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： 图像匹配**

**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 21 日**

目录

[一、实验目的与任务 3](#_Toc117286101)

[二、实验原理 3](#_Toc117286102)

[三、编程思路 4](#_Toc117286103)

[四、运行结果图 5](#_Toc117286104)

[五、实验总结 6](#_Toc117286105)

[附录 6](#_Toc117286106)

## 一、实验目的与任务

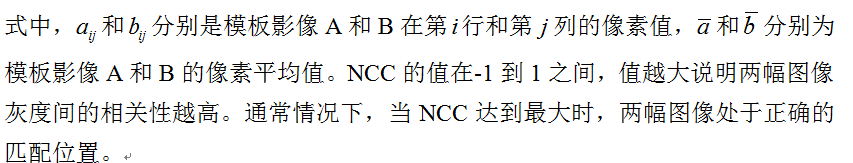
掌握基于归一化相关系数（[normalized](app:ds:normalized) [correlation](app:ds:correlation) [coefficient](app:ds:coefficient)，NCC）的图像匹配算法；

利用Vc++编写一个M文件，要求：

基于NCC的图像匹配函数，输出最终的匹配位置（行r，列c），函数名定义为match\_ncc.cpp；

NCC的计算公式：

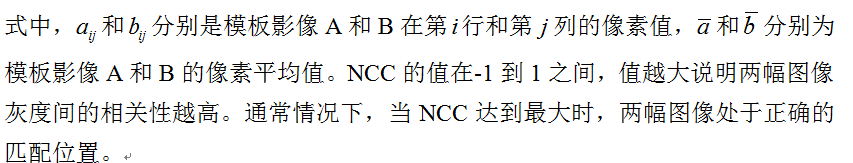




## 二、实验原理

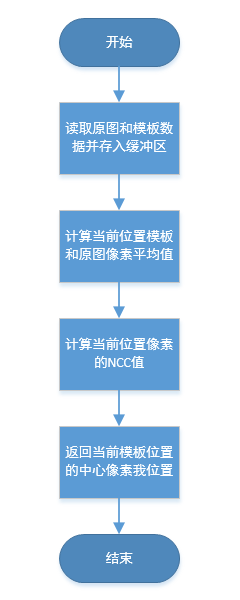
NCC的计算公式：





## 三、编程思路

读取模板图和原图数据后，创建缓冲区存储两张图片的像素数据。For循环遍历原图每一个元素，在每一个当前位置使用for循环计算模板和原图的像素平均值。利用上式计算出来的平均值，通过NCC公式，计算出当前位置的NCC值，并判断是否为最大值，得到最大位置NCC后，得到当前模板中心位置，返回给主函数输出。



## 四、运行结果图

D:\Study\大三\数字图像处理\2020114164刘义豪实验三\img.tif

图1.1 原图

D:\Study\大三\数字图像处理\2020114164刘义豪实验三\template.tif

图1.2 模板图

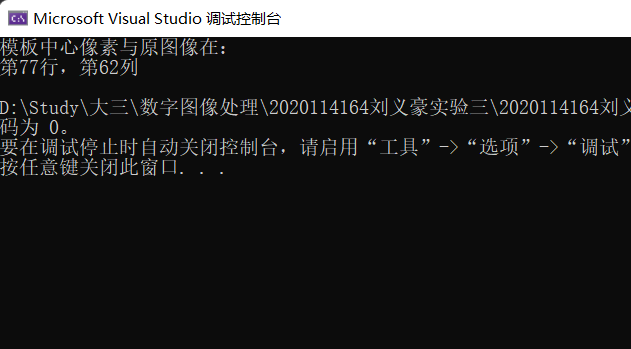


图1.3 程序运行成功截图

## 五、实验总结

本次实验运用到了上学期遥感原理课程学到的图像配准方法，通过计算每个像素点的NCC值从而得到模板在原图滑动过程中最大的NCC值，从而得到模板中心位置。

本次实验主要难点存在对NCC值的计算上，两次累和以及在循环内部还需进行循环计算相应位置像素平均值以及最后的NCC值。通过对公式的理解之后，慢慢找到了合理运用循环条件控制程序运行，从而同时计算出相应位置像素平均值以及对应的NCC值。

## 附录

***2020114164刘义豪.cpp***

#include <iostream>

#include<iostream>

#include "gdal.h"

#include "gdal\_priv.h"

#include "NCC.h"

int main()

{

GDALAllRegister();

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

char imgPath[] = "D:\\Study\\大三\\数字图像处理\\2020114164刘义豪实验三\\img.tif";

char tmpPath[] = "D:\\Study\\大三\\数字图像处理\\2020114164刘义豪实验三\\template.tif";

GDALDataset\* imgData = (GDALDataset\*)GDALOpen(imgPath, GA\_ReadOnly);

GDALDataset\* tmpData = (GDALDataset\*)GDALOpen(tmpPath, GA\_ReadOnly);

if (imgData == NULL || tmpData == NULL)

{

std::cout << "未能正确加载数据!!!" <<std:: endl;

system("pause");

GDALDestroyDriverManager();

}

int x = 0;

int y = 0;

bool flag;

flag = CalNCC(imgData, tmpData, x, y);

if (flag==false)

{

return 0;

}

std::cout << "模板中心像素与原图像在：" << std::endl;

std::cout << "第" << x << "行，第" << y << "列" << std::endl;

GDALClose(imgData);

GDALClose(tmpData);

GDALDestroyDriverManager();

return 0;

}

***NCC.h***

#pragma once

#include <iostream>

#include<iostream>

#include "gdal.h"

#include "gdal\_priv.h"

bool CalNCC(GDALDataset\* imgData, GDALDataset\* tmpData, int& x, int& y);

***NCC.cpp***

#include "NCC.h"

#include "stdio.h"

#include <cmath>

bool CalNCC(GDALDataset\* imgData, GDALDataset\* tmpData, int& x, int& y) {

int imgX = imgData->GetRasterXSize();

int imgY = imgData->GetRasterYSize();

int imgBandNum = imgData->GetRasterCount();

GDALDataType imgType = imgData->GetRasterBand(1)->GetRasterDataType();

int tmpX = tmpData->GetRasterXSize();

int tmpY = tmpData->GetRasterYSize();

int tmpBandNum = tmpData->GetRasterCount();

GDALDataType tmpType = tmpData->GetRasterBand(1)->GetRasterDataType();

if (tmpX > imgX || tmpY > imgY || tmpX % 2 == 0 || tmpY % 2 == 0) {

std::cout << "模板数据有误！" << std::endl;

system("pause");

return false;

}

GDALRasterBand\* imgBand = imgData->GetRasterBand(1);

GDALRasterBand\* tmpBand = tmpData->GetRasterBand(1);

unsigned char\* imgBuff = (unsigned char\*)CPLMalloc((double)imgX \* imgY \* imgType);

unsigned char\* tmpBuff = (unsigned char\*)CPLMalloc((double)tmpX \* tmpY \* tmpType);

imgBand->RasterIO(GF\_Read, 0, 0, imgX, imgY, imgBuff, imgX, imgY, imgType, 0, 0);

tmpBand->RasterIO(GF\_Read, 0, 0, tmpX, tmpY, tmpBuff, tmpX, tmpY, tmpType, 0, 0);

int maxX = 0;

int maxY = 0;

float maxNcc = 0;

int mTmpX = tmpX / 2;

int mTmpY = tmpY / 2;

//遍历原图所有点位

for (int i = 0; i < imgY - tmpY; i++) {

for (int j = 0; j < imgX - tmpX; j++) {

//每次到一个新点位，将像素和与平均值归零

double sumTmp = 0, sumImg = 0, aveTmp = 0, aveImg = 0;

//计算当前位置原图和模板像素平均值

for (int t = 0; t < tmpY; t++) {

for (int k = 0; k < tmpX; k++) {

//模板当前像素值

sumTmp += (double)tmpBuff[t \* tmpX + k];

//原图当前位置像素值

sumImg += (double)imgBuff[(t + i) \* imgX + (j + k)];

}

}

//计算当前位置原图和模版的像素平均值

aveImg = sumImg / ((double)imgX \* imgY);

aveTmp = sumTmp / ((double)tmpX \* tmpY);

//利用上式计算出当前位置的平均像素值，计算公式

double sumImg2 = 0, sumTmp2 = 0, Mul = 0;

//遍历当前位置所有位于模版位置内的模板元素和原图元素到达累和的效果

for (int m = 0; m < imgY - tmpY; m++) {

for (int n = 0; n < imgX - tmpX; n++) {

//原图的i行j列元素减掉平均值，累和

sumImg2 += pow(((double)imgBuff[(i + m) \* imgX + (j + n)] - aveImg), 2);

//模板的i行j列元素减掉平均值，累和

sumTmp2 += pow(((double)tmpBuff[m \* tmpX + n] - aveTmp), 2);

//原图的i行j列元素减掉平均值，乘以，模板的i行j列元素减掉平均值

Mul = ((double)imgBuff[(i + m) \* imgX + (j + n)] - aveImg) \* ((double)tmpBuff[m \* tmpX + n] - aveTmp);

}

}

//每一个位置计算一个NCC值

double Ncc = Mul / sqrt(sumImg2 \* sumTmp2);

//判断是否为最大值NCC

if (Ncc>maxNcc) {

maxNcc = Ncc;

maxX = j;

maxY = i;

}

//该位置（I行J列）判断完毕，进行下一个位置

}

}

//找到中心位置

x = maxX + 1 + mTmpX;

y = maxY + 1 + mTmpY;

CPLFree(tmpBuff);

CPLFree(imgBuff);

return true;

}