**暨南大学本科实验报告专用纸**

课程名称 数字图像处理实验 成绩评定

实验项目名称 图像的傅立叶变换 指导教师 刘晓翔

实验项目编号 08060275 实验项目类型 实验地点

学生姓名 陈旭天 学号 2021100733

学院 智能科学与工程学院 系 人工智能 专业

实验时间 2023 年 11 月 1 日 午～ 月 日 午 温度 ℃湿度

1. **实验目的**

参见具体的各个实验项目。  
①了解图像变换的意义和手段；②熟悉傅里叶变换的公式和基本性质；③掌握基本傅立叶变换（FT）及其反变换的编程方法；④掌握快速傅立叶变换（FFT）及其反变换的编程方法。

1. **实验内容和要求**

参见具体的各个实验项目。

利用Visual C++6.0软件开发工具编写程序，根据FT、FFT算法生成256灰度图像频谱图，根据FT、FFT反变换实现频域数据到原图像的恢复，程序执行结果正确。

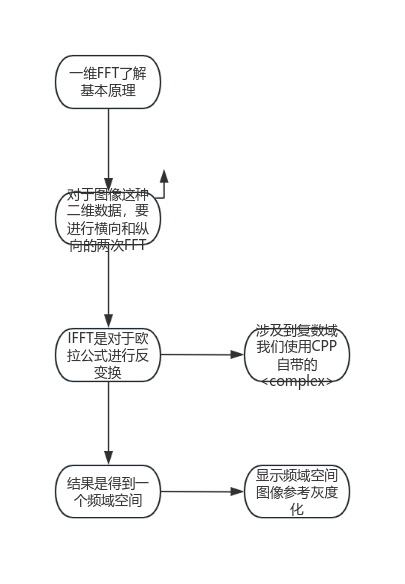
1. **主要仪器设备**

**仪器：**计算机

**实验环境：** Windows XP + Visual C++6.0

1. **实验原理**

画程序的流程图或N-S图。



1. **源程序**

写出程序的源程序。

实验中给出了FFT的源代码：

void IFFT(complex<double> \* FD, complex<double> \* TD, int r)

{

// 付立叶变换点数

LONG count;

// 计算付立叶变换点数

count = 1 << r;

// 分配运算所需存储器

complex<double> \* X = new complex<double>[count];

// 将频域点写入X

memcpy(X, FD, sizeof(complex<double>) \* count);

// 求共轭

for(int i = 0; i < count; i++)

X[i] = complex<double> (X[i].real(), -X[i].imag());

// 调用快速付立叶变换

FFT(X, TD, r);

// 求时域点的共轭

for(i = 0; i < count; i++)

TD[i] = complex<double> (TD[i].real() \* count, -TD[i].imag() \* count);

// 释放内存

delete X;

}

void IFFourier()

{

//图像的宽度和高度

int width = lpBitsInfo->bmiHeader.biWidth;

int height = lpBitsInfo->bmiHeader.biHeight;

int LineBytes = (width \* lpBitsInfo->bmiHeader.biBitCount + 31)/32 \* 4;

// FFT宽度（必须为2的整数次方）

int FFT\_w = 1;

// FFT宽度的幂数，即迭代次数

int wp = 0;

while(FFT\_w \* 2 <= width)

{

FFT\_w \*= 2;

wp ++;

}

// FFT高度（必须为2的整数次方）

int FFT\_h = 1;

// FFT高度的幂数，即迭代次数

int hp = 0;

while(FFT\_h \* 2 <= height)

{

FFT\_h \*= 2;

hp ++;

}

// 分配内存

complex<double>\* TD = new complex<double>[FFT\_w \* FFT\_h];

complex<double>\* FD = new complex<double>[FFT\_w \* FFT\_h];

int i, j;

for(i = 0; i < FFT\_h; i++) // 行

for(j = 0; j < FFT\_w; j++) // 列

FD[j + FFT\_w \* i] = gFD[i + FFT\_h\*j];

// 沿水平方向进行快速付立叶变换

for(i = 0; i < FFT\_h; i++)

IFFT(&FD[FFT\_w \* i], &TD[FFT\_w \* i], wp);

// 保存中间变换结果

for(i = 0; i < FFT\_h; i++)

for(j = 0; j < FFT\_w; j++)

FD[i + FFT\_h \* j] = TD[j + FFT\_w \* i];

// 沿垂直方向进行快速付立叶变换

for(i = 0; i < FFT\_w; i++)

IFFT(&FD[i \* FFT\_h], &TD[i \* FFT\_h], hp);

//为反变换图像分配内存

LONG size = 40 + 1024 + LineBytes \* height;

lpDIB\_IFT = (LPBITMAPINFO) malloc(size);

if (NULL == lpDIB\_IFT)

return;

memcpy(lpDIB\_IFT, lpBitsInfo, size);

//指向反变换图像数据指针

BYTE\* lpBits = (BYTE\*)&lpDIB\_IFT->bmiColors[256];

BYTE\* pixel;

double temp;

for(i = 0; i < FFT\_h; i++) // 行

{

for(j = 0; j < FFT\_w; j++) // 列

{

pixel = lpBits + LineBytes \* (height - 1 - i) + j;

temp= (TD[j\*FFT\_h + i].real() / pow(-1, i+j));

if (temp < 0)

temp = 0;

else if (temp >255)

temp = 255;

\*pixel = (BYTE)temp;

}

}

// 删除临时变量

delete FD;

delete TD;

delete gFD;

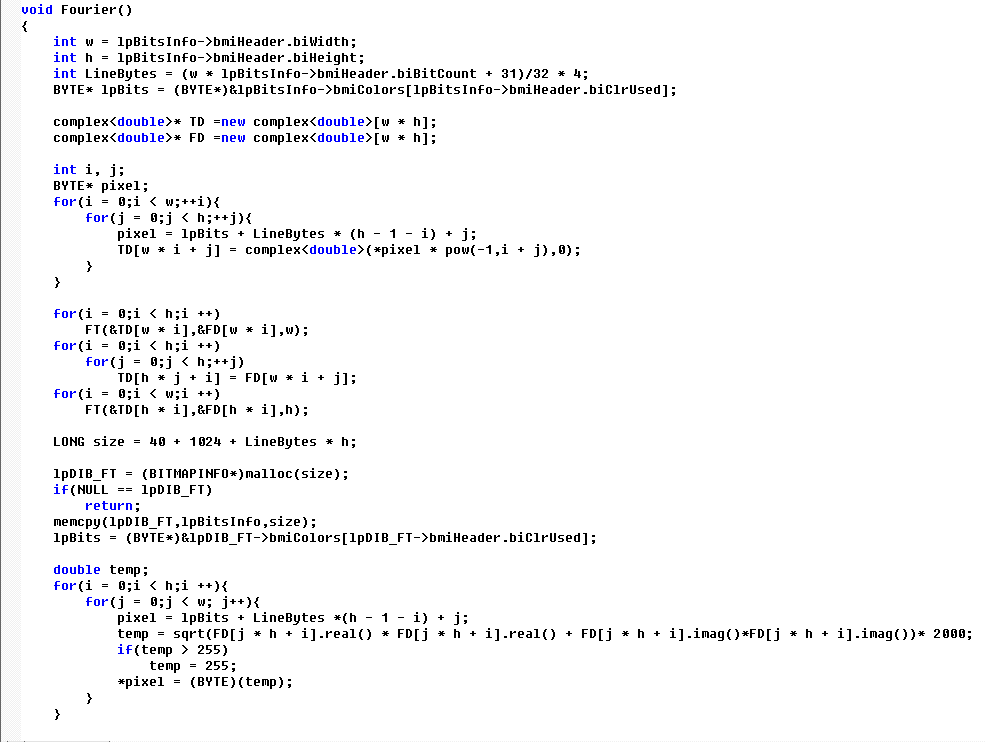
gFD = NULL;

}

需要注意的是，我们做反变换和变换的时候由于我们要直接操作图像本身，所以我们最好有一个temp变量去存储变换时会用到的中间结果，这点和两值交换的原理类似。

1. **实验步骤与调试**

主要是写出对源程序的语法错误以及逻辑错误的进行发现、修正以及调试的步骤。



在实验中，由于源代码对图像的傅里叶变换对每一个像素点进行了三次操作，时间复杂度为O（n3），所以运算需要几十秒钟，我们需要耐心等待。

1. **实验结果与分析**

写出最终逻辑正确的程序在各种实验数据下的实验结果以及进行简单的结果分析。

我们在

**暨南大学本科实验报告专用纸(附页)**

傅里叶变换得到频域空间效果：

