HW3: Filtering in the Frequency Domain

1．Exercise

1.1 Rotation

(1)先将 f(x,y)乘上(−1)^(x+y)，把 M*N 的图片原点移动到（M/2，N/2）

(2)接着对原图做傅里叶变换，得到 F（u，v）

(3)取共轭，由实数 f(x,y)的共轭对称性可知，F*（u，v）= F（-u，-v），与原来的图像关于中心对称

(4)对 F*（u，v）做逆变换

(5)实部再乘以(−1)^(x+y)，把图片的原点还原

最后，就会得到一张与原图显示一样，但是旋转了 180 度的图像。

1.2 Fourier Spectrum

频谱图中的明亮线和原始图中对应的轮廓线是垂直的，由此可判断，图(d)水平方向的增强是由于左右边框与原图的轮廓线造成的，垂直方向的增强，则是由于上下边框与原图的轮廓线导致。

1.3 Lowpass and Highpass

1．原理如下图

$$g = f * h \Rightarrow g(x,y) = \sum_{\xi=0}^{N-1}\sum_{\eta=0}^{N-1} f(x-\xi, y-\eta)h(\xi,\eta) \Rightarrow G = FH$$

$$G(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} g(x,y)e^{-j2\pi(ux+vy)/N} = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\left(\sum_{\xi=0}^{N-1}\sum_{\eta=0}^{N-1} f(x-\xi,y-\eta)h(\xi,\eta)\right)e^{-j2\pi(ux+vy)/N}$$

$$= \frac{1}{N}\left(\sum_{x-\xi=0}^{N-1}\sum_{y-\eta=0}^{N-1} f(x-\xi,y-\eta)e^{-j2\pi(u(x-\xi)+v(y-\eta))/N}\right)\left(\sum_{\xi=0}^{N-1}\sum_{\eta=0}^{N-1} h(\xi,\eta)e^{-j2\pi(u\xi+v\eta)/N}\right)$$

根据给出的滤波器,可得

$g(x,y) = f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1) - f(x+1,y-1) - 2f(x+1,y) - f(x+1,y+1)$

又因为 $G(u,v)=H(u,v)F(u,v)$根据傅里叶变换的平移性可得,

$H(u,v) = [e^{-j2\pi(u/M + v/N)} + 2e^{-j2\pi u/M} + e^{-j2\pi(u/M - v/N)} - e^{-j2\pi(-u/M + v/N)} - 2e^{-j2\pi(-u/M)} + e^{-j2\pi(-u/M - v/N)}] = -j4\cos(2\pi v/N)\sin(2\pi u/M)$

2. 证明:这是一个高通滤波器

将滤波器变换为频率中心对称

$H(u,v) = -j4\cos[2\pi(v-N/2)/N]\sin[2\pi(u-M/2)/M]$

由上式可知,$v=N/2,u=M/2$ 处有最小值,即可证得,这是一个高通滤波器

1.4 Exchangeability

证明:与顺序有关

先做卷积,再直方图均衡化的图像可以表示成:

$g(x,y) = h(x,y) * f(x,y)$
$g'(x,y) = T[g(x,y)]$

而,先直方图均衡化,再卷积的图像:

$g(x,y) = T[f(x,y)]g'(x,y) = h(x,y) * g(x,y)$

一般而言,直方图均衡化不是一个线性变化,则有 $T(h(x,y) * f(x,y)) != h(x,y) * T(f(x,y))$, 得证。

2. Programming Tasks

实验环境:ubuntu 14.04

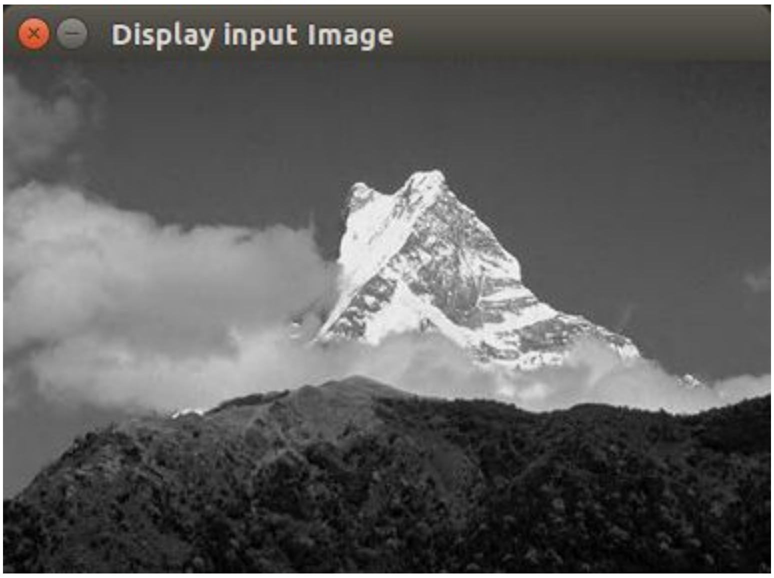
Opencv 3.0.0

运行操作:cmake .

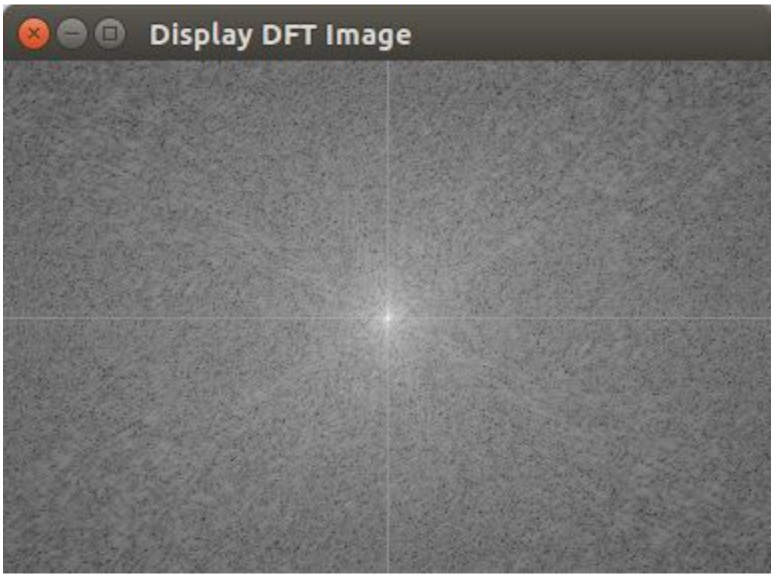
make
./dip_hw3

2.1 Pre-requirement
2.2 Fourier Transform

原图：



1．做傅里叶变换,显示频谱图如下



2．求傅里叶逆变换

3. 算法分析

(1)算法流程

①Dft2d(input, flags)其中 input 为 opencv 中的 Mat 类型变量,flags 为 bool 型,若 flags 为 true,则执行 DFT 操作,反之执行 IDFT 操作。

②先把输入图像转化成 complex 类型的二维数组,之后的操作也都是对 complex 类型的二维数组进行的。

③然后中心化,遍历数组的实部(因为此时虚部尚未赋值,所以为 0),各点乘以(-1)^(x + y)

④接着,进行 Y 方向的 DFT 变换

⑤之后 X 方向的 DFT 变换

⑥然后判断 flags 是否为真:

A. 若为真,则取频谱值,对数化,标定

B. 若为假,则进行 Y-IDFT、 X-IDFT 变换,去中心化(各点再乘一

次(-1)^(x + y))

⑦然后把上面处理的数组转换成 Mat 类型的 output 返回。

(3)遇到的 bug,以及解决方式:

①把数组转化成 Mat 类型的时候,没有分清楚 double、int 以及 uchar 类型的转换,导致图片无法显示。

解决方式:使用强制类型转换

②在前期中心化时,判断 flags 为真才执行,导致后面输出 IDFT 的还原图像时倒过来并且半透明的...

③debug 时 complex<double> output4_complex [row][col];

类似这样的声明无法执行,报段错误,可能是因为内存不够,

尝试这改成以下方式,就可以正常执行了

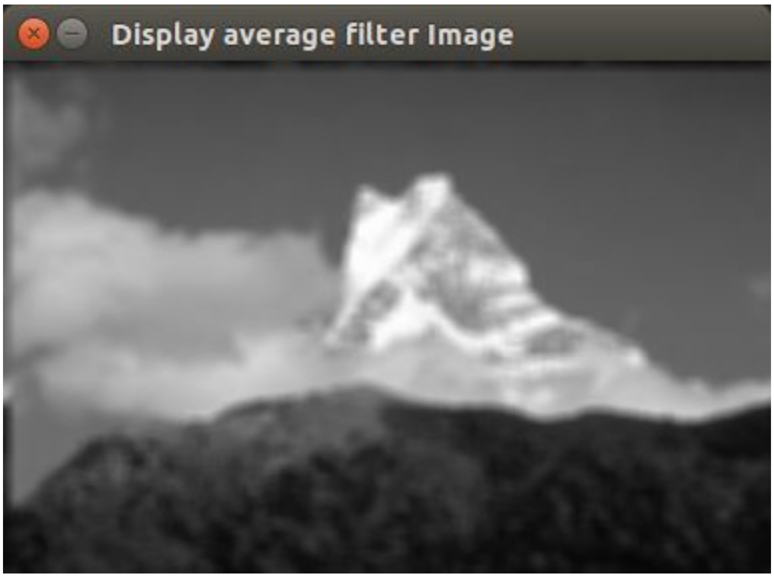
```
complex<double> output4_complex = new
complex<double>*[row];
For (int i = 0; i < row; i ++){
    Output4_complex[i] = new
    complex<double>[col];
    For (...) {
    //其他操作
    }
}
```

2.3 Bonus: Fast Fourier Transform

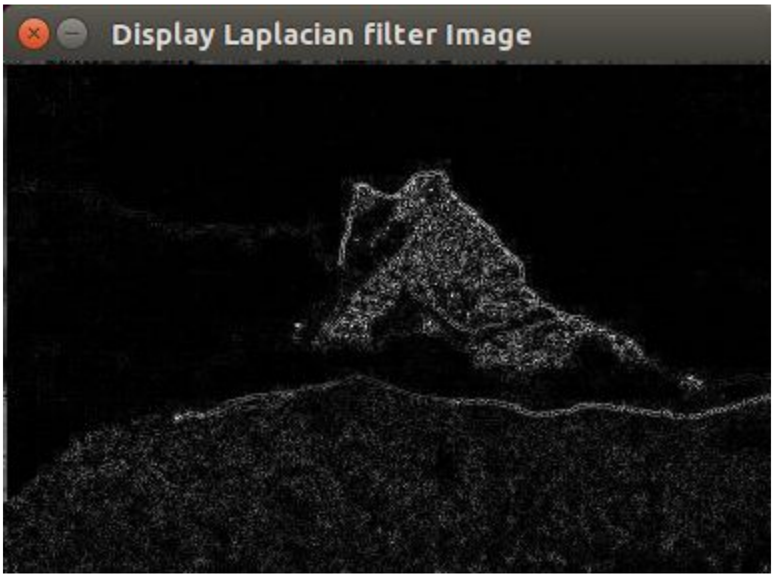
（看了挺久的...但是还是没怎么弄明白原理...）

2.4 Filtering in the Frequency Domain

1. 7*7



2. 3*3



3．算法

因为没有实现 FFT，所以采用了 DFT/IDFT 转换

①把 Mat 类型的图像，以及 double 数组的 filter 转换成 M*N 大小的 complex<double>数组，中心化，得到 f(x,y) 和 h(x,y)

②分别对其做 DFT 变换，得到 F(u,v)和 H(u.v)

③F(u,v)和 H(u,v)点乘，得到 G(u,v);

④G(u,v)做 IDFT 变换得到 g(x,y)

⑤g(x, y)再去中心化，标定，得到最后结果

7*7 平滑，过程分析如下

第一排第一张图为输入原图

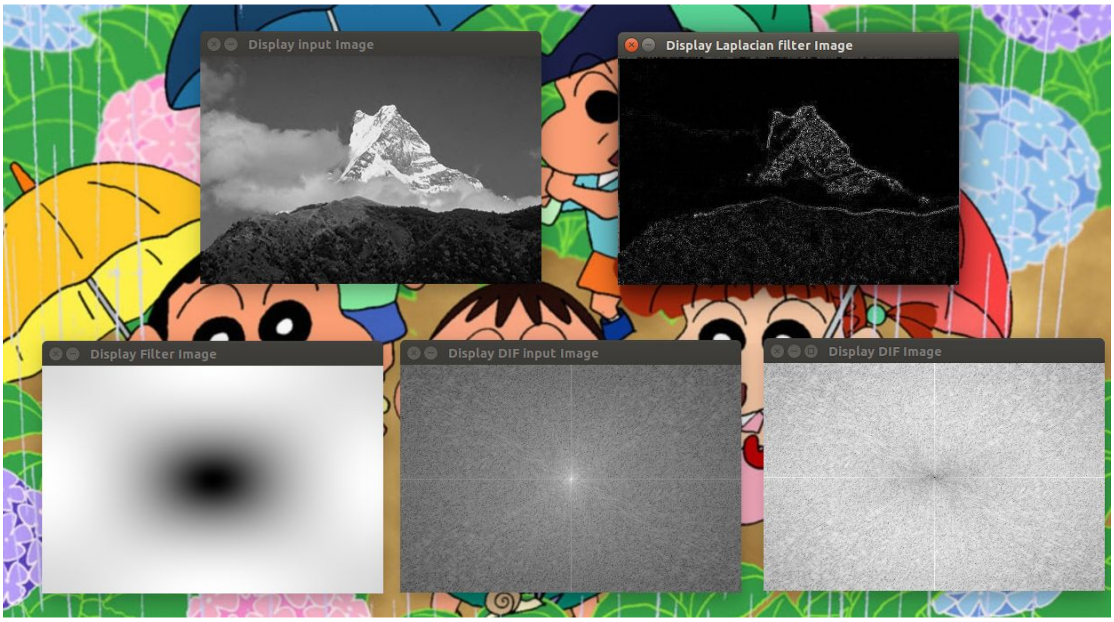
第一排第二张图为输出平滑图

第二排第一张图为滤波器频谱图

第二排第二张图为原图的频谱图

第二排第三张图为点乘之后的频谱图

3*3Laplacian 过程分析



第一排第一张图为输入原图

第一排第二张图为输出锐化图

第二排第一张图为滤波器频谱图

第二排第二张图为原图的频谱图

第二排第三张图为点乘之后的频谱图

一些 bug：

①代码复用度太高了

因为第一部分的代码只用了一个函数实现，所以会发现在写滤波器时的代码重复率很高（都怪自己开始写的时候没划分清楚模块），然后就对代码做了简单逻辑功能划分。

```
void release(complex<double>** input, int x)
```

因为功能实现需要用到很多 complex 数组，但是会爆栈，所以就用动态分配，这个函数用来释放空间。

```
complex<double>** dft(complex<double>** origin, bool flags, int row, int col)
```

这个是基于数组的 DFT/IDFT 实现，传入一个待处理的数组 origin，以及它的行数列数，bool 值为真表示 DFT 变换，反之则为 IDFT，最后返回一个变换过后的数组。

```
Mat show_dft(complex<double>** dft, int row, int col)
```

这个函数用于把经过 DFT 变换后的数组取频谱值，转换成 Mat 图像输出

```
//把IDFT变换后的数组转换成Mat形式，identify==true表示需要去中心化, demarcate标定
Mat show_idft(complex<double>** idft, int row, int col, bool identify, bool demarcate) {
```

这个函数用于把经过 IDFT 变化后的数组转换成 Mat 类型图像，因为需要判断是否需要标定和去中心化，所以加了两个 bool 值判断。

```
Mat dft2d(Mat &input, bool flags)
```

这是要求实现的函数，其中处理时会调用其他函数

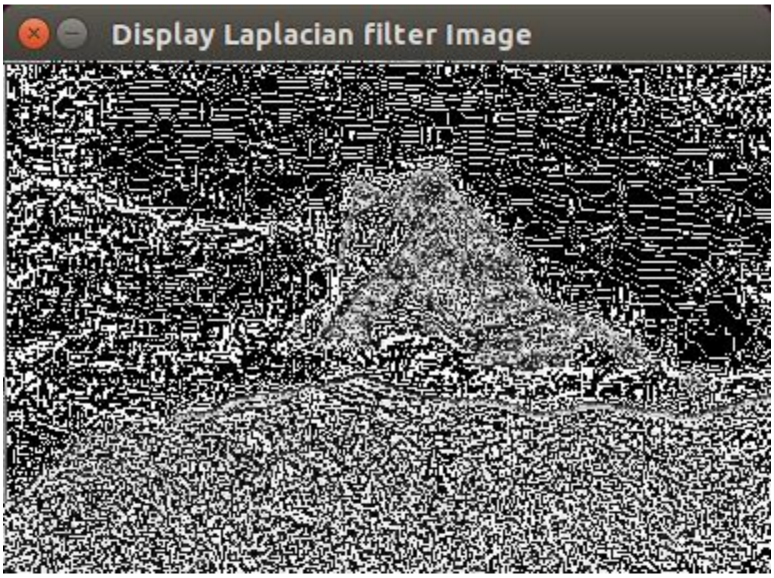
```
Mat filter2d_freq(Mat &input, double** filter, int size)
```

这个也是题意要求实现的，因为我设置的 filter 时数组，所以需要传入一个数组的大小来进行后继的处理。

②平滑算子处理过后的 f(x,y)是无负数的，而 Laplacian 算子处理过后，在最后输出的时候需要对原来的数组取绝对值。其中 demarcate 是一个 bool 值，为真表示需要标定。

```
if (demarcate) data[j] = (uchar) (fabs(temp) * rate);
```

若是不取绝对值，则会出错（如下图）:


Display Laplacian filter Image