图像处理与分析课后作业二

李一帆

November 1, 2020

1 问题 1

离散傅里叶变换的推导过程如下。

$$\begin{aligned} F(u,v) &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \\ &= \sum_{x=0}^{M-1} e^{-j2\pi\frac{ux}{M}} \sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\frac{vy}{N}} \\ &= \sum_{x=0}^{M-1} F(x,v)\, e^{-j2\pi\frac{ux}{M}} \end{aligned}$$

由上式可知，由于傅里叶变换具有可分离的特性，因此可以分别对行和列做 FFT 来进行等效，从而减少计算量。

此脚本主要包含两个函数，dft2D 以及 showimg。

- 函数 dft2D 输入为 imgPath，即图像的路径；输出为 fft2d，即经过二维快速傅里叶变换之后的图像。主要作用是对输入图像进行快速离散傅里叶变换；
- 函数 showimg 输入为图像 f，图像标题 title，以及保存位置 save。主要功能是显示以及保存图像。

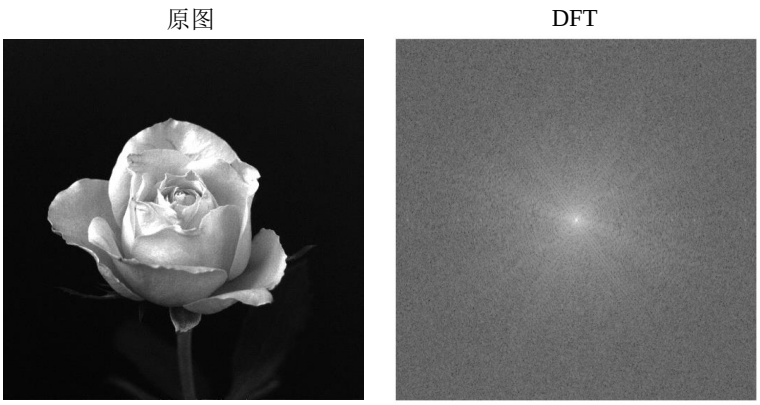

Figure 1: 原图以及经过二维快速傅里叶变换之后的图像对比

直接运行 Q1.py 后，会产生经过二维快速傅里叶变换之后的频谱图像，如 Figure 1 所示。

2 问题 2

逆离散傅里叶变换的推导过程如下所示。

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\, e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

$$MNf^*(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F^*(u,v)\, e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

其计算过程可由如下公式表示。

$$F(u,v) \rightarrow F^*(u,v)$$

$$F^*(u,v) \rightarrow MNf^*(x,y)$$

$$MNf^*(x,y) \rightarrow f(x,y)$$

首先对 $F(u,v)$ 求共轭，得到 $F^*(u,v)$。然后对 $F^*(u,v)$ 进行离散傅里叶变换（即利用第一问的公式）得到 $MNf^*(x,y)$。最后将 $MNf^*(x,y)$ 除以 MN，并取共轭，可以得到逆离散傅里叶之后的结果 $f(x,y)$。

本脚本主要由以下函数构成。

- 函数 idft2D 输入为经过二维快速傅里叶变换之后的图像 F（复数域），输出为逆离散傅里叶变换 idft 后的图像。

直接运行 Q2.py 后，会产生经过二维快速傅里叶逆变换之后的图像，如 Figure 2 所示。

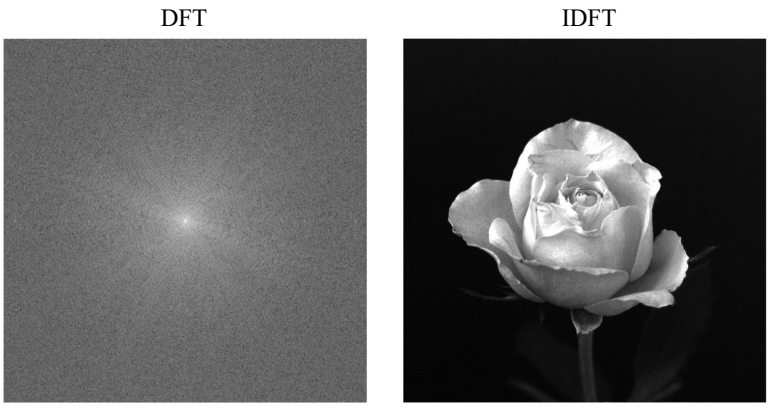

Figure 2: 二维快速傅里叶变换图像以及二维快速傅里叶逆变换后的图像对比

3 问题 3

本脚本的函数主要是前两个问题所用到的函数，因此在此不做详细介绍。处理过程如下，首先读取图像，将 RGB 图转换为灰度图并进行 min-max 归一化处理。然后对归一化后的图像调用 dft2D 函数进行离散傅里叶变换，然后调用 idft2D 函数进行逆离散傅里叶变换。最后将归一化图和逆离散傅里叶图做差，并转化为 uint8 类型，可以得到最终的效果。

运行 Q3.py 后，可以得到原归一化图与逆离散傅里叶变换后图像的差值，如 Figure 3 所示。

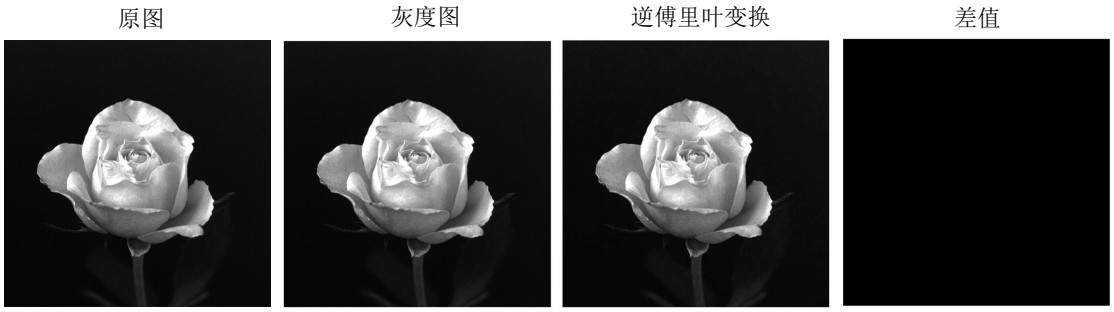

Figure 3: 原图与二维快速傅里叶逆变换图像对比

4 问题 4

对离散傅里叶后结果进行偏移的公式如下所示。

$$f(x,y)(-1)^{x+y} \Leftrightarrow F(u-M/2, v-N/2)$$

本脚本用到的新的函数如下。

- 函数 create_rectangle 输入为产生的小矩形的高 H 和宽 W，周围用 0 进行填充。以及输入参数 normalize，表示是否进行归一化。返回值为新建的矩形图像。

处理过程如下。首先利用 create_rectangle 创建一个长 60 宽 10 的矩形，并对图像进行归一化处理。然后调用 dft2D 函数，对生成的小矩形图像进行离散傅里叶变换（未偏移），然后利用上述公式进行偏移后，再利用离散傅里叶变换得到偏移后的的结果。最后对得到的频谱进行取对数运算，可以得到对数变换后的谱图。

直接运行 Q4.py 后，会得到下列结果。

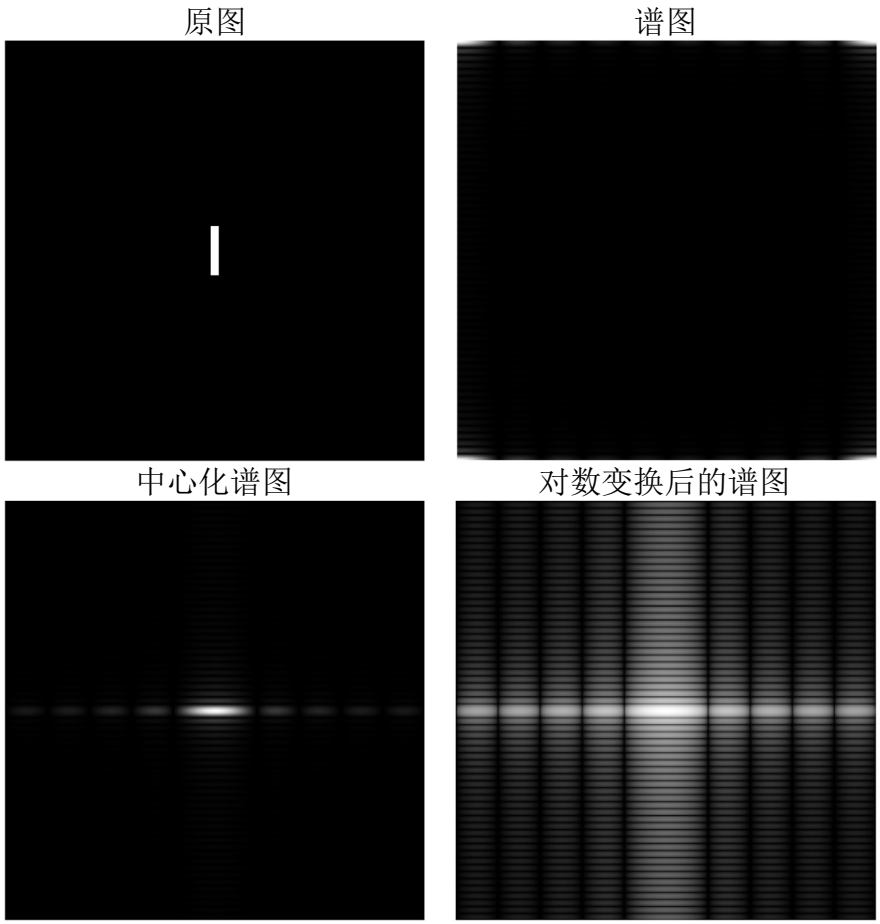

Figure 4: 原图与二维快速傅里叶逆变换图像对比

5 问题 5

本脚本用到的新的函数主要有两个，pad_img 和 show_dft_img。

- pad_img 输入为待填充的图像 img，输出为长宽为 2 的幂次的填充图像；
- show_dft_img 输入为 img_path，save_path。主要作用是显示、生成 dft 图像以及保存产生的图像。

由于基-2FFT 只能对 2 的幂次大小的序列长度进行处理，因此，如果输入图像的长宽不符合 2 的幂次的要求，则需要对原图像进行填充处理。本算法主要使用了 zero padding 和 replicate padding 两种填充方式。填充过程如下，首先对输入图像的长和宽分别取对数（向上取整），然后将得到的指数以 2 为底取幂，得到填充后图像的大小。然后，对矩形图按照填充后的大小进行填充，得到填充后的结果。最后，利用快速离散傅里叶得到对数谱图（显示时去掉填充部分）。

直接运行 Q5.py 可以得到最终的结果，如下图所示。

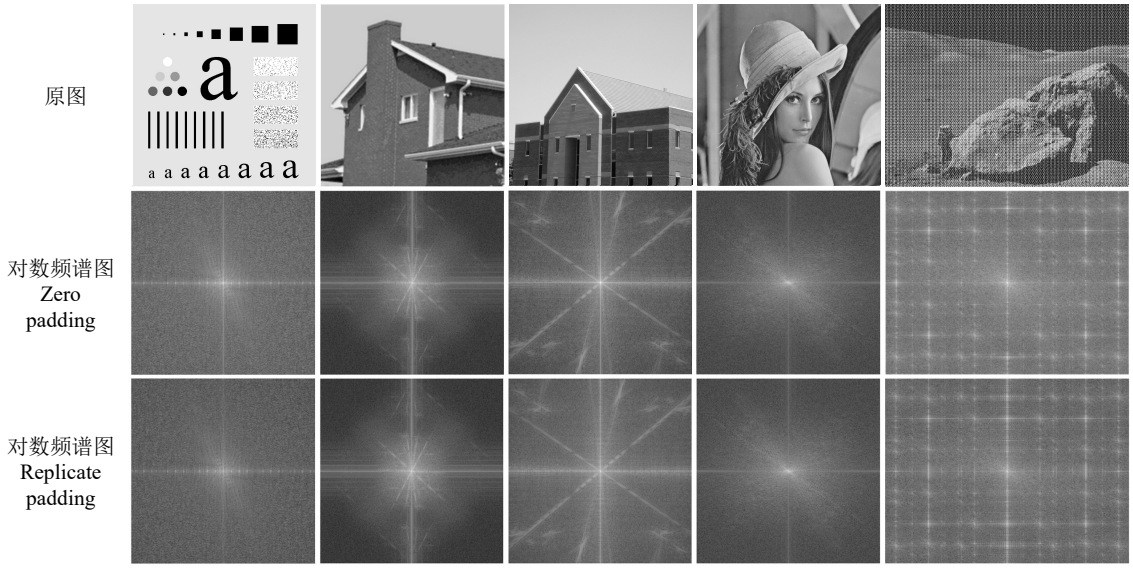

Figure 5: 不同 padding 下的频谱对比图

由上图可以看出，在使用 zero padding 时，谱图会偏亮一些，也就是说高频分量相对于 replicate padding 会多一些。主要原因在于 zero padding 的方式使得边沿跳变相较于 replicate padding 更加严重，导致在频域引入了更多高频分量。