**快速傅里叶变换的原理及其应用**

**摘要:**

快速傅氏变换（FFT），是离散傅氏变换的快速算法，它是根据离散傅氏变换的奇、偶、虚、实等特性，对离散傅立叶变换的算法进行改进获得的。它对傅氏变换的理论并没有新的发现，但是对于在计算机系统或者说数字系统中应用离散傅立叶变换，可以说是进了一大步。　傅里叶变换的理论与方法在“数理方程”、“线性系统分析”、“信号处理、仿真”等很多学科领域都有着广泛应用,由于计算机只能处理有限长度的离散的序列,所以真正在计算机上运算的是一种离散傅里叶变换.　虽然傅里叶运算在各方面计算中有着重要的作用，但是它的计算过于复杂，大量的计算对于系统的运算负担过于庞大，使得一些对于耗电量少，运算速度慢的系统对其敬而远之，然而，快速傅里叶变换的产生，使得傅里叶变换大为简化，在不牺牲耗电量的条件下提高了系统的运算速度，增强了系统的综合能力，提高了运算速度，因此快速傅里叶变换在生产和生活中都有着非常重要的作用，对于学习掌握都有着非常大的意义。

**关键词**：快速傅氏变换；图像处理；matlab

**前言**：

傅里叶变换在信号处理中具有十分重要的作用，但是基于离散时间的傅里叶变换具有很大的时间复杂度，根据傅里叶变换理论，对一个有限长度且长度为N的离散信号，做傅里叶变换的时间复杂度为，当N很大时，其实现的时间是相当惊人的（比如当N为时，其完成时间为

(为计算机的时钟周期） ），故其实现难度是相当大的，同时也严重制约了DFT在信号分析中的应用，故需要提出一种快速的且有效的算法来实现。  正是鉴于DFT极其复杂的时间复杂度，1965年..JWCooley和..JWTukey巧妙地利用 NW因子的周期性和对称性，提出了一个DFT的快速算法，即快速傅里叶变换（FFT），从 而使得DFT在信号处理中才得到真正的广泛应用。

# 傅立叶变化的原理；

## （1）原理

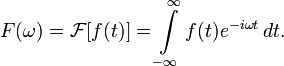
正交级数的展开是其理论基础！将一个在时域收敛的函数展开成一系列不同频率谐波的叠加，从而达到解决周期函数问题的目的。在此基础上进行推广，从而可以对一个非周期函数进行时频变换。

从分析的角度看，他是用简单的函数去逼近（或代替）复杂函数，从几何的角度看，它是以一族正交函数为基向量，将函数空间进行正交分解，相应的系数即为坐标。从变幻的角度的看，他建立了周期函数与序列之间的对应关系；而从物理意义上看，他将信号分解为一些列的简谐波的复合，从而建立了频谱理论。

当然Fourier积分建立在傅氏积分基础上，一个函数除了要满足狄氏条件外，一般来说还要在积分域上绝对可积，才有古典意义下的傅氏变换。引入衰减因子e^(-st)，从而有了Laplace变换。

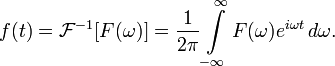
## （2）计算方法

连续傅里叶变换将平方可积的函数f（t）表示成复指数函数的积分或级数形式。



这是将频率域的函数F(ω)表示为时间域的函数f（t）的积分形式。

连续傅里叶变换的逆变换 (inverse Fourier transform)为



即将时间域的函数f（t）表示为频率域的函数F(ω)的积分。

一般可称函数f（t）为原函数，而称函数F(ω)为傅里叶变换的像函数，原函数和像函数构成一个傅里叶变换对（transform pair）。

**实例应用：**

例一

平稳信号：

x=2\*sin(2\*pi\*20\*t)+4\*sin(2\*pi\*60\*t)+8\*cos(2\*pi\*90\*t)+10\*sin(2\*pi\*120\*t)

利用Matlab语言编写的数字图像处理的例程如下：

clc

clear all

fs=100;N=128; %采样频率和数据点数

n=0:N-1;t=n/fs; %时间序列

x=2\*sin(2\*pi\*20\*t)+4\*sin(2\*pi\*60\*t)+8\*cos(2\*pi\*90\*t)+10\*sin(2\*pi\*120\*t); %信号

y=fft(x,N); %对信号进行快速Fourier变换

mag=abs(y); %求得Fourier变换后的振幅

f=n\*fs/N; %频率序列

subplot(2,2,1),plot(t,x);

xlabel('时间/s');

ylabel('振幅');title('滤波前时域图');

subplot(2,2,2);

plot(f,mag);

xlabel('频率/Hz');

ylabel('振幅');title('滤波前频域图');

N1=11;

wc=0.5;

hd=fir1(N1,wc);

z=filter(hd,1,x);

subplot(2,2,3);

plot(t,z);

xlabel('时间/s');

ylabel('振幅');title('滤波后时域图');

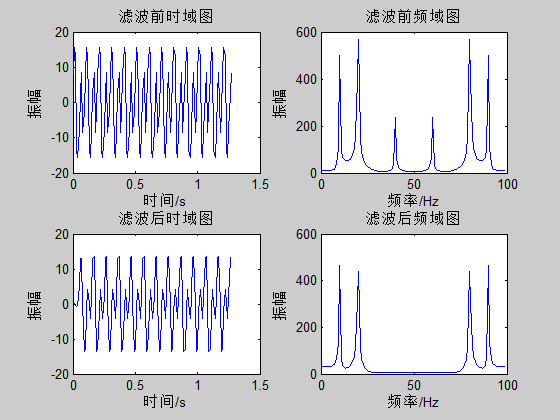
subplot(2,2,4);

plot(f,abs(fft(z)));

xlabel('频率/Hz');

ylabel('振幅');title('滤波后频域图');

在matlab中运行后，实验结果如图：



例二：

**（一）对原图像进行傅立叶变换，实验结果如图1：**

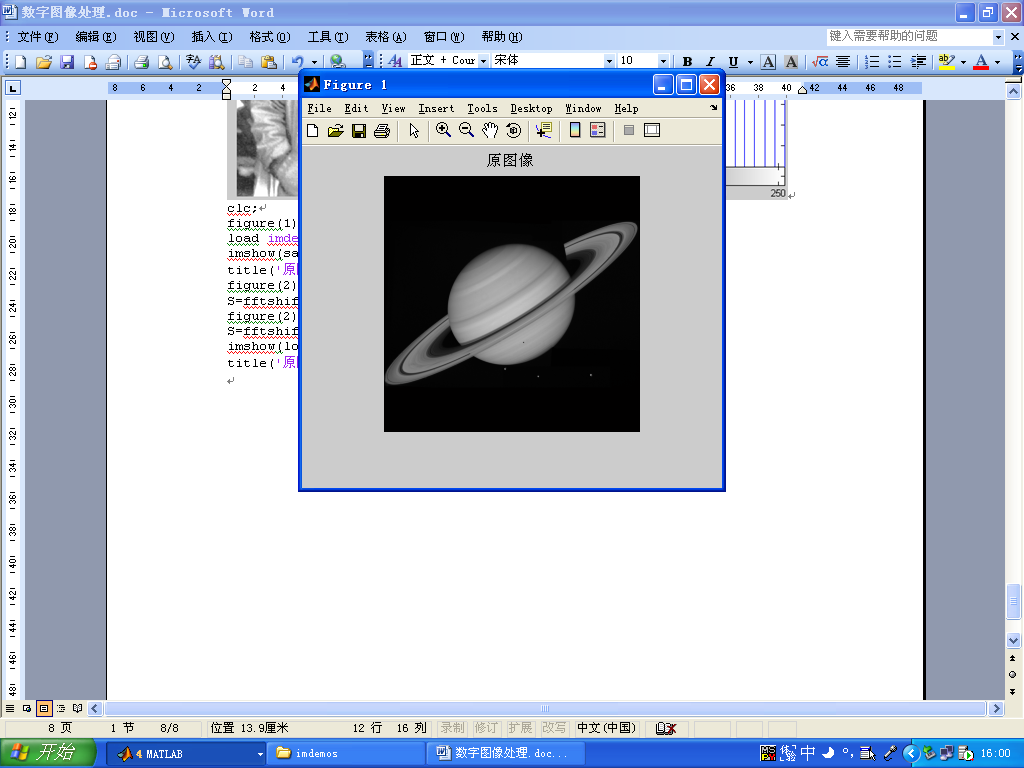
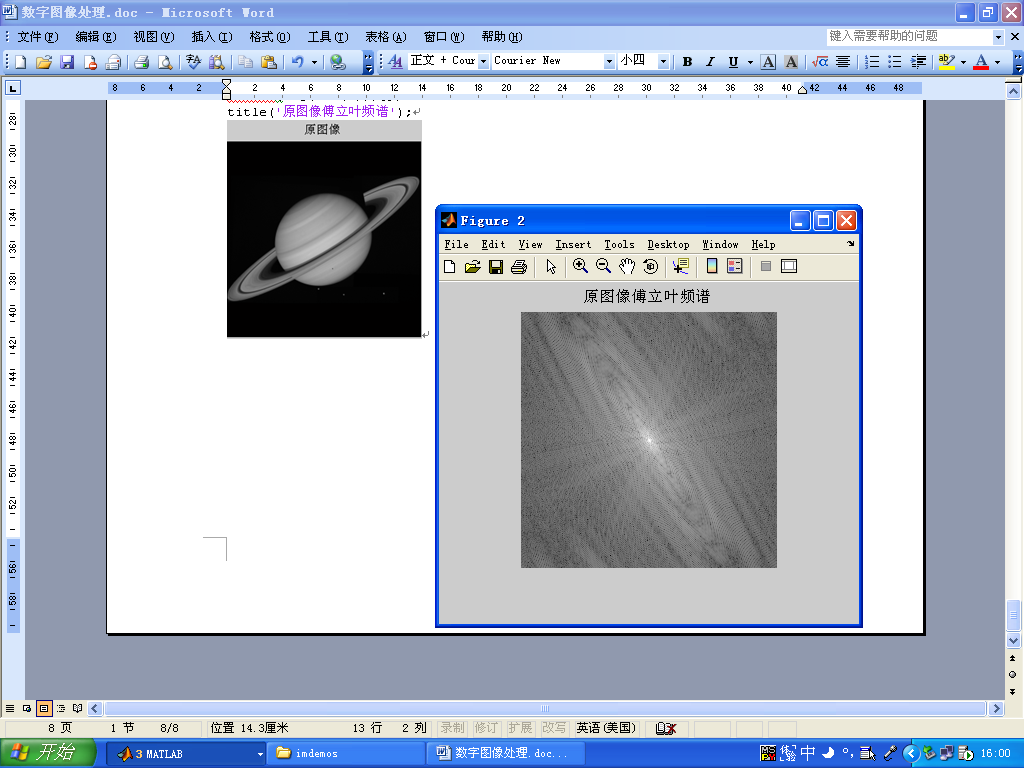
 

图1

分析：图像显示了原图像及其傅立叶频谱。观察傅立叶谱中心对称，在此图像进行傅立叶变换的计算之前被乘以，以此增强了灰度级细节。

（二）输出彩色图像greens.jpg的傅立叶频谱，实验结果如图2：

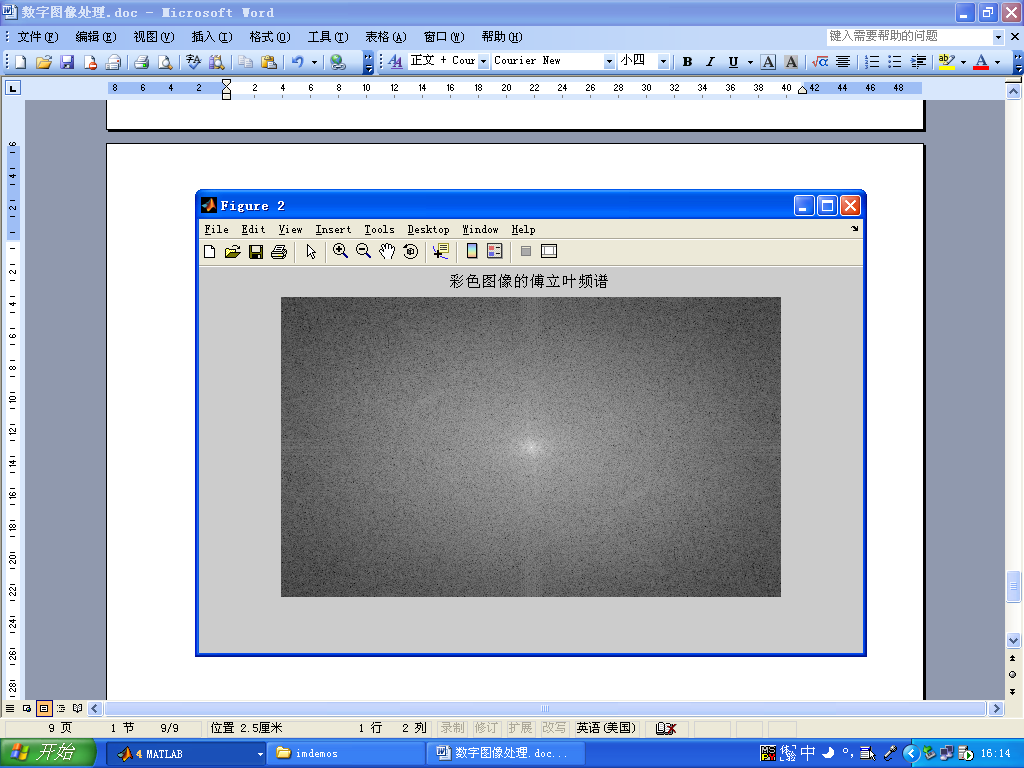
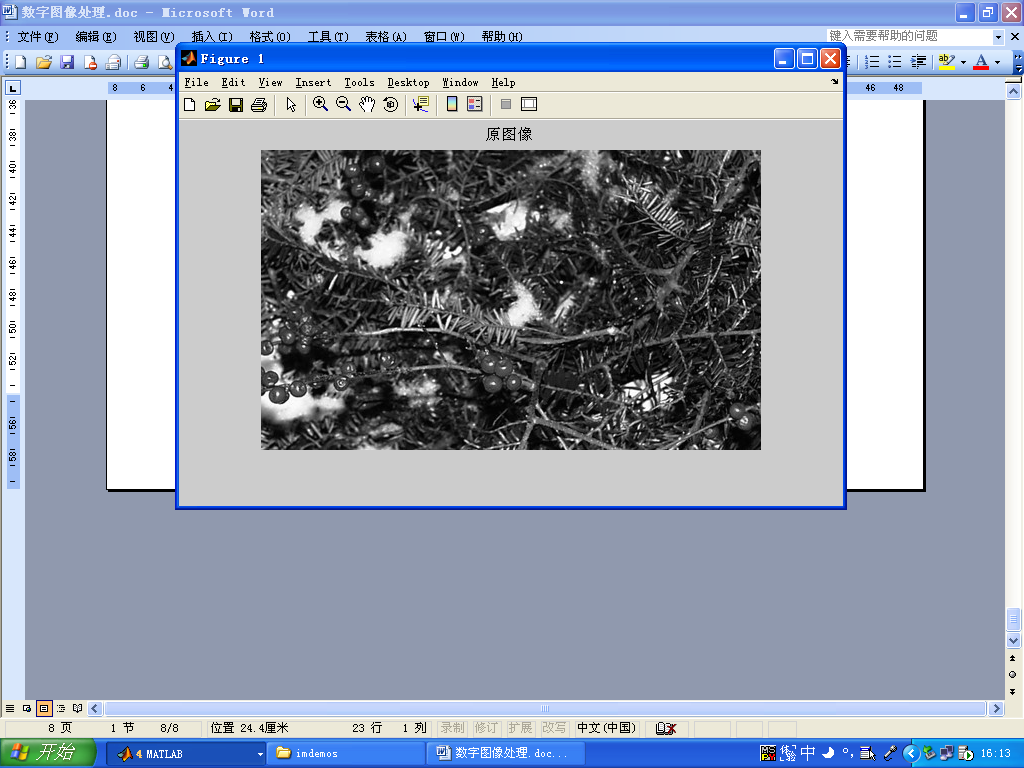


图2

分析：图像显示了原图像和其彩色图像傅立叶频谱。可以看出图像的频率分布是以原点为圆心，对称分布的。变换之后的图像在原点平移之前四角是低频，最亮，平移之后中间部分是低频，最亮，亮度大说明低频的能量大（幅角比较大）

（三）对彩色图像football.jpg进行二维DCT变换，实验结果如图3:

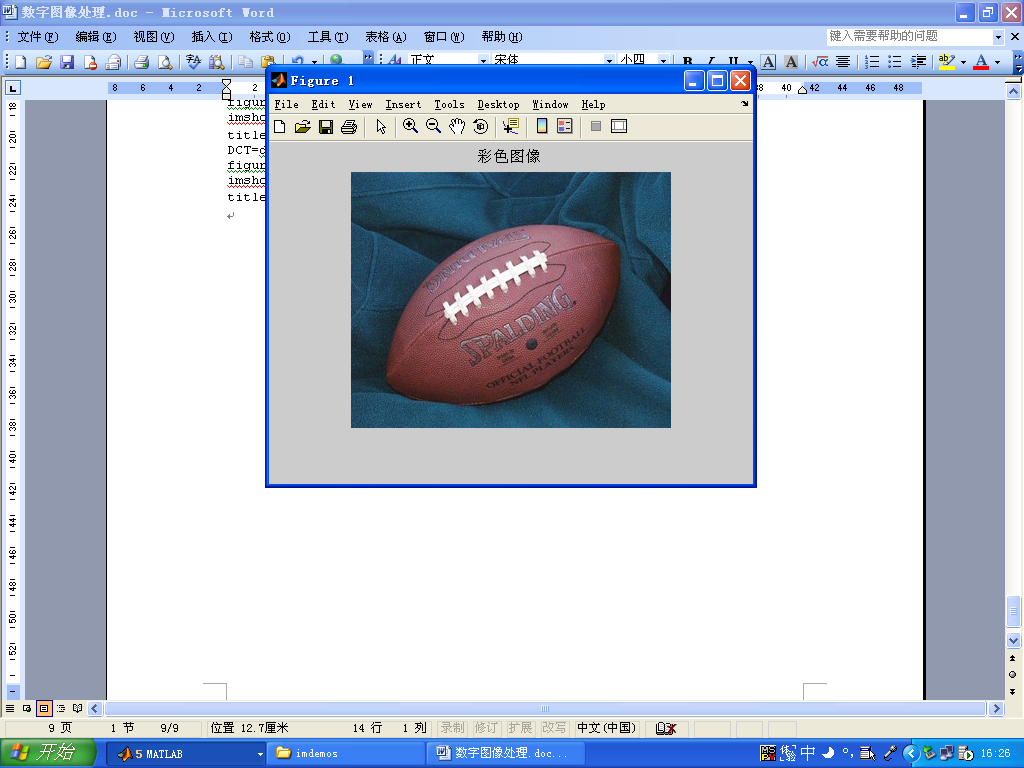
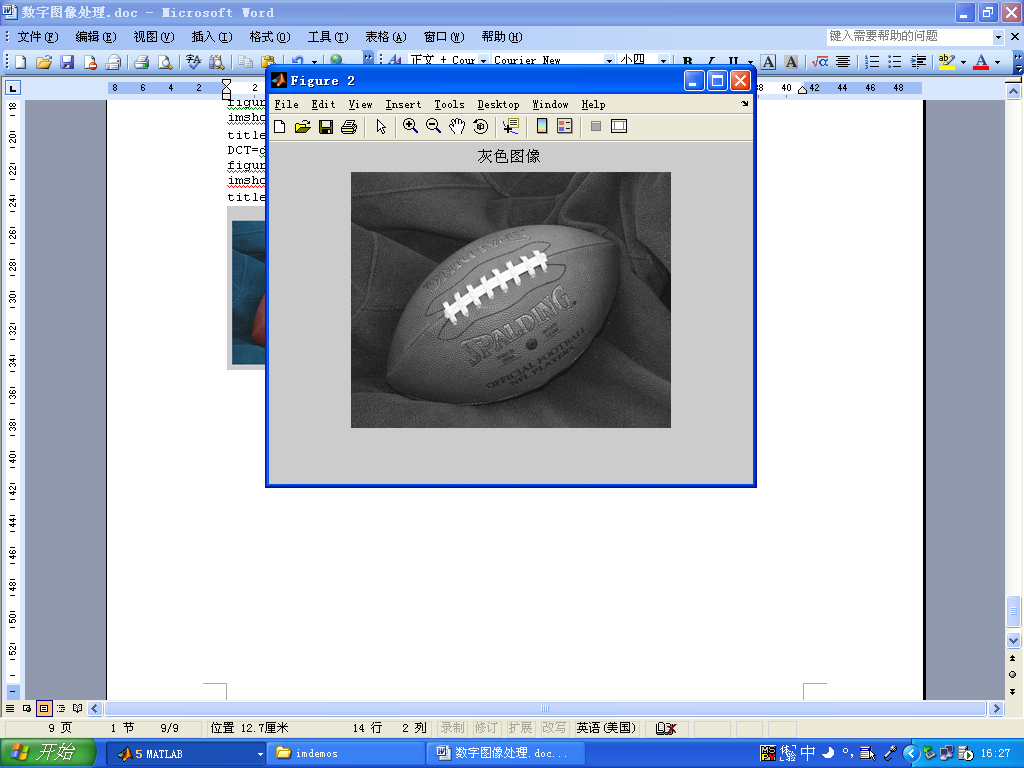
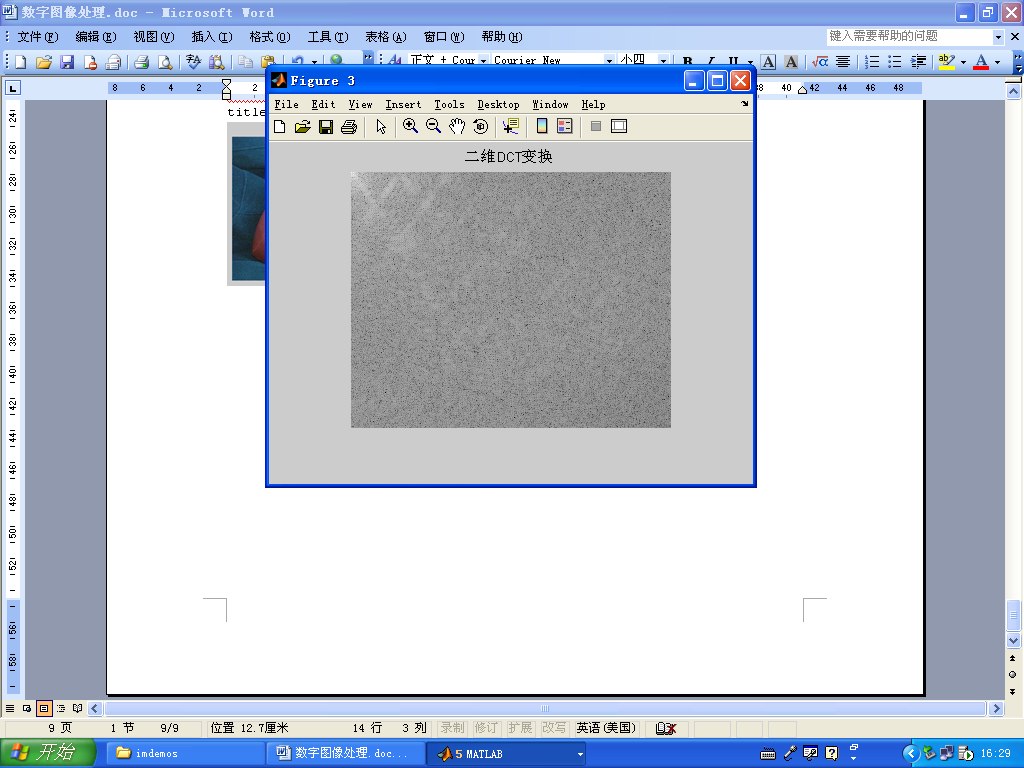
  

图3

分析：二维DCT变换后的频谱图亮点在左上角。

利用Matlab语言编写的数字图像处理的例程如下：

%傅立叶变换Matlab图像的DFT

clc;

figure(1);

load imdemos saturn2;

imshow(saturn2);

title('原图像');

figure(2);

S=fftshift(fft2(saturn2));

figure(2);

S=fftshift(fft2(saturn2));

imshow(log(abs(S)),[]);

title('原图像傅立叶频谱');

%彩色图像的傅立叶频谱

figure(1);

A=imread('greens.jpg');

B=rgb2gray(A);

imshow(B);

title('原图像');

S=fftshift(fft2(B));

figure(2);

imshow(log(abs(S)),[]);

title('彩色图像的傅立叶频谱');

%二维DCT变换

RGB=imread('football.jpg');

figure(1);

imshow(RGB);

title('彩色图像');

GRAY=rgb2gray(RGB);

figure(2);

imshow(GRAY);

title('灰色图像');

DCT=dct2(GRAY);

figure(3);

imshow(log(abs(DCT)),[]);

title('二维DCT变换');

例三：

利用Matlab语言编写的数字图像处理的例程如下:

clc

clear all

x=imread('C:\Users\kj\Desktop\1.jpg');

X=rgb2gray(x);

subplot(3,2,1);

imshow(X);

title('原图');

y=fft2(X);

Y=log(1+abs(y));

subplot(3,2,2);

imshow(Y,[]);

title('傅里叶变化图');

[m,n]=size(y);

for i=1:m

for j=1:n

if i>m/2-140 && i<m/2+140 && j>n/2-250 && j<n/2+250

y1(i,j)=y(i,j);

else

y1(i,j)=0;

end

end

end

subplot(3,2,3);

imshow(log(1+abs(y1)),[]);

title('对傅里叶变化后处理图');

z1=real(ifft2(y1));

Z1=uint8(z1);

subplot(3,2,4);

imshow(Z1);

title('对频率图处理之后的反变化图');

for i=1:m

for j=1:n

if i>m/2-140 && i<m/2+140 && j>n/2-250 && j<n/2+250

y2(i,j)=0;

else

y2(i,j)=y(i,j);

end

end

end

subplot(3,2,5);

imshow(log(1+abs(y2)),[]);

title('对傅里叶变化后处理图');

z2=real(ifft2(y2));

Z2=uint8(z2);

subplot(3,2,6);

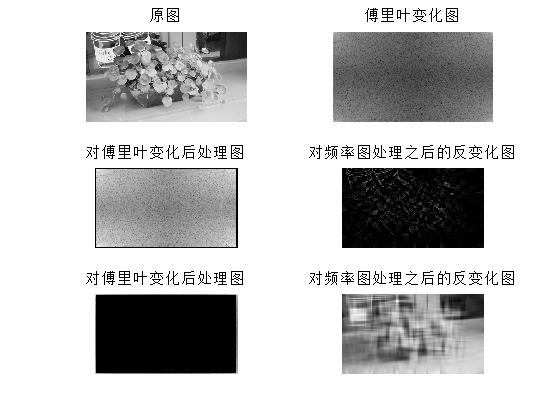
imshow(Z2);

title('对频率图处理之后的反变化图');

在matlab中运行后，实验结果如图：

原图





例四：

Matlab源程序如下：

clc

clear all

img=imread('Dolphin.jpg');

 subplot(2,2,1),imshow(img);

 title('原图');

f=rgb2gray(img);

F=fft2(f);

subplot(2,2,2),imshow(F);

title('傅里叶变换');

 %二维傅里叶变换

FS=fftshift(F);%频率图 %频谱  S=log(1+abs(FS));

subplot(2,2,3);imshow(S,[])

 title('直接变换频谱图');

%二维傅里叶逆变换

fr=real(ifft2(ifftshift(FS)));

%频域的图反变

ret=im2uint8(mat2gray(fr));

%取其灰度图

subplot(2,2,4),imshow(ret);

title('逆傅里叶变换');

I=imread('logo.tif');

figure(2);

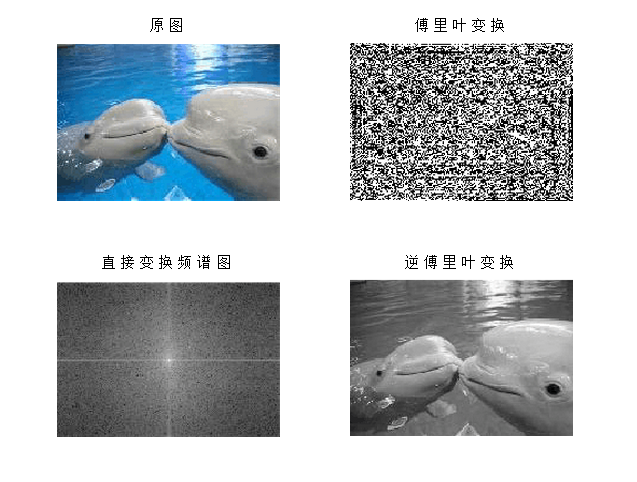
imshow(I);

DCT=dct2(I);

figure(3);

imshow(log(abs(DCT)),[ ]);

在matlab中运行后，实验结果如图：



**总结：**

因各个科学技术领域广泛的使用了FFT技术，它大大推动了信号处理技术的进步，现已成为数字信号处理强有力的工具，本论文将比较全面的叙述快速傅里叶变换算法原理、特点，并完成了基于MATLAB的实现。

**参考文献：**

【1】曾光奇 工程测试技术 华中科技大学出版社