离散傅里叶变换与离散余弦变换的实现与图像重构

实验人：李东嵘 16342080

实验目的：实现DFT, DCT算法及其相应逆变换，并以此为基础进行频率域重构

实验环境：Window10系统与Matlab R2016a

原理与方法：

离散傅里叶变换与离散余弦变换是数字图像处理领域常用的变换，它们可以将图像由空域转到频率域，以此为基础对频率域进行相应的操作与变换。

*1.*傅里叶变换的实现

我们将先讨论离散傅里叶变换的实现。

显然，离散傅里叶变换有如下表达式：

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-2j\pi ux/M}e^{-2j\pi vy/N}$$

其计算量及其庞大，如果使用for循环暴力求解的话，其时间复杂度约为N^4，对于单幅数字图像的大小而言，这一速度是无法接受的。

幸运的是，由于上式是若干个元素的线性组合，因此我们可以考虑将其拆分为矩阵形式，然后用MATLAB的矩阵运算功能快速求解傅里叶变换。其具体矩阵形式如下：

$$F = AfB$$

其中，$A_{ij} = e^{-2\pi jux/M}, B_{ij} = e^{-2\pi jvy/N}$. 因此，我们可以通过MATLAB编程实现这两个矩阵并从而快速求解傅里叶变换。具体算法为：

1. 对图像进行适当的预处理，并按照如上思路实现A,B矩阵
2. 计算F=AfB
3. 计算相应的实部，虚部，相位图和频谱

具体代码实现如下：

```
%function [r,im,phi,ang,output]=MyFFT(image)
```

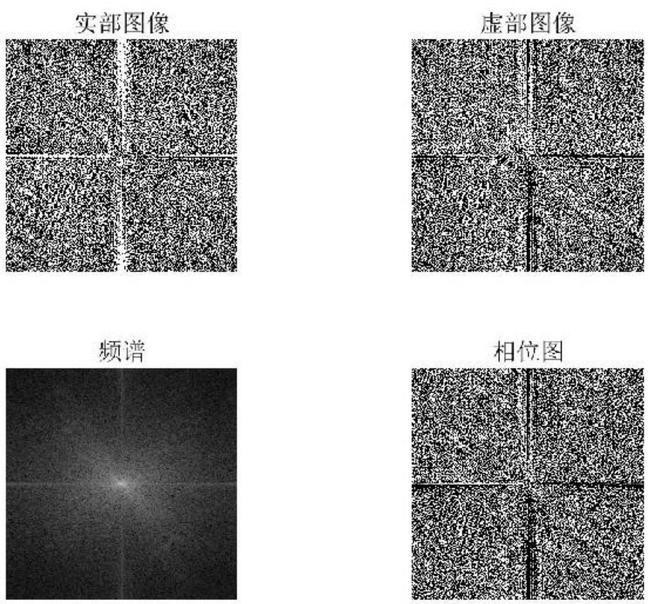
```
%example:
%[a,b,c]=MyFFT(lena);
%imshow(log(1+c),[])
%ang is the angle
%output is the complex image

e=exp(1);

s=size(image);
if size(s)>2
    image=rgb2gray(image);
end

image=im2double(image);

s=size(image);
i_height=s(1);
i_width=s(2);

G1=zeros(i_height,i_height);
G2=zeros(i_width,i_width);

for k=1:i_height
    for j=1:i_width
        image(k,j)=image(k,j)*((-1)^(k+j-2));
    end
end

for k=1:i_height
    for j=1:i_width
        G1(k,j)=e^(-i*2*pi*(k-1)*(j-1)/i_height);
        G2(k,j)=e^(-i*2*pi*(k-1)*(j-1)/i_width);
    end
end

output=G1*image*G2;
size(output);

r=real(output);
im=imag(output);
phi=abs(output);
ang=angle(output);

%end
```

我们将用具体的例子展示该算法的效果。出于数字图像处理的传统，我们使用Lena作为本算法的例子。下面是Lena的原图像：

我们接下来对Lena进行傅里叶变换：

```
[a,b,c,d,e]=MyFFT(lena);
```

于是我们得到了该图像的实部，虚部，频谱和相位图，使用subplot()函数后如下所示：



***2. DFT*特定频谱的还原**

为了开发特定频谱的还原算法，我们先实现傅里叶逆变换算法。傅里叶逆变换由如下公式给出：

$$f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)e^{2\pi jux/M}e^{2\pi jvy/N}$$

同样地，我们一样可以把该公式写成矩阵形式：

$$f = GFH$$

其中

$$G_{ij} = e^{2\pi jux/M}, H_{ij} = e^{2\pi jvy/N}$$

这样我们就实现了傅里叶逆变换。在这一基础上，我们进行高低频率的图像重构。

频率重构问题，从本质上来说是我们企图丢弃某些频段的信号，而保留其他频段的信号不变，对应到实际操作层面上，则是把相应要丢弃的频段对应的傅里叶变换值设置为0.

对于低频率重构问题，我们只保留低频部分，因此我们将图像除中央一个NXN的方块内的信号外，其他值全部置为0，并在此基础上进行傅里叶逆变换，便得到了低频重构的结果。

同样地，对于高频率重构问题，我们将图中央一个NXN方块内的低频信号置为0，其他保持不变，并进行傅里叶逆变换，则可得到高频重构的结果。

低频重构算法如下:

- 输入：傅里叶变换后的图像F
- 输出：用F的低频段重构所得图像f^*
- 具体算法：

1. 对F，只保留其中央NXN子矩阵的值，其余部分值全部置为0
2. 计算相应的G, H
3. 计算$f^* = GFH$

高频重构算法如下:

- 输入：傅里叶变换后的图像F
- 输出：高频段重构的图像f^{high}

1. 对F，将其中央NXN的一个子矩阵置为0，其余部分保持不变
2. 计算相应的G, H
3. 计算$f^{high} = GFH$

我们对Lena的傅里叶频谱进行频率重构。

低频率重构由如下代码实现:

```
%function image=LowRecon(f,scale)

%parameter scale stands for the size of the square we want
%to preserve

e=exp(1);

s=size(f);
i_height=s(1);
i_width=s(2);

%only preserve the signal with low frequency
tmp=f(i_height/2-scale/2:i_height/2+scale/2,i_width/2-scale/2:i_width/2+scale/2);
f=f-f;
f(i_height/2-scale/2:i_height/2+scale/2,i_width/2-scale/2:i_width/2+scale/2)=tmp;

%calculate the matrix
G3=zeros(i_height,i_height);
G4=zeros(i_width,i_width);

for k=1:i_height
    for j=1:i_width
        G3(k,j)=e^(i*2*pi*(k-1)*(j-1)/i_height);
        G4(k,j)=e^(i*2*pi*(k-1)*(j-1)/i_width);
    end
end

image=G3*f*G4;
%end
```

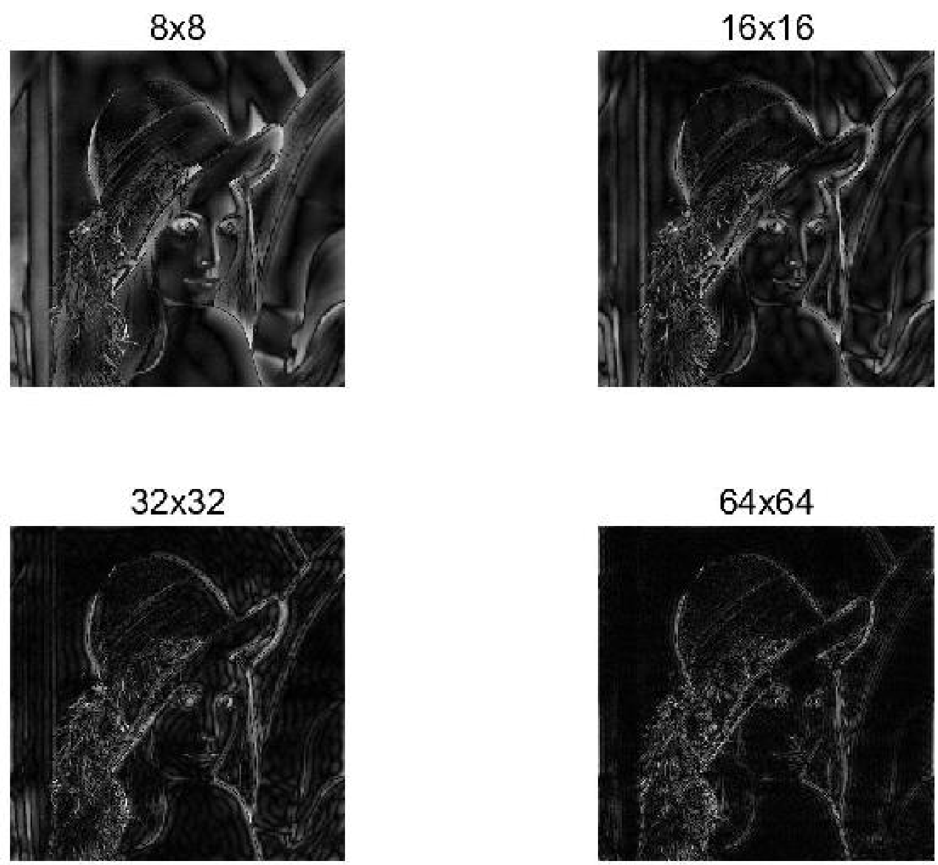
低频率重构效果如下所示:

高频率重构算法由如下代码给出:

```
%function image=HighRecon(f,scale)
%correlated codes remains for same as they are in LowRecon.m

e=exp(1);

s=size(f);
i_height=s(1);
i_width=s(2);


f(i_height/2-scale/2:i_height/2+scale/2,i_width/2-scale/2:i_width/2+scale/2)=0;

G3=zeros(i_height,i_height);
G4=zeros(i_width,i_width);

for k=1:i_height
    for j=1:i_width
        G3(k,j)=e^(i*2*pi*(k-1)*(j-1)/i_height);
        G4(k,j)=e^(i*2*pi*(k-1)*(j-1)/i_width);
    end
end

image=G3*f*G4;
```

```
%end
```

高频重构效果如下所示:



*3. DCT*的实现

我们已经实现了DFT，进一步，我们将对DCT展开研究。DCT由如下公式给出:

$$F(u,v) = c(u)c(v)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} cos(\pi(2x+1)u/(2M))cos(\pi(2y+1)v/(2N))f(x,y)$$

写成矩阵形式:

$$F = GfH$$

其中

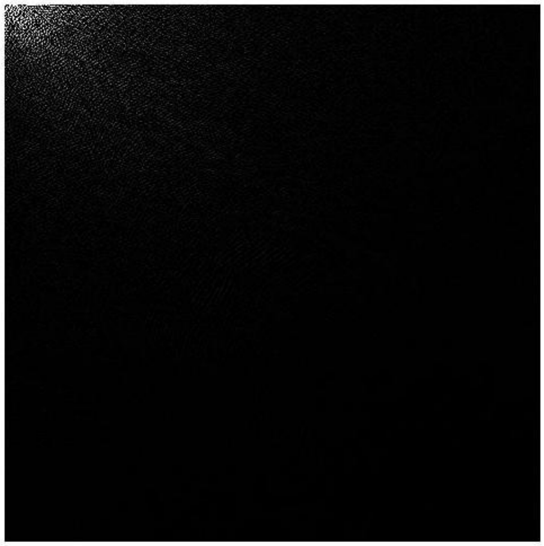
$$G_{ij} = c(i)cos((2j+1)i\pi/2M)$$

$$H_{ij} = c(j)cos((2i+1)j\pi/2N$$

和上面一样，我们通过编程实现相应的矩阵，具体MATLAB代码如下：

```
%function output=MyDCT(image)

e=exp(1);

%detect if it is a greyscale image
s=size(image);
if size(s)>2
    image=rgb2gray(image);
end

image=im2double(image);

s=size(image);
i_height=s(1);
i_width=s(2);

%generate c(x)
   % function y=c(u,n)
        if u==0
            y=sqrt(1/n);
        else
            y=sqrt(2/n);
        end
   % end

%obtain the matrix we desire
G1=zeros(i_height,i_height);
G2=zeros(i_width,i_width);

for k=1:i_height
    for j=1:i_height
        G1(k,j)=c(k-1,i_height)*cos(((2*j-1)*pi/(2*i_height))*(k-1));
    end
end

for k=1:i_width
    for j=1:i_width
        G2(k,j)=c(j-1,i_width)*cos(((2*k-1)*pi/(2*i_width))*(j-1));
    end
end

%calculate DCT
output=G1*image*G2;

%end
```

对Lena进行DCT变换，效果如下：



*4.DCT*的频率重构

我们可以对DCT进行相应频率的重构，首先我们要实现DCT的逆变换，DCT的逆变换可以直接写成如下矩阵形式：

$$f = AFB$$

其中

$$A_{ij} = c(j)cos((2i + 1)\pi j/(2M))$$

$$B_{ij} = c(i)cos((2j + 1)\pi i/(2N))$$

当进行频率重构时，其思路和上面的DFT频率重构是类似的。

当我们要进行低频重构时，我们只保留F左上角的一个NXN的正方形（即低频率部分）,其他部分置为0。

同理，当我们想进行高频重构时，我们将F左上角一个NXN的正方形置为0，其他部分保持不变。

低频重构算法由如下程序实现：

```
%function image=DCTLowRecon(f,scale)

%obtain the size of f
s=size(f);
i_height=s(1);
i_width=s(2);

%only preserve the area with low frequency
```

```
tmp=f(1:scale,1:scale);
f=f-f;
f(1:scale,1:scale)=tmp;

%define the matrix
G1=zeros(i_height,i_height);
G2=zeros(i_width,i_width);

%generate c(x)
    %function y=c(u,n)
        if u==0
            y=sqrt(1/n);
        else
            y=sqrt(2/n);
        end
    %end

%generate the matrix G and H
for k=1:i_height
    for j=1:i_height
        G1(k,j)=c(j-1,i_height)*cos((2*k-1)*pi*j/(2*i_height));
    end
end

for k=1:i_width
    for j=1:i_width
        G2(k,j)=c(k-1,i_width)*cos((2*j-1)*pi*k/(2*i_width));
    end
end

%obtain the reconstructed image
image=G1*f*G2;

%end
```

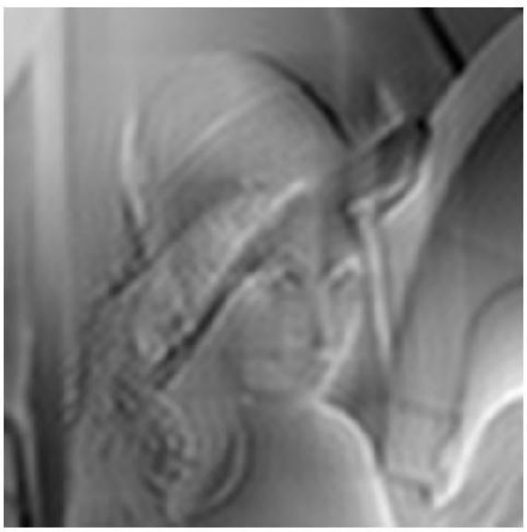
高频重构算法实现如下所示:

```
%function image=DCTHighRecon(f,scale)
%for code annotations, refer to the corresponding part in DCTHighRecon()

s=size(f);
i_height=s(1);
i_width=s(2);

%set areas with low frequency be 0
f(1:scale,1:scale)=0;

G1=zeros(i_height,i_height);
G2=zeros(i_width,i_width);

   % function y=c(u,n)
        if u==0
            y=sqrt(1/n);
        else
            y=sqrt(2/n);
        end
```

```
    %end

for k=1:i_height
    for j=1:i_height
        G1(k,j)=c(j-1,i_height)*cos((2*k-1)*pi*j/(2*i_height));
    end
end

for k=1:i_width
    for j=1:i_width
        G2(k,j)=c(k-1,i_width)*cos((2*j-1)*pi*k/(2*i_width));
    end
end

image=G1*f*G2;

%end
```

我们考察重构的效果, 64x64低频重构效果如图所示:

64x64



高频16x16重构效果如下图所示:

至此，我们已完成了DFT, DCT的实现，其相应的频率重构。本算法仍有改进空间，如我们可以用快速算法代替矩阵运算。由于本次作业涉及到的图像均较小，因此两算法间可能差异不大，但当图像进一步增大时，相信快速算法将会有更优良的效果。