地球科学与环境工程学院

**《数字图像处理》**

实 验 报 告

**实验名称： 植被指数**

**指导老师： 叶沅鑫**

**班 级： 测绘一班**

**姓 名： 刘义豪**

**学 号： 2020114164**

**2022年 10月 15 日**

目录

[一、实验目的与任务 2](#_Toc117027740)

[二、实验原理 3](#_Toc117027741)

[1、NDVI计算 3](#_Toc117027742)

[2、植被提取 3](#_Toc117027743)

[三、编程思路 3](#_Toc117027744)

[四、运行结果图 4](#_Toc117027745)

[五、实验总结 6](#_Toc117027746)

[附录 6](#_Toc117027747)

## 一、实验目的与任务

利用Vc++编写植被指数的程序

采用参数传递方式，并用一个变量输出运算结果；

要有说明部分，说明输入参数的格式，说明输出数据方式

将计算植被指数的结果用文件保存下来，命名为vegetable.tif

植被指数计算公式

## 二、实验原理

### **1、NDVI计算**

植被指数计算公式（NDVI）：

​

### 2、植被提取

近红外（NIR）的波段为0.78到3微米，红光（R）的波段为0.625到0.74微米。在植物的反射波谱特性曲线中，近红外和红光波段中间有较大的落差，这是植物在反射波谱特性曲线中特有的性质，利用这个性质可以在遥感图像中提取植被。

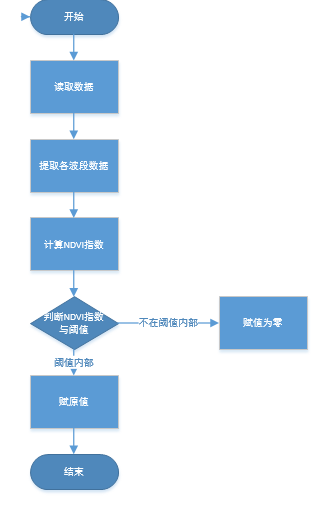
通过上一步计算出来的NDVI指数进行下一步操作，即NDVI越大，近红外和红外波段之间的落差越大，则越有可能是植被。

## 三、编程思路

本次实验其实就是对应了遥感原理课程中的特征提取。首先获取影像的基本数据，编写CalcuNDVI用于计算NDVI（植被指数）的值。其中分别读取R和NIR波段的数据，逐像元计算NDVI值。

计算完成NDVI指数之后，编写ExtractVegetation方法用于提取植被，通过判断NDVI值是否在规定的阈值范围中，来判定是否为植被区域，即把在阈值范围内的赋予255，之外的赋值为零，以实现二值化。

以下为编程思路的流程图。



## 四、运行结果图

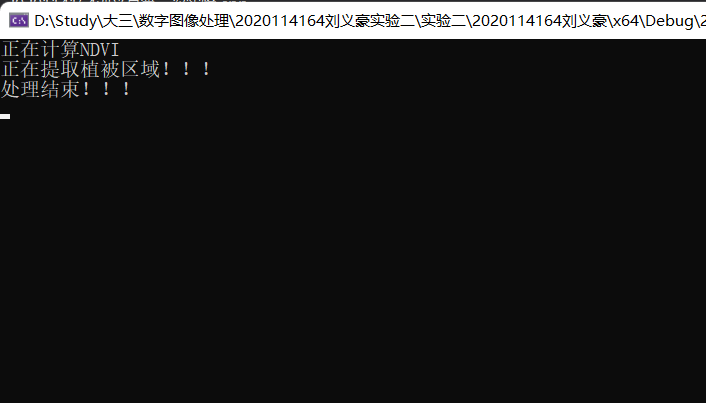


图1.1 程序运行成功截图

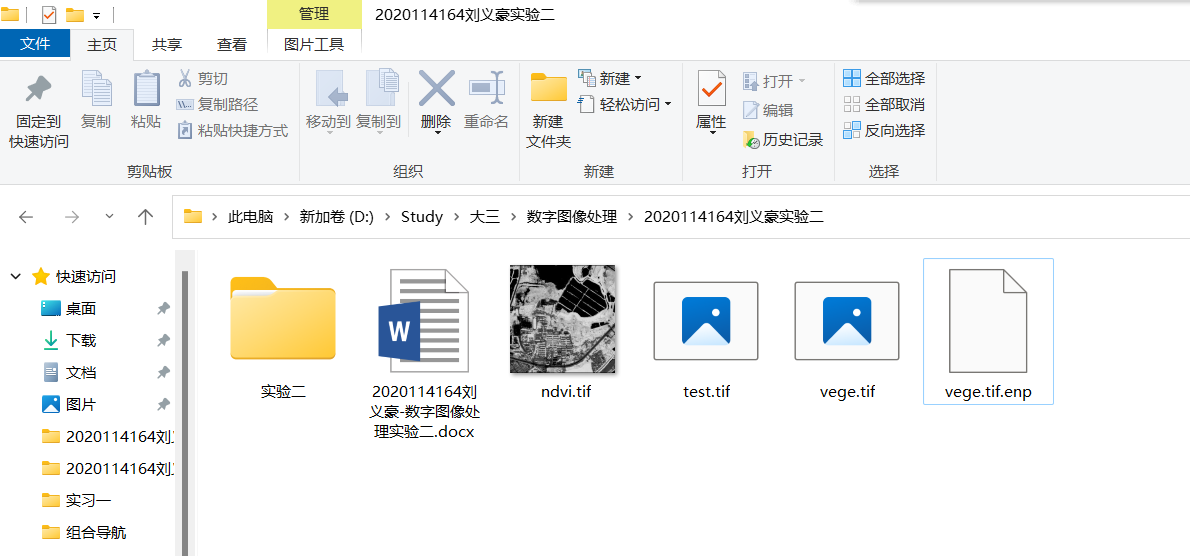


图1.2 从左至右依次为NDVI值计算图、原图、植被提取结果图



图1.3 提取NDVI指数

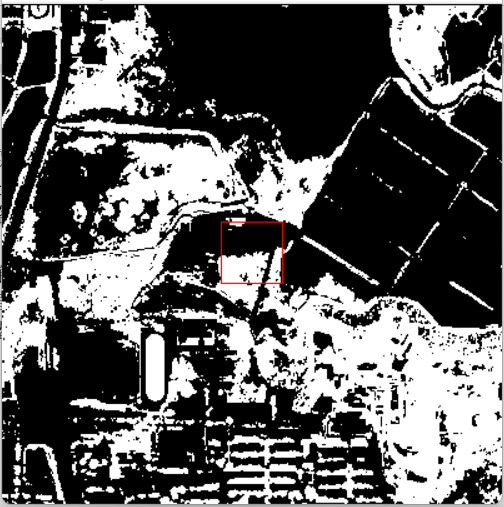


图1.4 提取植被完成后图像

## 五、实验总结

本次实验有了上学期特征提取的基础之后在原理上理解不难，外加上次实验对GDAL库的初步认识之后，很快便可以将代码初步写出来。但是需要注意的是本次实验图像与上次实验的数据格式不同，不可以用char来读取，使用float类型或short类型读取更为适合。

二值化处理在编程实现之中第一次遇见。但通过上网查阅得知二值化就是单一波段对阈值内部数据赋予最高值，阈值外的数据赋予最低值，实现二值化处理。

## 附录

#include<iostream>

#include "gdal.h"

#include "gdal\_priv.h"

#include "Func.h"

using namespace std;

int main()

{

GDALAllRegister();

CPLSetConfigOption("GDAL\_FILENAME\_IS\_UTF8", "NO");

// 读取影像

char openpath[] = "D:\\Study\\大三\\数字图像处理\\2020114164刘义豪实验二\\test.tif";

GDALDataset\* mDataset;

// 只读方式打开

mDataset = (GDALDataset\*)GDALOpen(openpath, GA\_ReadOnly);

if (mDataset == NULL)

{

cout << "未能正确加载数据!!!" << endl;

system("pause");

GDALDestroyDriverManager();

}

// 获取影像数据

int mX = mDataset->GetRasterXSize();

int mY = mDataset->GetRasterYSize();

GDALDataType mImgDataType = mDataset->GetRasterBand(1)->GetRasterDataType();

// 获取影像的地理信息和投影信息

double geoInformation[6];

mDataset->GetGeoTransform(geoInformation);

const char\* gdalProjection = mDataset->GetProjectionRef();

// 新建一个驱动，影像格式为GTiff

GDALDriver\* hDriver = GetGDALDriverManager()->GetDriverByName("GTiff");

// 调用CalcuNDVI 进行计算，需要先建立一个数据集保存NDVI结果,NDVI中的数据类型是float

char savepath[] = "D:\\Study\\大三\\数字图像处理\\2020114164刘义豪实验二\\ndvi.tif";

GDALDataset\* mNDVIset = hDriver->Create(savepath, mX, mY, 1, GDT\_Float32, NULL);

cout << "正在计算NDVI" << endl;

CalcuNDVI(mDataset, mNDVIset);

// 将影像的地理信息写到NDVI数据集中

mNDVIset->SetGeoTransform(geoInformation);

mNDVIset->SetProjection(gdalProjection);

// 调用ExtractVegetation进行提取，需要先建立一个数据集保存提取结果

char savepath2[] = "D:\\Study\\大三\\数字图像处理\\2020114164刘义豪实验二\\vege.tif";

GDALDataset\* mVegetationSet = hDriver->Create(savepath2, mX, mY, 4, mImgDataType, NULL);

cout << "正在提取植被区域！！！" << endl;

ExtractVegetation(mDataset, mNDVIset, mVegetationSet, 0.35, 0.7);

// 将影像的地理信息写到植被数据集中

mVegetationSet->SetGeoTransform(geoInformation);

mVegetationSet->SetProjection(gdalProjection);

// 关闭数据集，释放内存

GDALClose(mDataset);

GDALClose(mNDVIset);

GDALClose(mVegetationSet);

GDALDestroyDriverManager();

cout << "处理结束！！！" << endl;

getchar();

return 0;

}

#include "Func.h"

#include<iostream>

#include "gdal.h"

#include "gdal\_priv.h"

void CalcuNDVI(GDALDataset\* mDataset, GDALDataset\*& mNDVIset)

{

// 获取影像的参数

int mX = mDataset->GetRasterXSize();

int mY = mDataset->GetRasterYSize();

int mDataType = mDataset->GetRasterBand(1)->GetRasterDataType();

// 将影像中的四个波段取出,分别为NIR，R,G,B

GDALRasterBand\* mB = mDataset->GetRasterBand(1);

GDALRasterBand\* mG = mDataset->GetRasterBand(2);

GDALRasterBand\* mR = mDataset->GetRasterBand(3);

GDALRasterBand\* mNIR = mDataset->GetRasterBand(4);

// 申请缓冲区

float\* RedBuf = (float\*)CPLMalloc(sizeof(float) \* mX \* mY);

float\* NIRBuf = (float\*)CPLMalloc(sizeof(float) \* mX \* mY);

float\* mNDVIband = (float\*)CPLMalloc(sizeof(float) \* mX \* mY);

// 从R和NIR波段中取出数据放到缓冲区

mR->RasterIO(GF\_Read, 0, 0, mX, mY, RedBuf, mX, mY, GDT\_UInt32, 0, 0);

mNIR->RasterIO(GF\_Read, 0, 0, mX, mY, NIRBuf, mX, mY, GDT\_UInt32, 0, 0);

// 计算NDVI

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

{

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

{

float r = RedBuf[i \* mX + j];

float nir = NIRBuf[i \* mX + j];

mNDVIband[i \* mX + j] = (nir - r) / (nir + r);

}

}

// 将计算结果写入NDVI的数据集中

mNDVIset->GetRasterBand(1)->RasterIO(GF\_Write, 0, 0, mX, mY, mNDVIband, mX, mY, GDT\_Float32, 0, 0);

// 释放内存

CPLFree(RedBuf);

CPLFree(NIRBuf);

CPLFree(mNDVIband);

}

void ExtractVegetation(GDALDataset\* mDataset, GDALDataset\* mNDVIset, GDALDataset\*& VegetationSet, float th1, float th2)

{

// 获取影像参数

int mX = mDataset->GetRasterXSize();

int mY = mDataset->GetRasterYSize();

GDALDataType mImgDataType = mDataset->GetRasterBand(1)->GetRasterDataType();

GDALDataType mNDVIDataType = mNDVIset->GetRasterBand(1)->GetRasterDataType();

// 获取原始影像的四个波段

GDALRasterBand\* mBand1 = mDataset->GetRasterBand(1);

GDALRasterBand\* mBand2 = mDataset->GetRasterBand(2);

GDALRasterBand\* mBand3 = mDataset->GetRasterBand(3);

GDALRasterBand\* mBand4 = mDataset->GetRasterBand(4);

// 获取NDVI的波段

GDALRasterBand\* mNDVIBand = mNDVIset->GetRasterBand(1);

// 申请缓冲区，用来保存原始波段和NDVI的数据

unsigned short\* mBandBuf1 = (unsigned short\*)CPLMalloc(mX \* mY \* mImgDataType);

unsigned short\* mBandBuf2 = (unsigned short\*)CPLMalloc(mX \* mY \* mImgDataType);

unsigned short\* mBandBuf3 = (unsigned short\*)CPLMalloc(mX \* mY \* mImgDataType);

unsigned short\* mBandBuf4 = (unsigned short\*)CPLMalloc(mX \* mY \* mImgDataType);

float\* mNDVIBandBuf = (float\*)CPLMalloc(mX \* mY \* mNDVIDataType);

// 将各个波段的数据放入对应的缓冲区

mBand1->RasterIO(GF\_Read, 0, 0, mX, mY, mBandBuf1, mX, mY, mImgDataType, 0, 0);

mBand2->RasterIO(GF\_Read, 0, 0, mX, mY, mBandBuf2, mX, mY, mImgDataType, 0, 0);

mBand3->RasterIO(GF\_Read, 0, 0, mX, mY, mBandBuf3, mX, mY, mImgDataType, 0, 0);

mBand4->RasterIO(GF\_Read, 0, 0, mX, mY, mBandBuf4, mX, mY, mImgDataType, 0, 0);

mNDVIBand->RasterIO(GF\_Read, 0, 0, mX, mY, mNDVIBandBuf, mX, mY, mNDVIDataType, 0, 0);

// 申请缓冲区，用来保存提取出的植被区域

unsigned short\* mVegBand1 = (unsigned short\*)CPLMalloc(mX \* mY \* mImgDataType);

unsigned short\* mVegBand2 = (unsigned short\*)CPLMalloc(mX \* mY \* mImgDataType);

unsigned short\* mVegBand3 = (unsigned short\*)CPLMalloc(mX \* mY \* mImgDataType);

unsigned short\* mVegBand4 = (unsigned short\*)CPLMalloc(mX \* mY \* mImgDataType);

// 通过阈值来提取植被区域

for (int i = 0; i < mX \* mY; i++)

{

// 若在阈值范围内，赋对应的值，若不在，赋值为0

if (mNDVIBandBuf[i] >= th1 && mNDVIBandBuf[i] <= th2)

{

mVegBand1[i] = mBandBuf1[i];

mVegBand2[i] = mBandBuf2[i];

mVegBand3[i] = mBandBuf3[i];

mVegBand4[i] = mBandBuf4[i];

}

else

{

mVegBand1[i] = 0;

mVegBand2[i] = 0;

mVegBand3[i] = 0;

mVegBand4[i] = 0;

}

}

// 把提取结果放到数据集中

VegetationSet->GetRasterBand(1)->RasterIO(GF\_Write, 0, 0, mX, mY, mVegBand1, mX, mY, mImgDataType, 0, 0);

VegetationSet->GetRasterBand(2)->RasterIO(GF\_Write, 0, 0, mX, mY, mVegBand2, mX, mY, mImgDataType, 0, 0);

VegetationSet->GetRasterBand(3)->RasterIO(GF\_Write, 0, 0, mX, mY, mVegBand3, mX, mY, mImgDataType, 0, 0);

VegetationSet->GetRasterBand(4)->RasterIO(GF\_Write, 0, 0, mX, mY, mVegBand4, mX, mY, mImgDataType, 0, 0);

// 释放内存

CPLFree(mBandBuf1);

CPLFree(mBandBuf2);

CPLFree(mBandBuf3);

CPLFree(mBandBuf4);

CPLFree(mNDVIBandBuf);

CPLFree(mVegBand1);

CPLFree(mVegBand2);

CPLFree(mVegBand3);

CPLFree(mVegBand4);

}