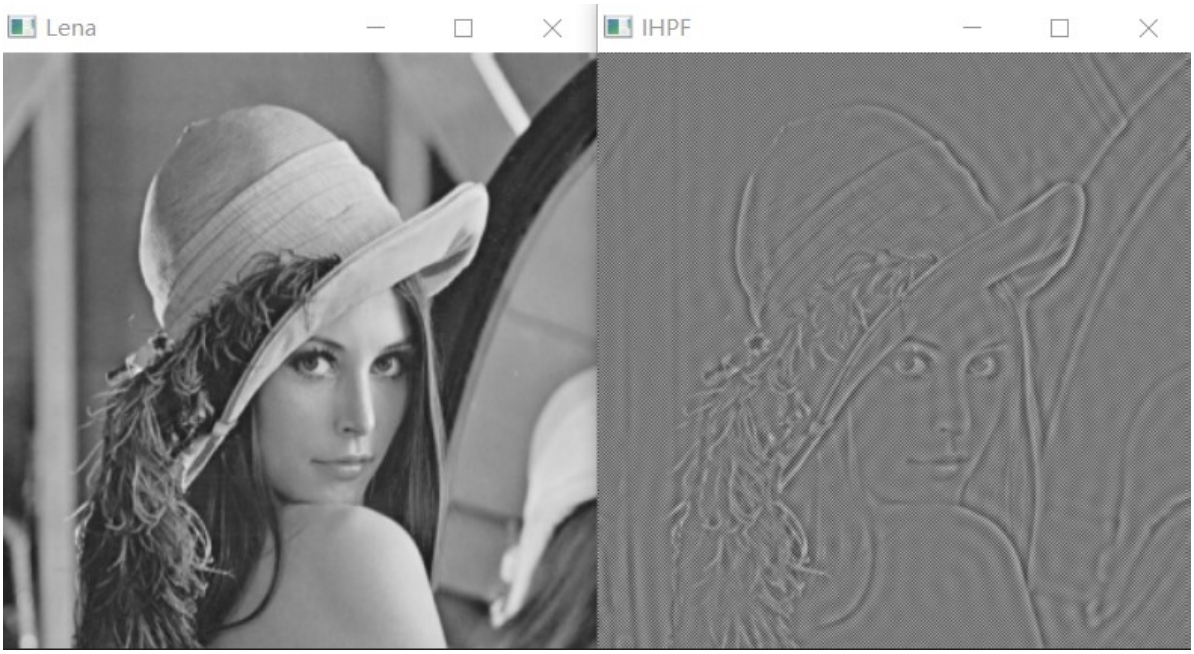
- [x] 理想低通滤波器



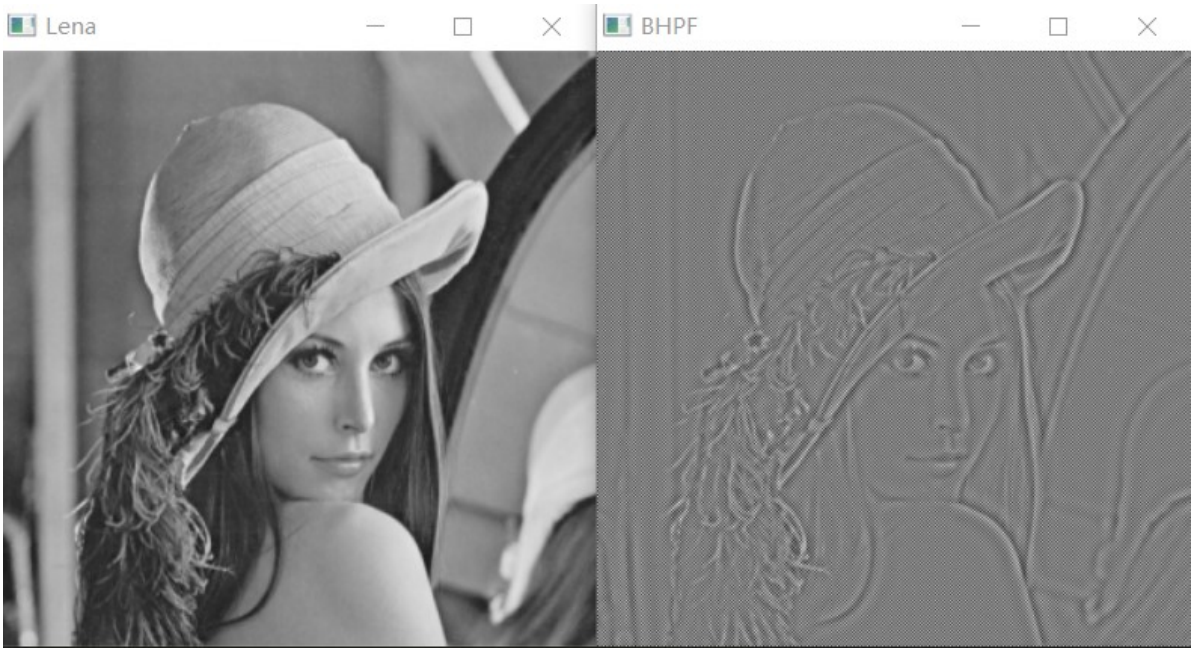
- [x] 布特沃斯低通滤波器



- [x] 理想高通滤波器



- [x] 布特沃斯高通滤波器



详细的代码见附件`lab5.cpp`。