华中科技大学

图 像 处 理 作 业

离散傅里叶变换（DFT）

院　　系: 人工智能与自动化学院

专 业 班 级: 自动化 1903

学 生 姓 名: 李子奥

学 生 学 号: U201914629

指 导 教 师: 谭山

2022 年 6 月 15 日

目　　录

1　1D 离散傅里叶变换的推导

1.1　连续变量的傅里叶变换

对于一连续变量 t 的连续函数 $f(t)$，其傅里叶变换为：

$$\begin{aligned} F(\mu) &= \boldsymbol{F}\{f(t)\} = \int_{-\infty}^{\infty} f(t)e^{-j2\pi\mu t}dt \\ F(\mu) &= F\{f(t)\} = \int_{-\infty}^{\infty} f(t)[\cos(2\pi\mu t) - j\sin(2\pi\mu t)]dt \end{aligned} \tag{1.1}$$

傅里叶变换是一个可逆的过程，其逆变换为：

$$f(t) = F^{-1}\{F(\mu)\} = \int_{-\infty}^{\infty} F(\mu)e^{j2\pi\mu t}d\mu \tag{1.2}$$

从输入变量与输入变量的意义上看，傅里叶变换的输入变量 t 为时域上的变量，傅里叶变换的输出变量 μ 为频域上的变量。因此，傅里叶变换及其逆变换可以看作信号在时域和频域上的相互转换。

1.2　离散变量的傅里叶变换

采样函数是一个无穷长、等间隔的单位脉冲序列，其定义为：

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T) \tag{1.3}$$

其中 ΔT 为采样周期，即两次采样操作之间间隔的时间。$s_{\Delta T}(t)$ 每隔固定时间产生一次脉冲，因此其时域函数图像如图1.1所示。

将连续函数 $f(t)$ 与采样函数相乘，能够得到离散化的函数 $\tilde{f}(t)$：

$$\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t)\delta(t - n\Delta T) \tag{1.4}$$

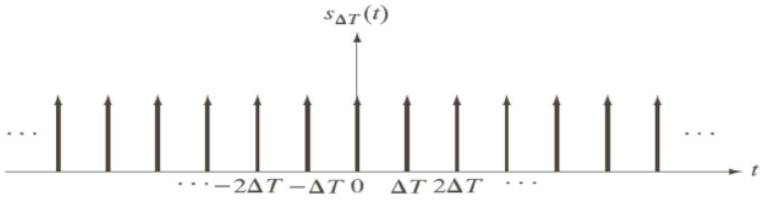

图 1.1　采样函数

根据公式1.1，离散函数 $\tilde{f}(t)$ 的傅里叶变换为：

$$\begin{aligned}\tilde{F}(\mu) = F\{\tilde{f}(t)\} &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(t)\delta(t - n\Delta T)e^{-j2\pi\mu t}dt \\ &= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} f(t)\delta(t - n\Delta T)e^{-j2\pi\mu t}dt \\ &= \sum_{n=-\infty}^{\infty} f(n\Delta T)e^{-j2\pi\mu n\Delta T} \\ &= \sum_{n=-\infty}^{\infty} f_n e^{-j2\pi\mu n\Delta T}\end{aligned} \tag{1.5}$$

其中 f_n 的定义为：

$$f_n = f(n\Delta T) = \int_{-\infty}^{\infty} f(t)\delta(t - n\Delta T)dt \tag{1.6}$$

2 编程实现 2D 离散傅里叶正变换及其逆变化

2D 离散情况下的傅里叶变换对分别为公式2.1和公式2.2。其中公式2.1为傅里叶正变化，公式2.2为傅里叶逆变换。

$$\begin{aligned} F(u,v) &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)} \\ x &= 0,1,2,\ldots M-1 \\ y &= 0,1,2,\ldots N-1 \end{aligned} \tag{2.1}$$

$$\begin{aligned} f(x,y) &= \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)} \\ u &= 0,1,2,\ldots M-1 \\ v &= 0,1,2,\ldots N-1 \end{aligned} \tag{2.2}$$

为了观察傅里叶变换后的频域图像，需要将频域中心从 $(0,0)$ 移动至 $(\frac{M}{2},\frac{M}{2})$。进行移动的公式为：

$$F(u-M/2,v-N/2) \Leftrightarrow f(x,y)(-1)^{x+y} \tag{2.3}$$

实验代码放在附录A中。

3 实验过程及结果

数据集中共有 5 章图片，由于我的代码依据公式使用迭代的方式进行傅里叶变换与傅里叶逆变换，对一张图片进行转换需要花较长时间，因此在读入图像后，先将原图像进行缩小得到输入图像 f。

该输入图像首先进行傅里叶正变化得到 DFT(f)，再通过傅里叶逆变换得到 IDFT(DFT(f))，称其为重构图像 g。为了验证程序的正确性，使用 MSE 和 PSNR 两个指标评估输入图像 f 和输出图像 g 之间的相似程度。

	barb	boat	lena	mandrill	peppers-bw
MSE	0	0	0	0	0
PSNR	inf	inf	inf	inf	inf

表 3.1 缩小至 128×128 后傅里叶逆变换重建结果

	barb	boat	lena	mandrill	peppers-bw
MSE	0	0	0	0	0
PSNR	inf	inf	inf	inf	inf

表 3.2 缩小至 64×64 后傅里叶逆变换重建结果

	barb	boat	lena	mandrill	peppers-bw
MSE	0	0	0	0	0
PSNR	inf	inf	inf	inf	inf

表 3.3 缩小至 32×32 后傅里叶逆变换重建结果

	barb	boat	lena	mandrill	peppers-bw
MSE	0	0	0	0	0
PSNR	inf	inf	inf	inf	inf

表 3.4 缩小至 16×16 后傅里叶逆变换重建结果

表3.1，表3.2，表3.3和表3.4分别为输入图像尺寸为 128，64，32 和 16 的情况下经过傅里叶逆变换还原后的图像 g 与输入图像 f 之间的对比。表中可以看出所有 MSE 指标都为 0，所有 PSNR 指标都为 inf，表示输入图像 f 与输出图像 g 完全一致。

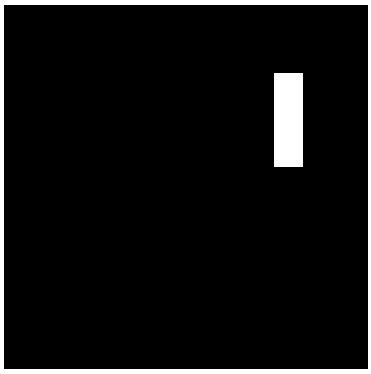

图 3.1　输入图像

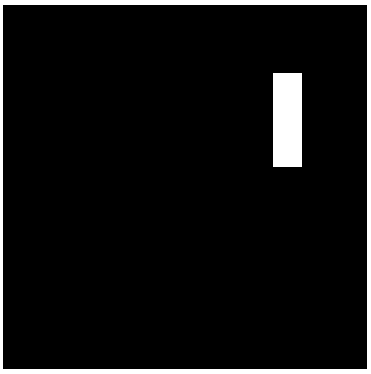

图 3.2　输出图像

图 3.3　频域图像

为了定性地观察傅里叶变换的过程，我们对比输入图像与输出图像之间的关系，同时观察其频域图像的形状与分布。为了尽可能地准确，在定性分析的过程的输入图像 f 的尺寸为 128×128。图3.1为输入图像，图3.2为经过傅里叶正变换与傅里叶逆变换得到的输出图像，图3.3为输入图像 f 经过傅里叶变换后的中心化谱的对数变换。进行对数变换的原因是：频率域上的值分布极差大，不进行对数

变化不利于观察。其余图片的转换结果放至附录B中。

附录 A　实验代码

实验代码

```
import os
from matplotlib.image import imsave
import matplotlib.pyplot as plt
import numpy as np
import PIL.Image as Img

OutputDir = "output"
ALL_FILES = []
CURRENT_FILE = ""
IF_SHOW = False
Suffix = ".png"
ShrinkWidth = 64

def show_pic(pic1: np.array, str1=''):
        global CURRENT_FILE

        pic1 = pic1.squeeze()

        filename = os.path.join(OutputDir, str1+'_'+str(ShrinkWidth)+'_'+
            CURRENT_FILE+'.png')
        imsave(filename, pic1, cmap=plt.get_cmap('gray'),vmin=0,vmax=255)

        if IF_SHOW:
                plt.imshow(pic1, cmap=plt.get_cmap('gray'),vmin=0,vmax=255)
                plt.xticks([])
                plt.yticks([])
                plt.title(CURRENT_FILE)
                plt.show()

# DFT
def dft(img):
        H, W, channel = img.shape

        G = np.zeros((H, W, channel), dtype=np.complex128)

        x = np.tile(np.arange(W), (H, 1))
        y = np.arange(H).repeat(W).reshape(H, -1)

        for c in range(channel):
                for v in range(H):
                        for u in range(W):
```

```
41                                  G[v, u, c] = np.sum(img[..., c] * np.exp(-2j *
                                        np.pi * (x * u / W + y * v / H)))

43          return G

45  def show_dft(img):
            H, W, _ = img.shape
47          ii, jj = np.meshgrid(np.arange(H), np.arange(W))

49          new_img = img * np.power(-1, ii +jj)[:, :, np.newaxis]
            dft_img = dft(new_img).squeeze()
51          dft_img = np.log(np.absolute(dft_img))

53          filename = os.path.join(OutputDir, 'DFT_'+str(ShrinkWidth)+'_'+
                CURRENT_FILE+Suffix)
            plt.imsave(filename, dft_img, cmap=plt.get_cmap('gray'))
55
            if IF_SHOW:
57                  plt.imshow(dft_img, cmap=plt.get_cmap('gray'))
                    plt.colorbar()
59                  plt.show()

61  # IDFT
    def idft(G):
63          H, W, channel = G.shape
            out = np.zeros((H, W, channel), dtype=np.float32)
65
            x = np.tile(np.arange(W), (H, 1))
67          y = np.arange(H).repeat(W).reshape(H, -1)

69          for c in range(channel):
                    for v in range(H):
71                          for u in range(W):
                                    out[v, u, c] = np.abs(np.sum(G[..., c] * np.
                                        exp(2j * np.pi * (x * u / W + y * v / H))))
                                        / (H*W)
73
            out = np.clip(out, 0, 255)
75          out = out.astype(np.uint8)

77          return out

79  def get_pic_loss(pic1, pic2):
            assert(pic1.shape == pic2.shape)
81          mse = np.mean(np.power(pic1 - pic2, 2))
            psnr = 10 * np.log10(255 ** 2 / mse)
83          return mse, psnr
```

```
85  def main():
        global CURRENT_FILE
87
        _, _, ALL_FILES = list(os.walk("./data"))[0]
89      print(ALL_FILES)

91      os.makedirs(OutputDir, exist_ok=True)

93      for file in ALL_FILES:
            CURRENT_FILE = os.path.splitext(os.path.basename(file))[0]
95
            pic = plt.imread(os.path.join("data", file))
97          img = Img.fromarray(pic).resize((ShrinkWidth, ShrinkWidth),
                Img.NEAREST)
            pic = np.array(img)
99          if pic.ndim == 2:
                pic = pic[:, :, np.newaxis]
101
            show_dft(pic)
103
            dft_pic = dft(pic)
105         idft_pic = idft(dft_pic)

107         mse, psnr = get_pic_loss(pic, idft_pic)

109         show_pic(pic, "Original")
            show_pic(idft_pic, "IDFT")
111         print('{}: MSE={}, PSNR={}'.format(CURRENT_FILE, mse, psnr))

113 if __name__ == "__main__":
        main()
```

附录 B　转换结果

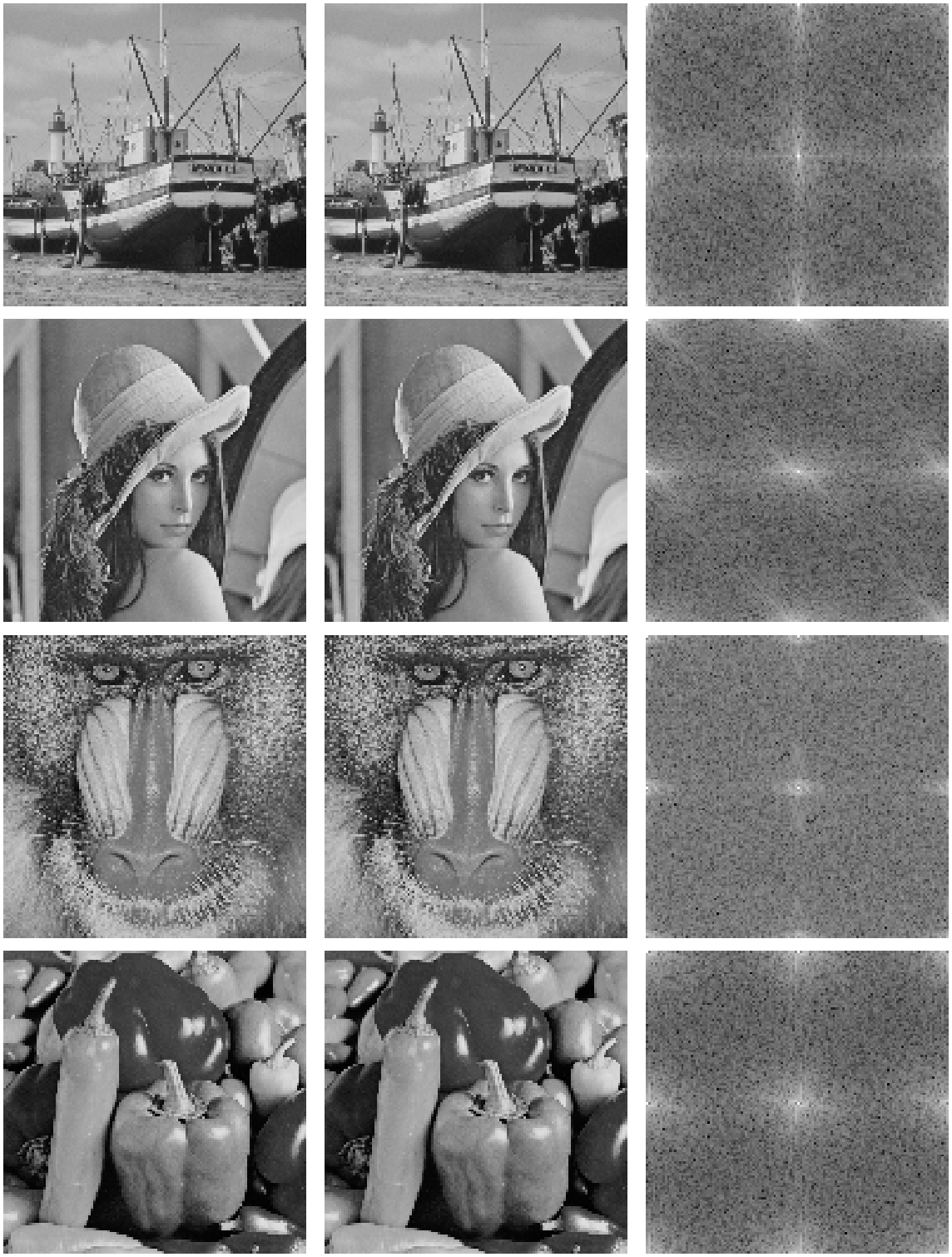

图 B.1　输入图像　　图 B.2　输出图像　　图 B.3　频域图像