二维信号（图像）的傅里叶变换

Dezeming Family

2022 年 4 月 17 日

DezemingFamily 系列书和小册子因为是电子书，所以可以很方便地进行修改和重新发布。如果您获得了 DezemingFamily 的系列书，可以从我们的网站 [https://dezeming.top/] 找到最新版。对书的内容建议和出现的错误欢迎在网站留言。

目录

一　二维图像 DFT

尽管二维信号的傅里叶变换可以直接通过张量积的形式导出，但我们还是倾向于直接给出二维傅里叶变换对：

$$F(u,v)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(x,y)e^{-j2\pi(ux+vy)}dxdy$$
$$f(x,y)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}F(u,v)e^{j2\pi(ux+vy)}dudv$$

然后是二维 DFT：

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

我们这么写可能更容易看出来效果：

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

其实就是先对横轴做 DFT，然后再对横轴做 DFT。

二　程序实现

直接用二维方法来实现更简单，但为了更清楚它和一维变换的关系，我们用两次一维傅里叶变换来实现。我们并不实现 FFT，而是实现最简单的 DFT 算法。

2 1　DFT 程序

实现程序如下：

```
import math
import numpy as np
class dft:
    def __init__(self,Num,xvalues):
        self.xvalues = xvalues
        self.yvalues = []
        self.Num = Num
        self.xk = []
        self.index = list(range(0,self.Num,1))
        self.Xk()
    def xjw(self,fre = []):
        for f in fre:
            p = 0
            i = 0
            for x in self.xvalues:
                p = math.e**(-1j * f * self.index[i]) * x + p
                i = i + 1
            self.yvalues.append(p)
    def Xk(self):
```

```
        W = []
        for x in self.index:
            W.append(x*2*math.pi/self.Num)
        self.xjw(W)
        for y in self.yvalues:
            self.xk.append(abs(y))
```

逆变换只需要改一行代码（类名改为 idft）：

```
    p = math.e**(-1j * f * self.index[i]) * x + p
    # 改为：
    p = math.e**(1j * f * self.index[i]) * x + p
```

2 2 程序测试

读取图像：

```
import cv2
img = cv2.imread("cat.png",cv2.IMREAD_GRAYSCALE)
info = img.shape
height = info[0]
width = info[1]
# 转为float类型
flo32_img = np.float32(img)
```

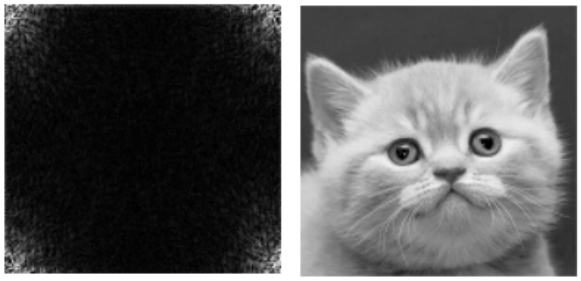
对每一排做 DFT，然后再对每一列做 DFT：

```
dst = np.zeros((height, width, 1), np.uint8)
dftResult = np.zeros((height, width, 1), np.complex)
# 先横着做一遍DFT再纵着做一遍DFT
dftw = []
for h in range(0, height):
    for w in range(0, width):
        dftw.append(flo32_img[h, w])
    DFT1 = dft(width, dftw)
    for w in range(0, width):
        dftResult[h, w] = DFT1.yvalues[w]
    dftw.clear()
dfth = []
for w in range(0, width):
    for h in range(0, height):
        dfth.append(dftResult[h, w])
    DFT1 = dft(height, dfth)
    for h in range(0, height):
        dst[h, w] = DFT1.xk[h] / (100) # 除以100是为了显示方便
        # 存储图像DFT的值
        dftResult[h, w] = DFT1.yvalues[h]
    dfth.clear()
cv2.imshow("dst",dst)
cv2.waitKey(0)
```

逆变换也同样如此，只不过输入是 DFT 图像。注意输出结果要除以 (width*height)

```
# 逆变换
dst2 = np.zeros((height, width, 1), np.uint8)
dftw = []
for h in range(0, height):
    for w in range(0, width):
        dftw.append(dftResult[h, w])
    IDFT1 = idft(width, dftw)
    for w in range(0, width):
        dftResult[h, w] = IDFT1.yvalues[w]
    dftw.clear()
dfth = []
for w in range(0, width):
    for h in range(0, height):
        dfth.append(dftResult[h, w])
    IDFT1 = idft(height, dfth)
    for h in range(0, height):
        dftResult[h, w] = IDFT1.yvalues[h]
        dst2[h, w] = IDFT1.xk[h] / (width * height)
    dfth.clear()
cv2.imshow("dst2",dst2)
cv2.waitKey(0)
```

显示结果（左边是傅里叶变换后的图像，右边是逆变换后的图像）：



参考文献

[1] Signals and Systems Alan V. Oppenheim, Alan S. Willsky, with S. Hamid Nawab. Prentice Hall, 2013.