2019年秋季 图像处理与分析 编程作业03

学院：人工智能学院 学号：201928014629008 姓名：牛李金梁

编程语言：Python3.6
必要依赖：numpy libopencv matplotlib

问题1 通过计算一维傅里叶变换实现图像二维快速傅里叶变换

具体见 F=dft2D(f)。

首先要建立一个和图像大小一样的二维数组 F，该数组类型要设为 complex。

之后调用 np.fft.fft()对图像的每一行进行一维傅里叶变换，将结果对应地存入 F 的行中。然后调用 np.fft.fft()对 F 的每一列进行一维傅里叶变换来改变 F 中的值，从而利用傅里叶变换的可分性完成了图像的二维傅里叶变换。

```
正变换与numpy差：
[[0.+0.j 0.+0.j 0.+0.j ... 0.+0.j 0.+0.j 0.+0.j]
 [0.+0.j 0.+0.j 0.+0.j ... 0.+0.j 0.+0.j 0.+0.j]
 [0.+0.j 0.+0.j 0.+0.j ... 0.+0.j 0.+0.j 0.+0.j]
 ...
 [0.+0.j 0.+0.j 0.+0.j ... 0.+0.j 0.+0.j 0.+0.j]
 [0.+0.j 0.+0.j 0.+0.j ... 0.+0.j 0.+0.j 0.+0.j]
 [0.+0.j 0.+0.j 0.+0.j ... 0.+0.j 0.+0.j 0.+0.j]]
```

通过打印与numpy中np.fft.fft2()的差发现该函数与np.fft.fft2()的结果完全相同。

问题2 图像二维快速傅里叶逆变换

具体见 f=idft2D(F)。

进行逆变换可以利用正变换实现。

首先将频域图像取共轭，然后进行傅里叶正变换，再除以图像的尺寸，再取一次共轭即可变回空间域。

通过与 np.fft.ifft2()比较，发现直接返回复数类型的数组即可。

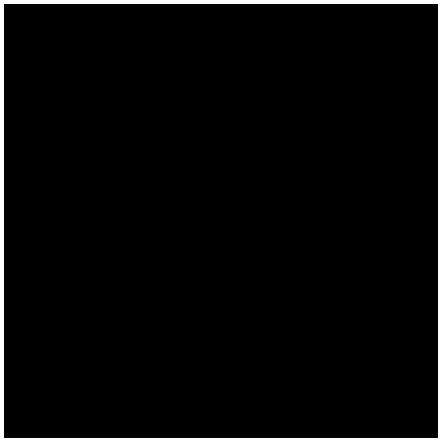
```
逆变换与numpy差：
[[-2.77555756e-17+1.30172023e-17j -4.85722573e-17+8.42967189e-18j
   4.16333634e-17+1.97358677e-17j ...  0.00000000e+00-4.34045093e-17j
  -9.71445147e-17+6.04696821e-18j -2.77555756e-17-3.70661618e-18j]
 [ 4.16333634e-17+9.20894220e-18j  5.55111512e-17+8.15862135e-18j
   1.38777878e-16+2.89719149e-17j ...  6.93889390e-17-3.69348717e-17j
  -2.77555756e-17+2.73506939e-18j  6.93889390e-17-7.24382576e-18j]
 [ 4.16333634e-17-9.20216594e-18j  4.16333634e-17-1.36880524e-17j
   9.71445147e-17-5.74288338e-18j ...  4.16333634e-17-5.05602437e-17j
  -4.16333634e-17-1.51491843e-17j  6.93889390e-17-2.41980372e-17j]
 ...
 [ 3.81639165e-17+1.24073386e-17j  4.85722573e-17+3.98444298e-18j
   1.49186219e-16+1.52296524e-17j ...  6.93889390e-17-4.75058428e-17j
  -2.08166817e-17-8.10695234e-18j  4.85722573e-17-1.26648366e-17j]
 [ 7.63278329e-17-1.06184050e-17j  8.32667268e-17-1.00973138e-17j
   1.59594560e-16-1.29528278e-17j ...  7.63278329e-17-4.38381902e-17j
  -2.08166817e-17-3.45835082e-17j  6.24500451e-17-3.77776694e-17j]
 [-1.38777878e-17+9.75104329e-18j  1.38777878e-17+2.61563774e-18j
   9.71445147e-17+2.16264452e-17j ...  4.16333634e-17-4.97420098e-17j
  -6.93889390e-17-6.69748951e-18j  0.00000000e+00-2.08234580e-17j]]
```

通过打印与 numpy 中 np.fft.ifft2()的差发现该函数与 np.fft.ifft2()的结果只存在浮点误差。

问题3 测试图像二维快速傅里叶变换与逆变换

首先读取图片，通过除以255进行归一化。

然后调用之前所写的dft2D()和idft2D()得到g，再利用np.abs()得到g的模，这就是变换后又逆变换的结果。考虑到该结果与原图的灰度区间不同，因此将这一结果映射到[0, 255]，然后用原图f减去该图并显示出来。



结果如图所示，同时通过打印发现每个像素值都是极小的浮点误差。

问题4 计算图像的中心化二维快速傅里叶变换与谱图像

首先新建一个uint8类型的全0灰度图，通过计算将[226:285, 251:260]设为1，即生成了矩形物体图像并归一化。

调用dft2D()进行傅里叶变换。

再实现一个中心化函数centered_spectrum()，即把图像从中心分成4部分，然后将右下移动到左上，右上移动到左下，左下移动到右上，左上移动到右下即可。

之后将中心化后的图像按公式对数化。

注意显示时要取模。而且plt.imshow()可以自行将结果映射到灰度图，所以直接显示即可。