IE307 视频编码与通信〇作业一

> 1.选择至少一张图片，分别进行DFT和DCT正反变换，观察并简单分析结果。
>
> 2.选择一张图片，将其按8x8分块，对每一块分别作8x8的2D-DCT变换， 并保留左上角前6六条对角线上的系数（其余置0）后作8x8的反变换， 比较得到的图像与原图像并分析。
>
> 3.选择两张大小相同的图像，分别进行DFT变换后，置换两幅图像的幅度和相位信息后再作反变换，观察并分析结果。

由于自己实现的函数效率太低，在处理较大尺寸的图片时耗时较长，不便于观察程序正确性，因此在接下来的部分中，我统一选择了64 * 64大小的图像用于处理。

Task 1

实验任务

> 选择至少一张图片，分别进行DFT和DCT正反变换，观察并简单分析结果。

实验原理

傅里叶变换是一种线性积分变换积分变换，用于信号在时域和频域之间的变换，在物理学和工程学中有许多应用。因其基本思想首先由法国学者约瑟夫·傅里叶系统地提出，所以以其名字来命名以示纪念。实际上傅里叶变换就像化学分析，确定物质的基本成分；信号来自自然界，也可对其进行分析，确定其基本成分。

在信号处理领域，可以将傅里叶变换用于图像压缩。由于图像/视频为数字信号，因此使用傅里叶变换的离散形式（DFT）。将离散信号变换到频域后，适当删减掉高频信号（人眼不太敏感的部分），再反变换回时域，可以有效压缩图像的大小，但是视觉效果与原先相差无几。

2-D DFT变换的公式如下:

$$F[k,l] = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f[m,n]e^{-j2\pi(\frac{km}{M})+\frac{ln}{N}}, k=0,1,\ldots,M-1; l=0,1,\ldots,N-1$$

对应的反变换公式如下:

$$f[m,n] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} F[k,l]e^{j2\pi(\frac{km}{M})+\frac{ln}{N}}, m=0,1,\ldots,M-1; n=0,1,\ldots,N-1$$

类似的变换方法还有离散余弦变换（DCT）。DCT只使用实数。

2-D DCT变换的公式如下:

$$X[k,l] = C(k)C(l)\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x[m,n]cos[\frac{(2m+1)k\pi}{2N}]cos[\frac{(2n+1)l\pi}{2N}], k,l=0,1,\ldots,N-1$$

对应的反变换公式如下:

$$x[m,n] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} C(k)C(l)X[k,l]cos[\frac{(2m+1)k\pi}{2N}]cos[\frac{(2n+1)l\pi}{2N}], k,l=0,1,\ldots,N-1, m,n=0,1,\ldots,N-1$$

$$C(k)=C(l)=\begin{cases}\sqrt{\dfrac{1}{N}} & k,l=0\\ \sqrt{\dfrac{2}{N}} & otherwise\end{cases}$$

实验过程

根据DFT和DCT公式，实现了`matlab`版的代码。由于对每一个像素都需要遍历整张图才能得到变换值，所以总体为四重循环，复杂度为$O(n^4)$。

核心逻辑见下:

DFT

```
t = zeros(M, N);
t = double(t);
for m = 0:M-1
    for n = 0:N-1
        for p = 0:M-1
            for q = 0:N-1
                t(m+1,n+1) = t(m+1,n+1) + double(o(p+1,q+1)) * exp(((-2i)*pi)*(double(m*p)/M+double(n*q)/N));
            end
        end
        t(m+1,n+1) = t(m+1,n+1) / (M*N);
    end
end
```

IDFT

```
t = double(zeros(M, N));
for m = 0:M-1
    for n = 0:N-1
        for p = 0:M-1
            for q = 0:N-1
                t(m+1, n+1) = t(m+1, n+1) + double(o(p+1,q+1)) * exp(((2i)*pi)*((double(m*p)/M)+(double(n*q)/N)));
            end
        end
    end
end
```

DCT

```
t = double(zeros(M, N));
for m = 1:M
    for n = 1:N
        for i = 1:M
            for j = 1:N
                t(m, n) = t(m, n) + o(i, j) * cos((2 * i - 1) * (m - 1) * pi / (2 * M)) * cos((2 * j - 1) * (n - 1) * pi / (2 * N));
            end
        end
        if (m==1)
            ci = 1/sqrt(M);
        else
            ci = sqrt(2/M);
        end
        if (n==1)
            cj = 1/sqrt(N);
        else
            cj = sqrt(2/N);
        end
        t(m, n) = t(m, n) * ci * cj;
    end
end
```

IDCT

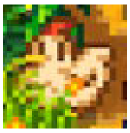
```
t = double(zeros(M, N));
for m = 1:M
    for n = 1:N
        for i = 1:M
            for j = 1:N
                if (i==1)
                    ci = 1/sqrt(M);
                else
                    ci = sqrt(2/M);
                end
                if (j==1)
                    cj = 1/sqrt(N);
                else
                    cj = sqrt(2/N);
                end
                t(m, n) = t(m, n) + ci * cj * o(i, j) * cos((2 * m - 1) * (i - 1) * pi / (2 * M)) * cos((2 * n - 1) * (j - 1) * pi / (2 * N));
            end
        end
```

```
    end
end
```

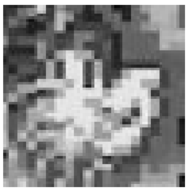
结果分析

DFT

接下来，对该图分别进行DFT变换分析：



首先变换为灰度图像：



然后将灰度图像进行DFT变换，并将变换后的结果进行反变换，得到还原的图像：

可以看到和原图几乎没有任何区别。

（由于进行DFT变换后的数据为复数，因此不展示中间结果）

DCT

接下来，对该图进行变换分析：

首先转化为灰度图像：

进行DCT变换，得到下图：

对中间结果进行IDFT，得到还原的图像：

可以看到和原图几乎没有任何区别。

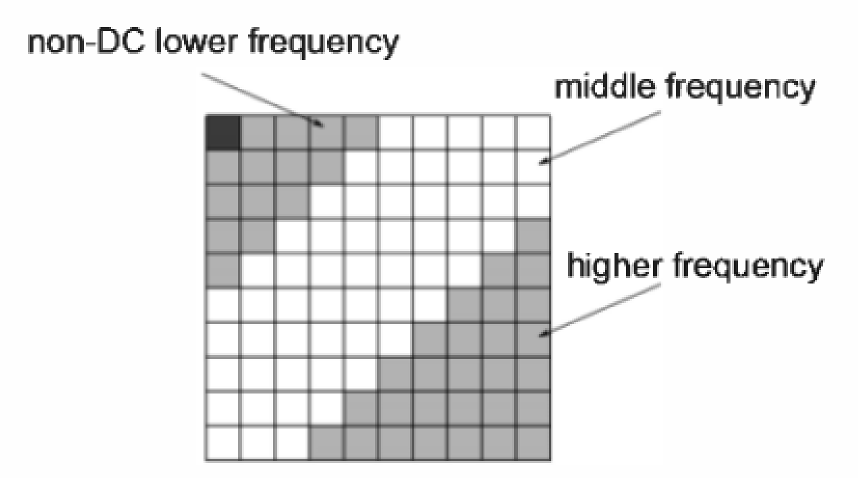
Task 2

试验任务

> 选择一张图片，将其按8x8分块，对每一块分别作8x8的2D-DCT变换， 并保留左上角前6六条对角线上的系数（其余置0）后作8x8的反变换， 比较得到的图像与原图像并分析。

实验原理

DCT变换后图像的信息密度是不均匀的，从左上角到右下角分别是从低频分量到高频分量过度，左上角的第一个值为直频分量。根据人眼的视觉特性，靠近变换后图像中左上部分的值表示了画面的主要部分，在原始图像中被人眼感知到的成分更多，相对而言更为重要；而右下部分主要是一些细节，对于人眼而言并不是特别敏感。因此如果适当地沿对角线删减掉部分信息，反变换之后仍然可以很大程度上还原出原始图像的信息，这也是DCT用于图像压缩的原理。



实验过程

使用了matlab自带的blockprochh函数辅助进行分块处理。因此，需要完成blockproc中的函数句柄。

在自定函数中，主要完成了三部分工作：

- 对输入的块进行DCT变换；
- 将变换后的矩阵进行一些处理（将某些反对角线元素设为零）；
- 将处理过的矩阵进行IDCT变换。

其中，DCT和IDCT已经在`Task1`中描述过了，重点看对反对角线元素的处理：

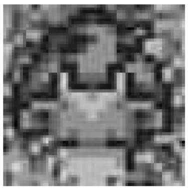
```
% ... DCT transformation ...

for i = 0:M-1
    for j = 0:N-1
        if i+j >= 6
            inter(i+1, j+1) = 0;
        end
    end
end

% ... IDCT transformation ...
```

结果分析

接下来采用该图片进行测试：

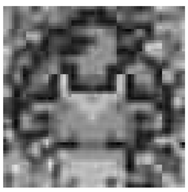
转化为灰度图像：

保留前6条对角线



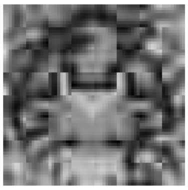
看上去和灰度图像差别不大。

保留前5条对角线



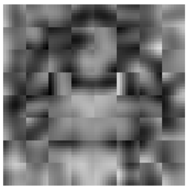
已经出现了块之间的分界线。

保留前4条对角线



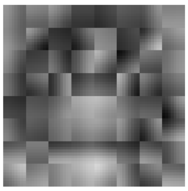
块效应已经比较明显了。

保留前3条对角线



只能看出大致的轮廓；仔细看每个块内，都只有模糊的一道线。

保留前2条对角线



基本上看不出什么信息。

保留直流分量

已经完全退化成了8*8的马赛克图像，每个块退化为了一个像素。

Task 3

实验任务

> 选择两张大小相同的图像，分别进行DFT变换后，置换两幅图像的幅度和相位信息后再作反变换，观察并分析结果。

实验原理

对图像进行傅里叶变换后，图像的幅度表示为$A(f)=\sqrt{Re(f)^2+Im(f)^2}$，相位表示为$\varphi(f)=\arctan\frac{Imf}{Ref}$。理论上，幅度信息主要是图像地纹理，而相位信息更多地反应了图像的轮廓，且人眼对这些信息更敏感。因此，如果将两幅尺寸相同的图像进行处理，并保留各自的幅度信息但交换各自的相位信息，之后进行反变换，得到的图像应该是保留了原本图片的纹理，但是带有另一张图片的轮廓。

实验过程

类似于`Task2`，此部分的任务也分为3步：

- 对两幅图片进行DFT变换；
- 保留DFT变换后图像矩阵各自的幅度，但交换相位信息；
- 对处理过的矩阵进行IDFT变换。

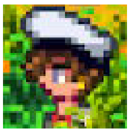
省略DFT和IDFT的部分，核心代码逻辑如下：

```
% ... DFT transformation ...

% extract the amplitude and phase of 2 images

abs1 = abs(inter1);
angle1 = angle(inter1);
abs2 = abs(inter2);
angle2 = angle(inter2);

% keep origin amplitude but exchange the phase

```

```
t1 = abs1.*cos(angle2) + abs1.*sin(angle2).*1i;
t2 = abs2.*cos(angle1) + abs2.*sin(angle1).*1i;

% ... IDFT transformation ...
```

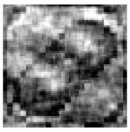
结果分析

对这两张图进行相位交换：



首先转换为灰度图像：

交换了幅度信息并反变换后，得到了变换后的结果：



可以看到，两张图片保留了各自的纹理，但交换了轮廓，这与实验前的估计一致，说明了DFT变换后的矩阵中的幅度主要保存图像的纹理细节，而相位主要保留图像的轮廓细节。