**南昌航空大学**

**21学年—22学年第 2 学期 医疗软件技术基础 实验四**

专业名称： 生物医学工程 实验学时： 2

学号： 19084131 姓名： 张逸辰

实验题目：DICOM文件读取

实验环境： vs2019

实验目的：

1．掌握DICOM文件的格式；

2．掌握使用C读取DICOM文件中的信息和图像。

3. 掌握OPENCV库的使用

实验内容：

（1）通过open函数读取DICOM文件；

（2）顺序读取各种Tag；

（3）通过TAG，读取相应信息；

（4）通过TAG读取图像信息，并进行调窗和显示。

实验要求：

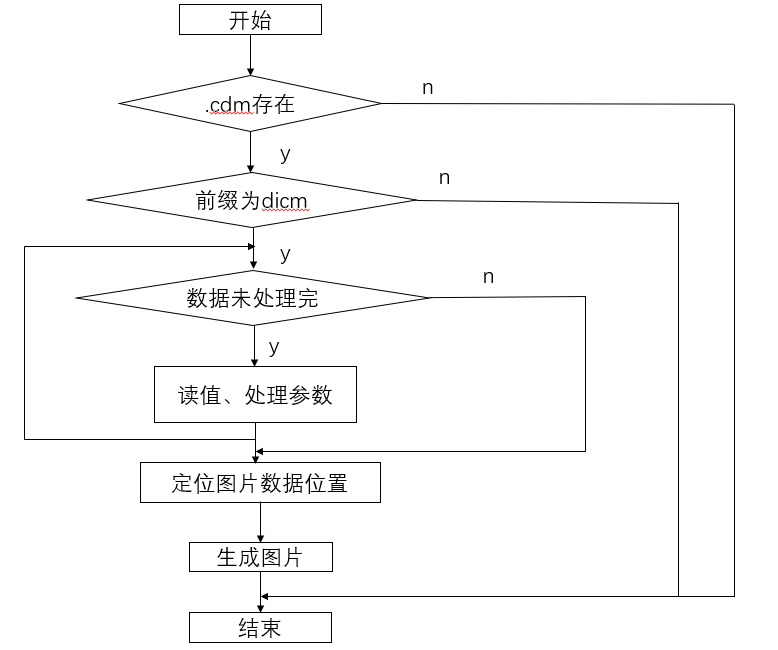
(1) 详细报告OPENCV的配置。

(2) 程序要添加适当的注释，程序的书写要采用缩进格式。

(3) 对DICOM文件中的重要信息进行输出。

(4) 根据实验报告模板详细书写实验报告,在实验报告中给出算法的流程图。

实验流程图：



实验程序及注释：

// 11111.cpp : 定义控制台应用程序的入口点。

//

//#include <opencv2/opencv.hpp>

#include "highgui.h"

/\*#pragma comment(lib, "opencv\_core2411d.lib")

#pragma comment(lib, "opencv\_imgproc2411d.lib")

#pragma comment(lib, "opencv\_highgui2411d.lib")\*/

using namespace cv;

using namespace std;

struct TagValue

{

unsigned short tag1;

unsigned short tag2;

};

// 原始象素类型

enum E\_SourcePixelType

{

ESourcePixelType\_U16, // USHORT

ESourcePixelType\_I16, // SHORT

};

int main()

{

bool isVR=true;

bool isLitteEndian=true;

int file\_length=0;

char VR[3];

unsigned int pixDataLen=0;

unsigned int pixDataOffset=0;

unsigned short channle=0;

unsigned short rows=0;

unsigned short cols=0;

unsigned short dataLen=0;

unsigned short validLen=0;

E\_SourcePixelType pixelType;

int windowsWidth=0;

int windowsCenter=0;

bool ZeroIsBlack=true;

float RescaleSlope =0.06;

float RescaleIntercept=0;

FILE \*fp;

fp=fopen("11.dcm","rb");

if(fp==NULL)

{

printf("can not open file!");

return 0;

}

fseek(fp,0,SEEK\_END);

file\_length=ftell(fp);

fseek(fp,0,SEEK\_SET);

fseek(fp,128,SEEK\_SET);

char headchar[5];

memset(headchar,0,5);

int read\_num = fread(headchar,1,4,fp);

if(read\_num!=4)

{

fclose(fp);

return 0;

}

if(strcmp(headchar,"DICM"))

{

fclose(fp);

printf("File is not DICM");

return 0;

}

while(ftell(fp)+6<file\_length)

{

TagValue tag;

unsigned int len;

memset(VR,0,3);

fread(&tag,sizeof(TagValue),1,fp);

int index=ftell(fp);

if(tag.tag1==0x02)

{

fread(VR,1,2,fp);

if(!strcmp(VR,"OB")||!strcmp(VR,"OW")||!strcmp(VR,"SQ"))

{

fseek(fp,2,SEEK\_CUR);

fread(&len,sizeof(unsigned int),1,fp);

}

else

{

unsigned short l;

int ss = fread(&l,sizeof(unsigned short),1,fp);

int a=ftell(fp);

len =(unsigned int)l ;

}

}

else if(tag.tag1==0xfffe)

{

if(tag.tag2==0xe000||tag.tag2==0xe00d||tag.tag2==0xe0dd)

{

fread(&len,sizeof(unsigned int),1,fp);

}

}

else if(isVR==true)

{

fread(VR,1,2,fp);

int a= ftell(fp);

if(!strcmp(VR,"OB")||!strcmp(VR,"OW")||!strcmp(VR,"SQ"))

{

fseek(fp,2,SEEK\_CUR);

fread(&len,sizeof(unsigned int),1,fp);

}

else

{

unsigned short l;

l=sizeof(unsigned short);

fread(&l,sizeof(unsigned short),1,fp);

a= ftell(fp);

len =(unsigned int)l ;

}

}

else if(isVR==false)

{

fread(&len,sizeof(unsigned int),1,fp);

}

if(tag.tag1==0x02&&tag.tag2==0x10)

{

char msg[124];

memset(msg,0,124);

fread(msg,1,len,fp);

if(!strcmp(msg,"1.2.840.10008.1.2.1"))

{

isLitteEndian=true;

isVR=true;

}

else if(!strcmp(msg,"1.2.840.10008.1.2.2"))

{

isLitteEndian=false;

isVR=true;

}

else if(!strcmp(msg,"1.2.840.10008.1.2"))

{

isLitteEndian=true;

isVR=false;

}

}

else if(tag.tag1 ==0x28 && tag.tag2==0x103)

{

unsigned short m;

fread(&m,sizeof(unsigned short),1,fp);

if(m==0)

{

pixelType = ESourcePixelType\_U16;

}

else if(m ==1)

{

pixelType = ESourcePixelType\_I16;

}

}

else if(tag.tag1==0x7fe0&&tag.tag2==0x10)

{

pixDataLen=len;

pixDataOffset=ftell(fp);

fseek(fp,len,SEEK\_CUR);

}

else if(tag.tag1==0x28&&tag.tag2==0x10)

{

fread(&rows,sizeof(unsigned short),1,fp);

}

else if(tag.tag1==0x28&&tag.tag2==0x11)

{

fread(&cols,sizeof(unsigned short),1,fp);

}

else if(tag.tag1==0x28&&tag.tag2==0x02)

{

fread(&channle,sizeof(unsigned short),1,fp);

}

else if(tag.tag1==0x28&&tag.tag2==0x101)

{

fread(&validLen,sizeof(unsigned short),1,fp);

}

else if(tag.tag1==0x28&&tag.tag2==0x100)

{

fread(&dataLen,sizeof(unsigned short),1,fp);

}

else if(tag.tag1==0x28&&tag.tag2==0x1050)

{

char msg[11];

memset(msg,0,11);

fread(msg,1,len,fp);

windowsCenter=atoi(msg);

}

else if(tag.tag1==0x28,tag.tag2==0x1051)

{

//fseek(fp,len,SEEK\_CUR);

char msg[40];

memset(msg,0,40);

fread(msg,1,len,fp);

windowsWidth=atoi(msg);

}

else if(tag.tag1==0x0028&&tag.tag2==0x0004)

{

char msg[40];

memset(msg,0,40);

fread(msg,1,len,fp);

if(!strcmp(msg,"MONOCHROME1 "))

{

ZeroIsBlack=false;

}

else if(!strcmp(msg,"MONOCHROME2 "))

{

ZeroIsBlack=true;

}

}

else if(tag.tag1==0x0028&&tag.tag2==0x1052)

{

char msg[40];

memset(msg,0,40);

fread(msg,1,len,fp);

RescaleIntercept=atof(msg);

}

else if(tag.tag1==0x0028&&tag.tag2==0x1053)

{

char msg[40];

memset(msg,0,40);

fread(msg,1,len,fp);

RescaleSlope =atof(msg);

}

else

{

char msg[1024];

memset(msg,0,1024);

fread(msg,1,len,fp);

}

}

fseek(fp,pixDataOffset,SEEK\_SET);

if(windowsCenter==0&&windowsWidth==0)

{

windowsWidth = 1 << validLen;

windowsCenter = windowsWidth / 2;

}

//int min\_value,max\_value;

//min\_value=windowsCenter-windowsWidth/2.0+0.5;

//max\_value=windowsCenter+windowsWidth/2.0+0.5;

//double pers = 255.0/(max\_value-min\_value);

Mat src;

Mat src2;

int nPixel= 0;

double fCtA = 0;

double fCtB = 0;

fCtA = (double)256 /windowsWidth;

fCtB = 128 - 256 \* (double)windowsCenter / windowsWidth;

if (fCtB < 0)

{

fCtB = 0;

}

if (fCtB > 255)

{

fCtB = 255;

}

if(channle==1)

{

src.create((int)rows,(int)cols,CV\_8UC1);

src2.create((int)rows,(int)cols, CV\_16SC1);

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

{

for(int j=0;j<cols;j++)

{

unsigned short gray=0;

short gray2 = 0;

unsigned char pix[2];

fread(pix,1,2,fp);

if(pixelType == ESourcePixelType\_U16)

{

if(validLen<=8)

{

if(isLitteEndian)

{

gray=pix[0];

}

else

{

gray=pix[1];

}

}

else

{

long temp = 0;

if(isLitteEndian)

{

gray=\*(unsigned short\*)pix;

if(gray > 32767)

{

gray = 32767;

}

temp =gray\*RescaleSlope+RescaleIntercept;

temp = temp \* fCtA + fCtB;

}

else

{

gray=pix[1]+pix[0]\*256;

temp =gray\*RescaleSlope+RescaleIntercept;

temp = temp \* fCtA + fCtB;

}

int nValue = (int)temp;

if(nValue>0xff)

{

nValue=0xff;

}

else if(nValue<0)

{

nValue=0;

}

nPixel = nValue;

}

}

else if(pixelType ==ESourcePixelType\_I16)

{

if(validLen<=8)

{

if(isLitteEndian)

{

gray2=pix[0];

}

else

{

gray2=pix[1];

}

}

else

{

long temp = 0;

if(isLitteEndian)

{

gray2=\*(short\*)pix;

if(gray2 > 32767)

{

gray2 = 32767;

}

if(gray2 < -32767)

{

gray2 = -32767;

}

temp =gray2\*RescaleSlope+RescaleIntercept;

temp = temp \* fCtA + fCtB;

}

else

{

gray2=pix[1]+pix[0]\*256;

temp=gray2\*RescaleSlope+RescaleIntercept;

temp = temp \* fCtA + fCtB;

}

int nValue = (int)temp;

if(nValue>0xff)

{

nValue=0xff;

}

else if(nValue<0)

{

nValue=0;

}

nPixel = nValue;

}

}

if(!ZeroIsBlack)

{

nPixel=255-nPixel;

}

src.at<uchar>(i,j)=nPixel;

}

//std::cout<<std::endl;

}

}

else if(channle==3)

{

src.create((int)rows,(int)cols,CV\_8UC3);

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

{

for(int j=0;j<cols;j++)

{

unsigned char pix[3];

fread(pix,1,3,fp);

src.at<Vec3b>(i,j)[0]=pix[2];

src.at<Vec3b>(i,j)[1]=pix[1];

src.at<Vec3b>(i,j)[2]=pix[0];

}

}

}

fclose(fp);

cvNamedWindow("Dicomimage",0);

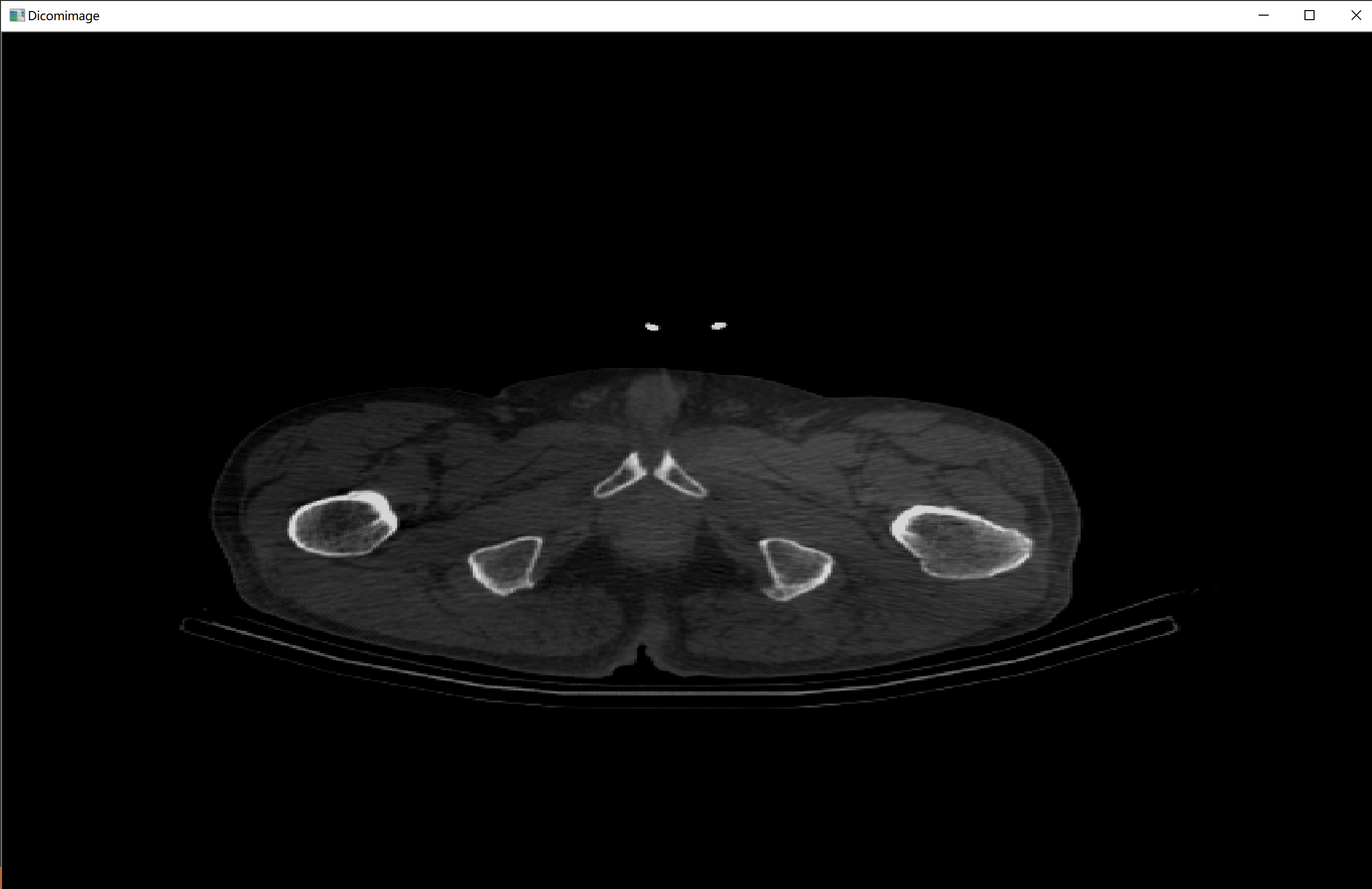
imshow("Dicomimage",src);

cvWaitKey(0);

return 0;

}

实验结果：



实验小结：