遥感数字图像处理实验课

图像傅里叶变换及快速傅里叶变换

李荣昊

2021年10月

实验内容

- 了解快速傅里叶变换
- 将给定的图像进行快速傅立叶变换
- 将已变换的图像进行快速傅立叶反变换，得到原来图像
- 观察比较傅里叶变换前后图像，理解傅里叶变换的物理意义

遥感数字图像处理

- 图像变换的目的

1. 使图像处理问题简化
2. 有利于图像特征的提取
3. 有助于从概念上对图像信息的理解

- 图像变换的形式：

1. 一般来说，是二维正交变换，变换是可逆的
2. 一般来说，正变换和逆变换不能太过于复杂
3. 正交变换的特点是在变换域中图像能量集中在低频率分布上，而边缘、线状物体反映在高频率上，有利于图像处理。

遥感数字图像处理

- 图像变换实例

1. 对数变换（乘除变加减）
2. 拉式变换（微分方程的求解）
3. 傅里叶变换（频谱分析和滤波）

- 图像变换的应用:

1. 图像增强与恢复
2. 特征提取
3. 图像压缩
4. 形状分析

遥感数字图像处理

- **本质**

以数字的形式输入到计算机中，利用一定的数学方法，按照数字图像的规律进行变换，将一幅图像变为另一幅经过修改（改进）的图像，由图像得到图像的过程。

- **方法**

离散方法 → 空间域

傅里叶逆变换 ↑↓ 傅里叶变换

连续方法 → 频率域

遥感数字图像处理

- **频率域的来源**
  - 对于任何一个波，可以将其表示为:

$$x(t) = A_0 + \lim_{N\to\infty}\sum_{n=1}^{N} A_n sin(2\pi n f_0 t + \varphi_n)$$

  - 考虑$e^{i\varphi} = cos\varphi + i\sin\varphi$, 得:

$$x(t) = \sum_{n=-\infty}^{+\infty} c_n e^{i2\pi n f_0 t}$$

- **频率域的来源**
  - 设x(t)的变化范围为有限区间[$t_0, t_0 + \mathrm{T}$], 用$e^{-i2\pi n f_0 t}$同乘以上式两边，并从t_0到$t_0 + \mathrm{T}$进行积分，得：

$$\int_{t_0}^{t_0+T} x(t)e^{-i2\pi m f_0 t}dt = \int_{t_0}^{t_0+T}\left[\sum_{n=-\infty}^{+\infty} c_n e^{i2\pi(n-m)f_0 t}\right]dt$$

$$= \sum_{n=-\infty}^{+\infty} c_n \int_{t_0}^{t_0+T} e^{i2\pi(n-m)f_0 t}\,dt$$

  - 当 $f_0 = \frac{1}{\mathrm{T}}$ 时，经计算，积分$\int_{t_0}^{t_0+T} e^{i2\pi(n-m)f_0 t}\,dt$，当$n - m = 0$时为T，当$n - m \neq 0$时为0，因此上面的等式右边的和号中只剩下$n = m$那一项，得：

遥感数字图像处理

- **频率域的来源**

$$c_n = \frac{1}{\mathrm{T}} \int_{t_0}^{t_0+T} x(t) e^{-i2\pi m f_0 t} dt$$

- 考虑$[-\infty, +\infty]$区间，则可得：

$$c_n = \frac{1}{\mathrm{T}} \int_{-\infty}^{+\infty} x(t) e^{-i2\pi m f_0 t} dt$$

- 对原函数而言：

$$x(t) = \sum_{n=-\infty}^{+\infty} c_n e^{i2\pi n f_0 t} = \int_{-\infty}^{+\infty} c_n e^{i2\pi m f_0 t} dt$$

- **傅里叶变换的作用**

  - 表现信号变化的快慢

    信号变化的快慢与频率域的频率有关，噪声、边缘、跳跃部分代表图像的高频分量，背景区域和慢部分代表图像得到低频分量。

遥感数字图像处理

- 连续函数的傅里叶变换

若把一个一维输入信号作一维傅立叶变换，该信号就被变换到频域上的一个信号，即得到了构成该输入信号的频谱，频谱反映了该输入信号由哪些频率构成。这是一种分析与处理一维信号的重要手段。

- 当一个一维型号f(x)满足狄里赫莱条件，即：

1. 具有有限个间断点
2. 具有有限个极值点
3. 绝对可积

傅里叶变换

- **连续傅里叶变换（FT）**

➢ 一维傅里叶变换

$$\Im\{f(x)\} = F(\mu) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi\mu x} dx$$

$$f(x) = \Im^{-1}\{F(\mu)\} = \int_{-\infty}^{\infty} F(u) e^{j2\pi\mu x} d\mu$$

➢ 二维傅里叶变换

$$\Im\{f(x,y)\} = F(\mu,\nu) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) e^{-j2\pi(\mu x+\nu y)} dxdy$$

$$f(x,y) = \Im^{-1}\{F(\mu,\nu)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,\nu) e^{j2\pi(ux+\nu y)} d\mu d\nu$$

$f(x)$代表原函数即空域, F(μ)代表变换后的函数即频域

遥感数字图像处理

- **傅里叶变换的作用**

要在数字图像处理中应用傅立叶变换， 还需要解决两个问题：
1. 在数学中进行傅立叶变换的f(x)为连续（模拟）信号， 而计算机处理的是数字信号（图像数据）
2. 数学上采用无穷大概念，而计算机只能进行有限次计算。通常，将受这种限制的傅立叶变换称为离散傅立叶变换（Discrete Fourier Transform，DFT)

傅里叶变换

- 一维离散傅里叶变换

$$\Im\{f(x)\} = F(\mu) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi\mu x} dx$$

将时间和频率离散化

$$F(k) = \Im[f(n)] = \sum_{n=0}^{N-1} f(n) e^{-j\frac{2\pi kn}{N}}, k = 0,1,2,\ldots, N-1$$

对于位于时域的N点序列f(n) 0≤n< N，其离散傅里叶变换形式

令$W_N = e^{-j\frac{2\pi}{N}} \Rightarrow$

$$F(k) = \Im[f(n)] = \sum_{n=0}^{N-1} f(n) W_N^{nk}$$

离散傅里叶逆变换表示为：

$$f(n) = \Im^{-1}[F(k)] = \frac{1}{N} \sum_{k=0}^{N-1} F(k)\, e^{j\frac{2\pi kn}{N}} = \frac{1}{N} \sum_{k=0}^{N-1} F(k)\, W_N^{-kn}$$

- **傅里叶变换**

由欧拉公式可知：

$$e^{j\theta} = cos\theta + jsin\theta$$

将上带入，并利用cos(-θ)=cos(θ)，有:

$$F(u) = \sum_{x=0}^{N-1} f(x)\left(\cos\left(\frac{2\pi ux}{N}\right) - jsin\left(\frac{2\pi ux}{N}\right)\right)$$

可见，离散序列的傅立叶变换仍是一个离散的序列，每一个u对应的傅立叶变换结果是所有输入序列f(x)的加权和（每一个f(x)都乘以不同频率的正弦和余弦值），u表示每个傅立叶变换结果的频率

- **傅里叶变换**

通常，傅里叶变换为负数形式，我们可以写作：

$$F(u) = R(u) + jI(u)$$

式中，R（u）和I（u）分别是F（u）的实部和虚部，也可以表示为指数形式：

$$F(u) = |F(u)|e^{j\phi(u)}$$

$$|F(u)| = \sqrt{R(u)^2 + I(u)^2}$$ 频谱

$$\phi(u) = \arctan(\frac{I(u)}{R(u)})$$ 相位谱

傅里叶变换

- **一维离散傅里叶变换**

一维离散傅里叶变换的矩阵形式，如下所示。

$$\begin{bmatrix} F(0) \\ F(1) \\ \vdots \\ F(N-1) \end{bmatrix} = \begin{bmatrix} W^{0} & W^{1\times 0} & W^{2\times 0} & \cdots & W^{(N-1)\times 0} \\ W^{0\times 1} & W^{1\times 1} & W^{2\times 1} & \cdots & W^{(N-1)\times 1} \\ \vdots & \vdots & \vdots & & \vdots \\ W^{0\times(N-1)} & W^{1\times(N-1)} & W^{2\times(N-1)} & \cdots & W^{(N-1)\times(N-1)} \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ \vdots \\ f(N-1) \end{bmatrix}$$

从式中，我们可以看到，计算每个频域分量，需要进行N次乘法和N-1次加法，完成整个变换需要N^2次乘法，N(N-1)次加法运算，计算复杂度是$O(N^2)$。

傅里叶变换

- **一维离散傅里叶变换**

二维离散傅里叶变换的实现：

[100]:
```
def DFT(img):
    h,w = img.shape
    dft = np.zeros((h,w)).astype(complex)
    x = np.arange(w).repeat(h).reshape(w,-1).transpose()
    y = np.arange(h).repeat(w).reshape(h,-1)
    for v in range(h):
        for u in range(w):
            e = np.exp(-2j*np.pi*(x*u/w+y*v/h))
            dft[v,u] = np.sum(img*np.power(-1,x+y)*e)
    return np.abs(dft)
```

快速傅里叶变换(FFT)

- W_N^{nk} 的特征

欧拉公式：

$$e^{ix} = \cos(x) + i\sin(x)$$

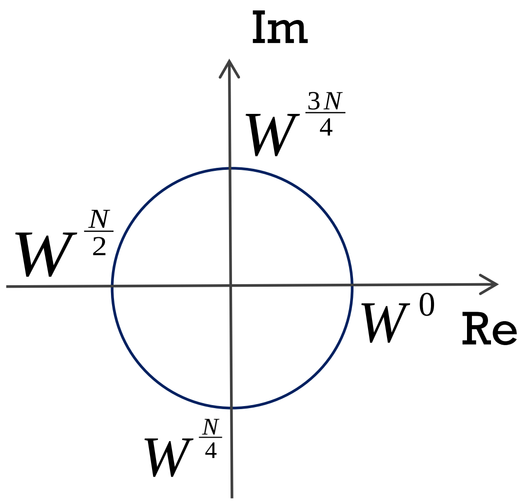
$$W = e^{-j\frac{2\pi}{N}}$$

$$W^N = e^{-j\frac{2\pi}{N}\cdot N} = e^{-j2\pi} = 1$$

$$W^{\frac{N}{2}} = -1$$

$$W^{\frac{N}{4}} = -j$$

$$W^{\frac{3N}{4}} = j$$



快速傅里叶变换(FFT)

- W_N^{nk} 的特征

(1) W_N^{nk}的周期性

$$W_N^{nk} = W_N^{(n+N)k} = W_N^{n(k+rN)} = W_N^{nk+rN} = W_N^{(n+lN)(k+hN)}$$

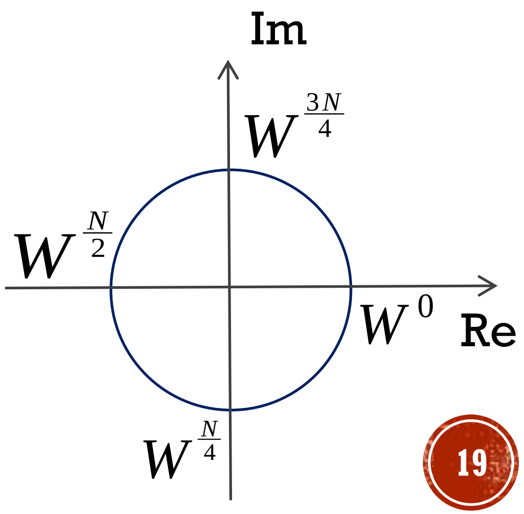
以N为周期，其中，k=0,1,2,…,N-1; n=0,1,2,…,N-1；r为整数。

例如N=8时，

$$W_8^9 = W_8^{1+8} = W_8^1, W_8^{42} = W_8^{2+5\times 8} = W_8^2$$

(2) W_N^{nk}的对称性

$$W^{m\times n+\frac{N}{2}} = W^{m\times n} \times W^{\frac{N}{2}} = -W^{m\times n}$$



快速傅里叶变换(FFT)

- W_N^{nk} 的特征

N=4的系数矩阵：

由 $W^{m\times n}$ 的周期性可以得出：$W^4=W^0, W^6=W^2, W^9=W^1$;
由 $W^{m\times n}$的对称性可以得出：$W^3=-W^1, W^2=-W^0$;

$$\begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \xrightarrow[\text{周期性}]{\text{对称性}} \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & -W^0 & -W^1 \\ W^0 & -W^0 & W^0 & -W^0 \\ W^0 & -W^1 & -W^0 & W^1 \end{bmatrix}$$

快速傅里叶变换(FFT)

- **FFT算法思想**

　　根据奇偶性，把f(n)序列分解成若干短序列，巧妙结合系数矩阵元素。

　　f(n)序列分解成偶数列和奇数列，一个序列由f(n)的偶数点组成(n=0, 2, 4…)，另一个序列由f(n)的奇数点组成(n=1, 3…)，如下：

$$\begin{cases} g(n)=f(2n) \\ h(n)=f(2n+1) \end{cases} \quad n=0,1,2,3,\cdots,\frac{N}{2}-1$$

DFT

$$F(k)=\sum_{N=0}^{N-1} f(n)\bullet W_N^{n\cdot k}$$

$$=\sum_{N\text{为偶数}} f(n)W_N^{n\cdot k}+\sum_{N\text{为奇数}} f(n)W_N^{n\cdot k}$$

快速傅里叶变换(FFT)

- FFT算法思想

$$F(k)=\sum_{n=0}^{N-1} f(n)\bullet W_N^{n\cdot k}$$

$$=\sum_{n=0}^{\frac{N}{2}-1} f(2n)\cdot W_N^{k(2n)}+\sum_{n=0}^{\frac{N}{2}-1} f(2n+1)\cdot W_N^{k(2n+1)}$$

$$=\sum_{n=0}^{\frac{N}{2}-1} f(2n)\cdot W_{\frac{N}{2}}^{nk}+\sum_{n=0}^{\frac{N}{2}-1} f(2n+1)\cdot W_{\frac{N}{2}}^{nk}\cdot W_N^k$$

●●●

$$=G(k)+W_N^k H(k)$$

$$W_{2N}^k=W_N^{\frac{k}{2}}$$

$$F(k)=G(k)+W_N^k H(k)$$

2×(N/2)²+N次乘法**VS**
N²次乘法

快速傅里叶变换(FFT)

- FFT算法思想

$$F(k) = G(k) + W_N^k H(k)$$

其中:

$$G(k) = \sum_{r=0}^{\frac{N}{2}-1} f(2r) \cdot W_{\frac{N}{2}}^{rk}$$

称为原始序列f(n)n为偶数点的 $\dfrac{N}{2}$ 点DFT

$$H(k) = \sum_{r=0}^{\frac{N}{2}-1} f(2r+1) \cdot W_{\frac{N}{2}}^{rk}$$

称为原始序列f(n)n为奇数点的 $\dfrac{N}{2}$ 点DFT

快速傅里叶变换(FFT)

- FFT算法思想

$$F\left(k+\frac{N}{2}\right)=G\left(k+\frac{N}{2}\right)+W_N^{k+\frac{N}{2}}H\left(k+\frac{N}{2}\right)$$

又因为：

$$W_{\frac{N}{2}}^{r\left(k+\frac{N}{2}\right)}=W_{\frac{N}{2}}^{rk}$$

$$W_N^{k+\frac{N}{2}}=-W_N^k$$

所以：

$$F\left(k+\frac{N}{2}\right)=G(k)-W_N^kH(k)$$

G(k)和H(k)可以完全表示F(k)

- FFT算法思想

$$F(k)=G(k)+W_N^kH(k)$$

其中：

$$G(k)=\sum_{r=0}^{\frac{N}{2}-1}f(2r)\cdot W_{\frac{N}{2}}^{rk}$$

称为原始序列f(n)n为偶数点的 $\frac{N}{2}$ 点DFT

$$H(k)=\sum_{r=0}^{\frac{N}{2}-1}f(2r+1)\cdot W_{\frac{N}{2}}^{rk}$$

称为原始序列f(n)n为奇数点的 $\frac{N}{2}$ 点DFT

快速傅里叶变换(FFT)

- FFT算法思想

$$F(k)=G(k)+W_N^k H(k)$$ （以N=8为例）

$$\begin{cases} F(0)=G(0)+W_8^0\cdot H(0)\\ F(1)=G(1)+W_8^1\cdot H(1)\\ F(2)=G(2)+W_8^2\cdot H(2)\\ F(3)=G(3)+W_8^3\cdot H(3)\\ F(4)=G(4)+W_8^4\cdot H(4)\\ F(5)=G(5)+W_8^5\cdot H(5)\\ F(6)=G(6)+W_8^6\cdot H(6)\\ F(7)=G(7)+W_8^7\cdot H(7) \end{cases}$$

$$G(m+4)=G(m)$$
$$H(m+4)=H(m)$$
$$W_8^{m+4}=-W_8^m\ (m=0,1,2,3)$$

对称性，周期性

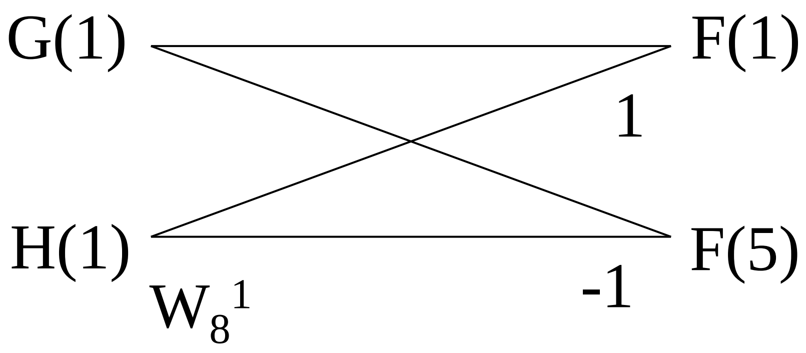
$$\begin{cases} F(0)=G(0)+W_8^0\cdot H(0)\\ F(1)=G(1)+W_8^1\cdot H(1)\\ F(2)=G(2)+W_8^2\cdot H(2)\\ F(3)=G(3)+W_8^3\cdot H(3)\\ F(4)=G(0)-W_8^0\cdot H(0)\\ F(5)=G(1)-W_8^1\cdot H(1)\\ F(6)=G(2)-W_8^2\cdot H(2)\\ F(7)=G(3)-W_8^3\cdot H(3) \end{cases}$$

快速傅里叶变换(FFT)

- **蝶形运算流程图**

由G(1)、H(1)、F(1)和F(5)所构成的结构为蝶形运算单元，它的左方两个节点为输入节点，代表输入数值。右方两个节点为输出节点，表示输入数值的叠加。运算由左向右进行。线旁的W_8^1和$-W_8^1$为矩阵系数，如下图：

$$F(1)=G(1)+W_8^1 * H(1)$$
$$F(5)=G(1)-W_8^1 * H(1)$$

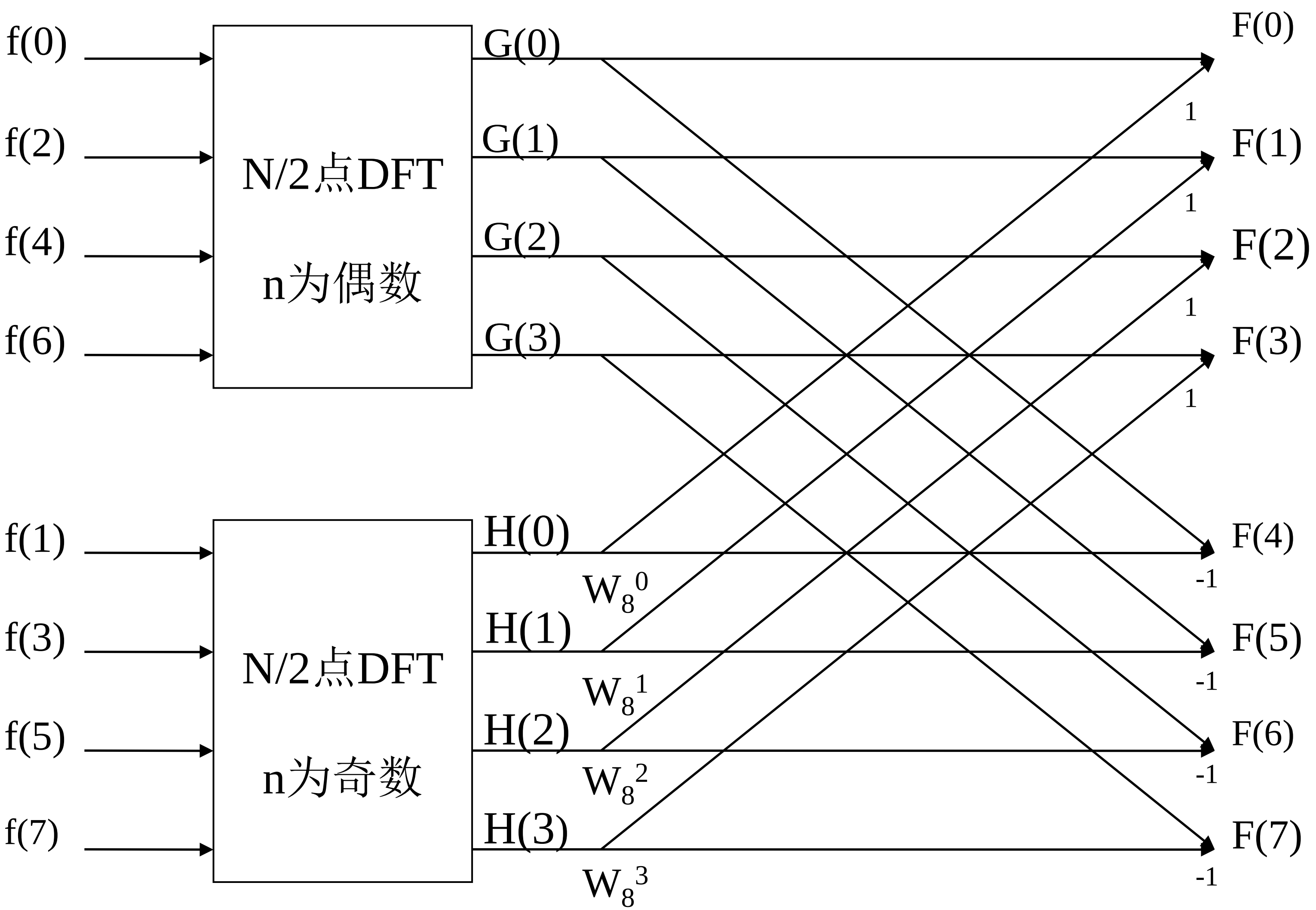


$$\begin{cases} F(k) = G(k) + W_N^k * H(k) \\ F\left(k+\frac{N}{2}\right) = G(k) - W_N^k * H(k) \end{cases}$$

k=0,1…,N-1

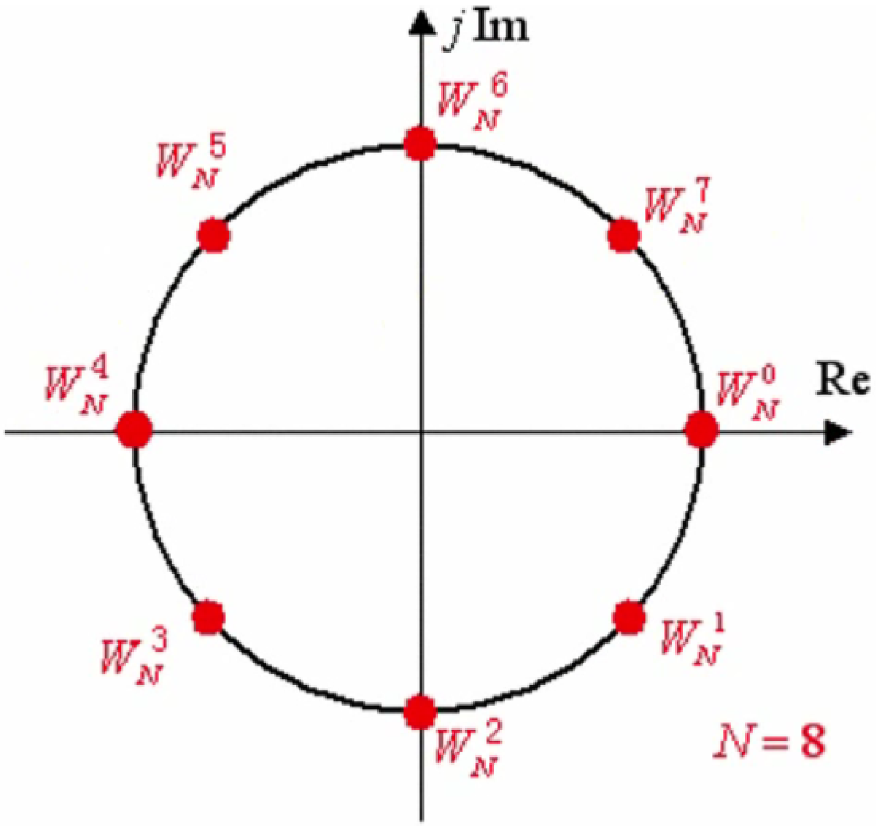
快速傅里叶变换(FFT)

- 蝶形运算流程图



快速傅里叶变换(FFT)

- FFT算法思想



快速傅里叶变换(FFT)

- **FFT算法思想**

G(m)和H(m)都是N/2点的DFT，如果N/2为偶数，则可继续对他们进行奇偶分组，如下：

偶数序列的奇偶序列 →
$$\begin{cases} a(n)=g(2l) \\ b(n)=g(2l+1) \end{cases} \quad (l=0,1,2,3,\cdots,\frac{N}{4}-1)$$

奇数序列的奇偶序列 →
$$\begin{cases} c(n)=h(2l) \\ d(n)=h(2l+1) \end{cases} \quad (l=0,1,2,3,\cdots,\frac{N}{4}-1)$$

$$F(k)=G(k)+W_N^k H(k)$$
$$W_{2N}^k=W_N^{\frac{k}{2}}$$

→

$$G(m)=A(m)+W_{\mathrm{N}}^{2m}\cdot B(m)$$
$$H(m)=C(m)+W_N^{2m}\cdot D(m)$$

快速傅里叶变换(FFT)

- FFT算法思想

$$\begin{cases} G(0) = A(0) + W_8^0 \cdot B(0) \\ G(1) = A(1) + W_8^2 \cdot B(1) \\ G(2) = A(0) - W_8^0 \cdot B(0) \\ G(3) = A(1) - W_8^2 \cdot B(1) \end{cases}$$

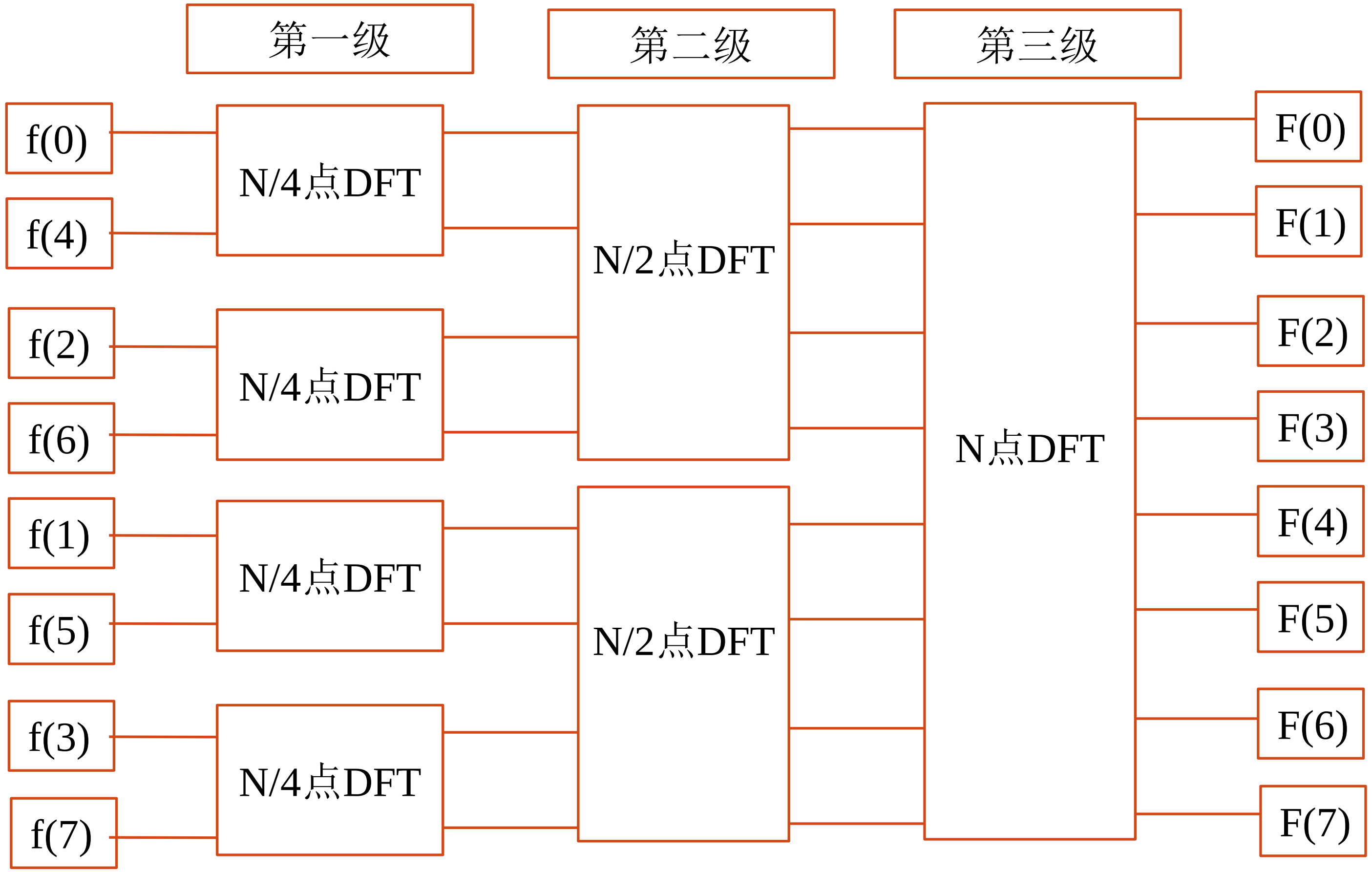
$$\begin{cases} H(0) = C(0) + W_8^0 \cdot D(0) \\ H(1) = C(1) + W_8^2 \cdot D(1) \\ H(2) = C(0) - W_8^0 \cdot D(0) \\ H(3) = C(1) - W_8^2 \cdot D(1) \end{cases}$$

$$\Rightarrow \begin{cases} \mathrm{F}(\mathrm{m}) = \mathrm{G}(\mathrm{m}) + W_N^{2m} * H(m) \\ \mathrm{F}\left(\mathrm{m} + \frac{N}{4}\right) = \mathrm{G}(\mathrm{m}) - W_N^{2m} * H(m) \end{cases}$$

m=0,1…,N/2-1

快速傅里叶变换(FFT)

- 8点DFT逐级分解运算框图



快速傅里叶变换(FFT)

- FFT算法思想

至此，A(m)、B(m)、C(m)和D(m)都已经是2点的DFT，它们可以由原始数据f(n)直接求出，计算公式如下：

0，1，2，3，4，5，6，7	⇨	0，2，4，6， 1，3，5，7	⇨	0，4， 2，6， 1，5， 3，7

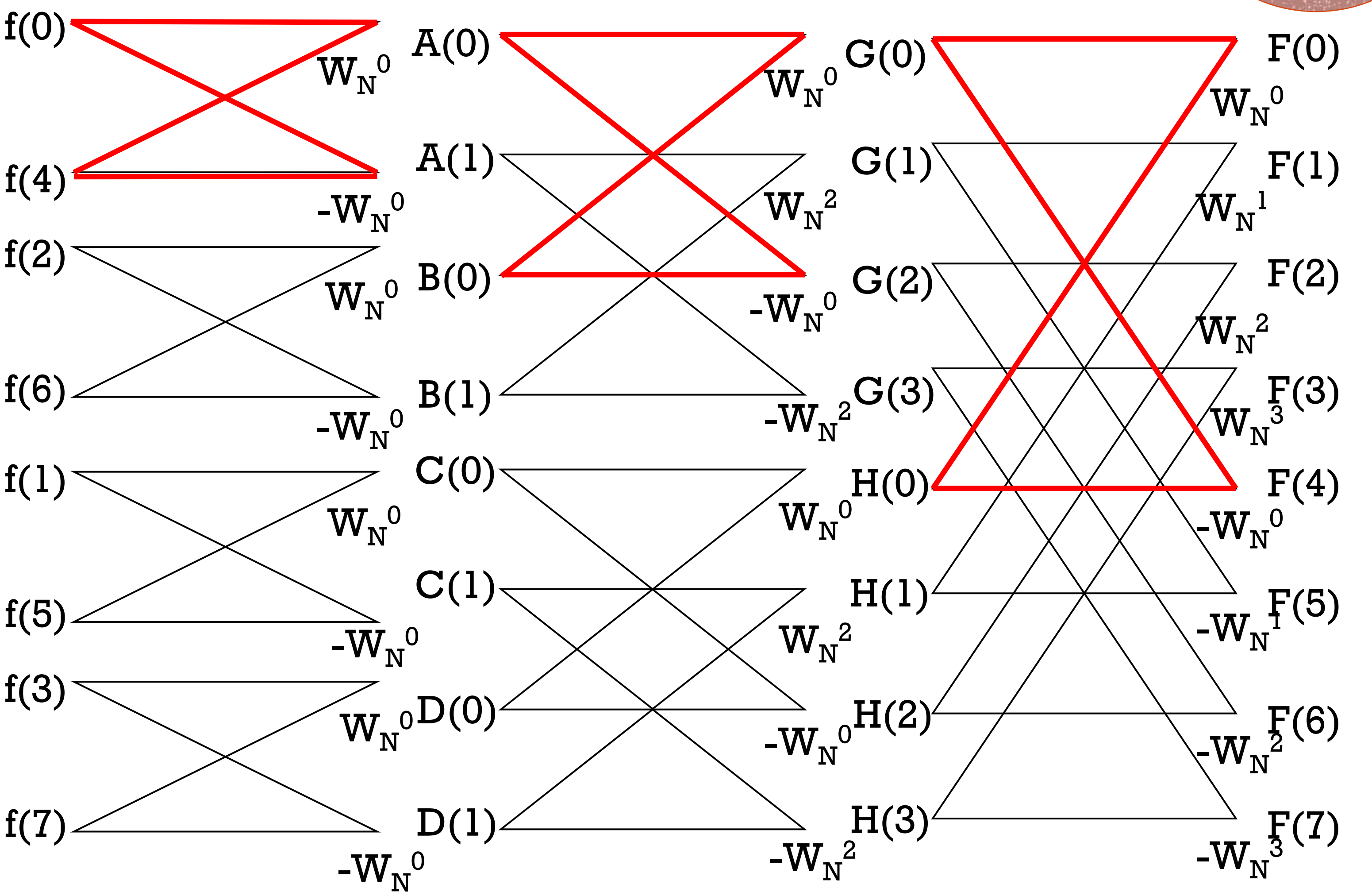
$$\begin{cases} A(0)=f(0)+W_8^0\cdot f(4) \\ A(1)=f(0)-W_8^0\cdot f(4) \end{cases} \qquad \begin{cases} B(0)=f(2)+W_8^0\cdot f(6) \\ B(1)=f(2)-W_8^0\cdot f(6) \end{cases}$$

$$\begin{cases} C(0)=f(1)+W_8^0\cdot f(5) \\ C(1)=f(1)-W_8^0\cdot f(5) \end{cases} \qquad \begin{cases} D(0)=f(3)+W_8^0\cdot f(7) \\ D(1)=f(3)-W_8^0\cdot f(7) \end{cases}$$

快速傅里叶变换(FFT)

$$f(x) \Leftrightarrow \begin{cases} A(x) \\ B(x) \\ C(x) \\ D(x) \end{cases} \Leftrightarrow \begin{cases} G(x) \\ H(x) \end{cases} \Leftrightarrow F(x)$$

- 8点DFT蝶形流图



快速傅里叶变换(FFT)

- **DFT蝶形流图顺序**

观察蝶形流图，可以看出，输出序列F(x)是按照x从小到大的顺序排列的，而输入序列f(x)不是从小到大的顺序，**而是按照所谓的码位倒序排列的。**

如果把自然顺序的十进制数转换成二进制数，然后将这些二进制的首末位倒序再重新转换成十进制数，那么这时的十进制数的排列就是码位倒序排列——比特倒转法。

快速傅里叶变换(FFT)

- DFT蝶形流图码位倒序

十进制数	二进制数	二进制码位倒序	十进制码位倒序
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

快速傅里叶变换(FFT)

- Rader(雷德)算法

◆ 按自然顺序排列的二进制数，其下面一个数总是比其上面一个数大1，即下面一个数是上面一个数在最低位加1并向高位进位而得到的。而倒位序二进制数的下面一个数是上面一个数在最高位加1并由高位向低位进位而得到。

十进制数	二进制数	二进制码位倒序	十进制码位倒序
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

快速傅里叶变换(FFT)

十进制数	二进制数	二进制码位倒序	十进制码位倒序
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

- Rader算法

◆若已知某个倒位序J，要求下一个倒位序数：

- ●首先判断J的最高位是否为0，这可与k=N/2相比较，因为N/2总是等于100...的。
  - ●如果k>J，则J的最高位为0，只要把该位变为1(J与k=N/2相加即可)，就得到下一个倒位序数；
  - ●如果k<=J，则J的最高位为1，可将最高位变为0(J与k=N/2相减即可)。
- ●然后(k<=J时)还需判断次高位，这可与k=N\4相比较，若次高位为0，则需将它变为1(加N\4即可)其他位不变，既得到下一个倒位序数；若次高位是1，则需将它也变为0。然后再判断下一位，以此循环。

快速傅里叶变换(FFT)

- FFT运算量分析（N=8）

每按奇偶分解一次即可得一级蝶形流图，共有$\log_2 8=3$级蝶形流图

→

每级蝶形流图都有N/2 =4个蝶形单元，三级共有蝶形单元12个

↓

每个蝶形单元需要一次复数加法和两次复数乘法运算
8点FFT一共需要复数加法运算12次，复数乘法运算24次

←

直接采用DFT的定义计算，8点DFT一共需要复数乘法运算$N^2=64$次，复数加法运算$N(N-1)=56$次

$N\log_2 N$ vs N^2
随着N的增大，FFT的效率将急剧增加

快速傅里叶变换(FFT)

- **FFT算法总结**
  - 找到DFT对称的联系，降低复杂度；
  - 只适用于N为2的整数次幂的情况；

图像快速傅里叶变换

- 二维离散傅里叶变换性质–分离性

$$F(\mu,\upsilon)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}e^{-j2\pi(\frac{\mu x}{M}+\frac{\nu y}{N})}f(x,y)$$

f(x, y)	$f(x,y)$

x维度FFT ↓

F(μ, y)	$\mathrm{F}(\mu,y)=\sum_{x=0}^{N-1}e^{-i2\pi\left(\frac{\mu x}{N}\right)}f(x,y)$

y维度FFT ↓

F(μ, v)	$\mathrm{F}(\mu,\mathrm{v})=\sum_{y=0}^{N-1}e^{-i2\pi\left(\frac{vy}{N}\right)}F(\mu,y)$

图像快速傅里叶变换

- **频谱原点移动**

在数字图像处理中，通常将傅里叶变换频谱原点移动到M×N的中心，以便能清楚地分析傅里叶变换的情况。

$$设\begin{cases}\mu_0 = \dfrac{M}{2} \\ \upsilon_0 = \dfrac{N}{2}\end{cases},则 e^{j2\pi\left(\frac{\mu_0 x}{M}+\frac{\upsilon_0 y}{N}\right)} = e^{j\pi(x+y)} = (-1)^{x+y}$$

$$\Im[f(x,y)(-1)^{x+y}] = F(\mu - \frac{M}{2}, \upsilon - \frac{N}{2})$$

平移特性

DFT/FFT/IDFT/IFFT实 现

- Python库的调用

傅里叶变换，第一行是将原图进行傅里叶变换，第二行是将变换后的图像进行平移

[43]:
```
fft_unshift = np.fft.fft2(img1)
fft_shift = np.fft.fftshift(fft_unshift)
```

傅里叶逆变换

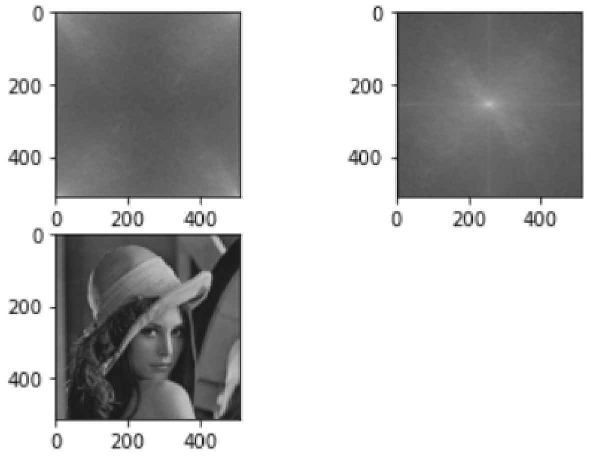
[44]:
```
ifft = np.fft.ifftshift(fft_shift)
ifft_shift = np.fft.ifft2(ifft)
```

DFT/FFT/IDFT/IFFT实 现

其中越靠近中心位置频率越低，越亮（灰度值越高）的位置代表该频率的信息振幅越大。

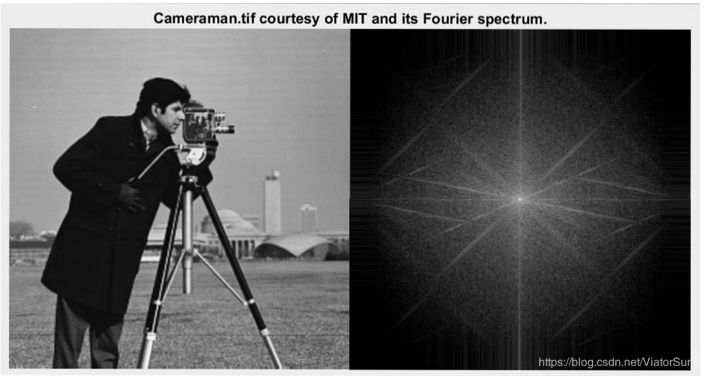
[45]:
```
fig = plt.figure()
ax = fig.add_subplot(221)
plt.imshow(np.log(np.abs(fft_unshift)),cmap = "gray")
#plt.colorbar(shrink = 0.5)
ax = fig.add_subplot(222)
plt.imshow(np.log(np.abs(fft_shift)),cmap = "gray")
ax = fig.add_subplot(223)
plt.imshow(np.abs(ifft_shift),cmap = "gray")
```

[45]: <matplotlib.image.AxesImage at 0x1fe04180790>



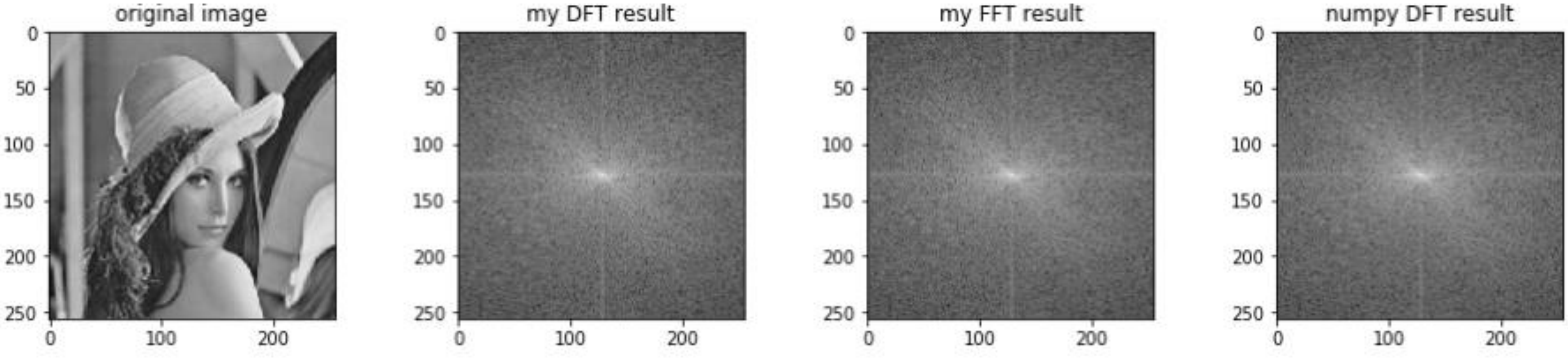
图像快速傅里叶变换

- 频谱原点移动



图像DFT/FFT/IDFT/IFFT

- DFT/FFT



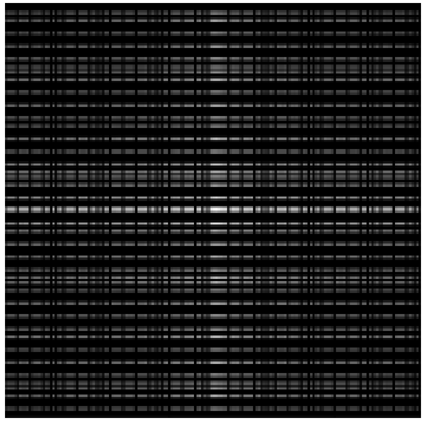
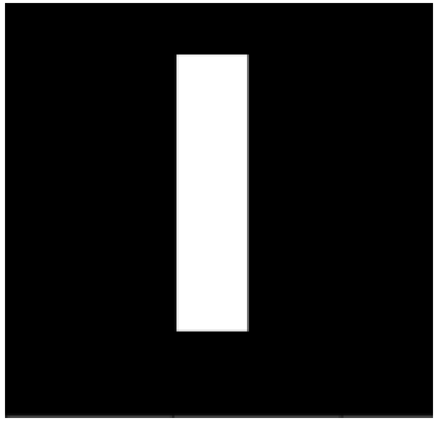
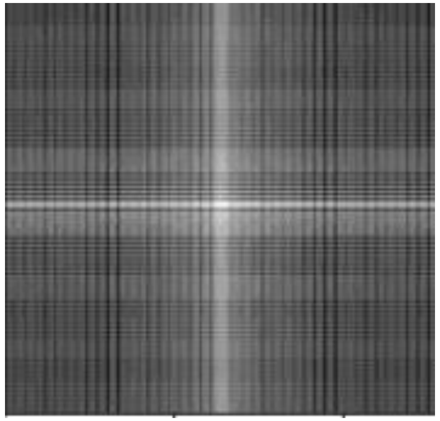
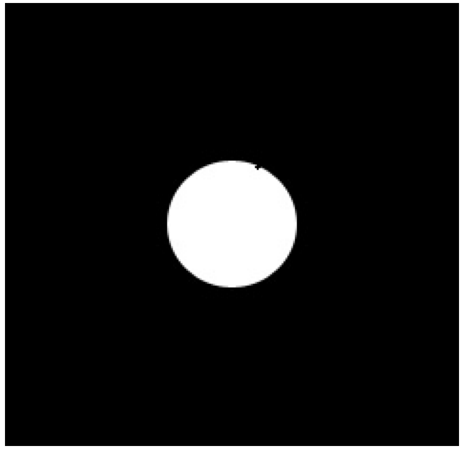
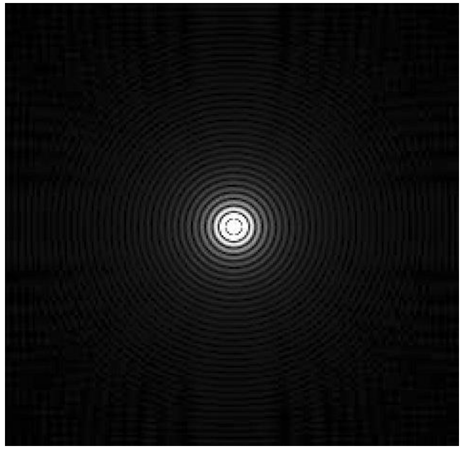
DFT/FFT/IDFT/IFFT实 现

- **基本图形绘制**

```
import matplotlib.pyplot as plt
%matplotlib inline

fig = plt.figure(facecolor='black') #设置背景色

ax = fig.add_subplot(111)

plt.axis('off') #关闭坐标轴

rect = plt.Rectangle((0.2,0.75), 0.4, 0.15, color = 'r', alpha = 0.5)#左下起点，长，宽，颜色，α

circ = plt.Circle((0.7,0.2), 0.15, color = 'b', alpha = 0.5)#圆心，半径，颜色，α

pgon = plt.Polygon([[0.15, 0.15], [0.35, 0.4], [0.2, 0.6]], color = 'g', alpha = 0.5 )

ax.add_patch(rect)

ax.add_patch(circ)

ax.add_patch(pgon)
fig.savefig("G:img.png",bbox_inches='tight', dpi=fig.dpi, pad_inches=0.0, facecolor='black') #保存图片
```

DFT/FFT/IDFT/IFFT实 现

- DFT/FFT

DFT/FFT/IDFT/IFFT实 现

- **滤波**

- 高通滤波：让高频信息通过
- 低通滤波：让低频信息通过

$$H(u,v)=\begin{cases}1, & D(u,v)\le D_0\\ 0, & D(u,v)>D_0\end{cases}$$

一个理想的低通滤波器

$$\mathrm{D}(u,v)=\sqrt{(u-M/2)^2+(v-N/2)^2}$$

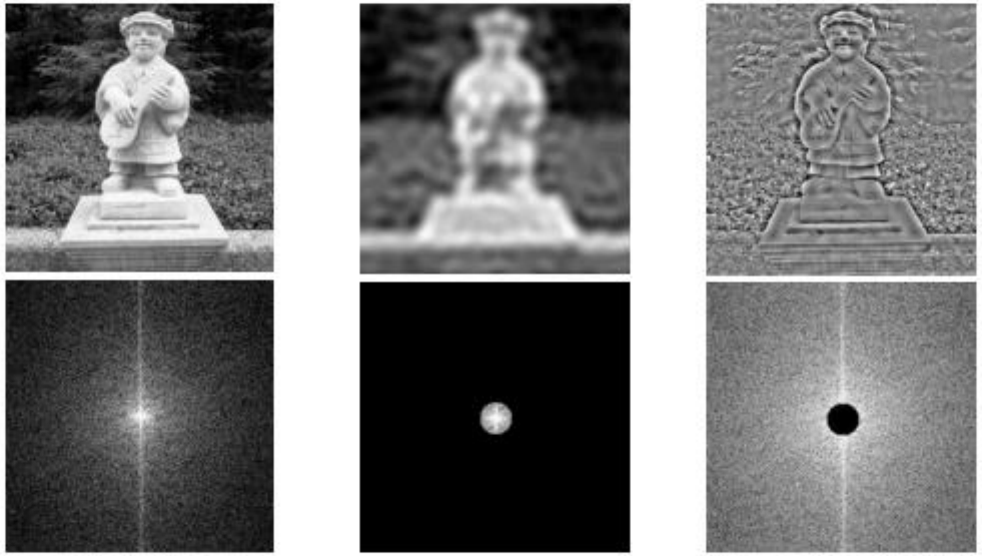
DFT/FFT/IDFT/IFFT实 现

- 频率域增强

1. 计算图像的傅里叶变换
2. 频域结果与某转移函数相乘
3. 逆傅里叶变换

- 方法

1. 低通滤波
2. 高通滤波
3. 带通和带阻滤波
4. 同态滤波



作业-2

- **编程实现（需要简洁的代码注释）**

（1）打开一张灰度图片;

（2）实现DFT与IDFT;

（4）找出（构建）一个经过傅里叶变换后为单一频率的图像，说明思路

（4）实现理想的高通滤波、低通滤波和带通滤波，对比结果，阐述其原理与作用

（5）完成实验报告

注： DFT,IDFT,滤波算法均需要各位自己实现,不可调用现有运算库。代码请写在一个ipynb文件中，以markdown（推荐）或注释来阐明哪些代码实现了作业的哪个部分