《数字图像处理课程实习》

实习报告

学　院:　　遥感信息工程学院

班　级:　　2208班
2210班
2207班

实习地点:　325机房

指导教师:　李刚

组　员:　　2022302131044　祁红利

2022302131131　程欣悦

2022302131290　阮思琳

2024年　1 月　7 日

基于灰度共生矩阵的纹理特征提取和分析

1. 研究内容与目标

1.1. 研究内容：

1.灰度共生矩阵（GLCM）：掌握灰度共生矩阵的概念，得到图像中有某种空间位置关系的两个像素灰度的联合分布，位置关系一般为0度、45度、90度和135度。

2.特征提取方法：基于灰度共生矩阵的纹理特征提取过程。包括能量（二阶矩）、对比度、相关性、熵以及逆差矩的常用纹理特征的计算和提取。

3.应用场景：基于灰度共生矩阵的纹理特征提取在遥感图像分析领域的应用，主要是林地提取方面的应用，即使用不同的纹理特征对林地进行提取。

4.实验与分析：利用实际数据集进行实验，验证所提方法的可行性和有效性。比较不同特征提取方法的效果，找到对于林地提取的准确判断的特征提取和对应阈值。

1.2. 研究目标：

1.理论方面：理解和掌握基于灰度共生矩阵的纹理特征提取的理论和方法。

2.技术方面：做出有效的基于灰度共生矩阵的纹理特征提取方法，得到图像的特征值。

3.应用方面：将技术成果应用于实际问题中，解决林地提取的应用问题。

4.合作方面：通过小组作业的形式提高沟通和交流能力，提升合作能力，增加合作实践。

2. 算法原理

2.0.图像预处理：

使用个人作业中将图像从16位转换成8位并且进行灰度变换拉伸的程序对于图像进行预处理。

2.1.计算灰度共生矩阵：

灰度共生矩阵的定义：考虑二阶统计量，研究有空间关系的像素对：

$$P(i,j|d,\theta)$$

表示即在给定空间距离d和方向θ时，灰度以i为起始点（行），出现灰度级j（列）的概率（对频数进行归一化，即除以所有频数之和），它们构成了灰度共生矩阵的元素。

空间位置关系：假设一对像素的空间位置关系表示为δ=(a,b),(a,b)的取值不同，灰度共生矩阵中的值不同。对于较细的纹理，选取(1,0),(1,1),(2,0)等值。a,b取值较小对应于变化缓慢

的纹理图像，其灰度共生矩阵对角线上的数值较大。纹理的变化越快，则对角线上的数值越小，而对角线两侧的值增大。

角度关系：定义角度0度、45度、90度和135度和整数距离d=1。扫描图像的每个像素并将其存储为“参考像素”。然后将参考像素与距离d的像素进行比较。每次找到参考值和邻居值对时，GLCM的相应行和列递增1。

在图像中任意一点(x,y)及偏离它的一点(x+a,y+b)构成点对。设该点对的灰度值为(f1,f2)，假设图像的最大灰度级为L,则f1与f2的组合共有L*L种。对于整张图像，统计每一种(f1,f2)值出现的次数，然后排列成一个方阵。

通过研究图像中像素间的灰度空间相关性来描述图像的纹理特征。在灰度共生矩阵中，像素对的“元素”由两个相邻的像素组成，并存储它们的灰度值。每个像素对出现的次数及其出现的位置，表示了像素对在整个图像中的频率和相互关系。

2.2.提取纹理特征：

由于灰度图的纹理是由不同灰度值在空间位置上以某种模式反复出现而形成的，因而在图像空间中相隔某距离的两像素之间会存在一定的灰度关系，即图像中灰度值存在空间相关性。在利用这样的相关性得到灰度共生矩阵之后，便可以利用获取常用的纹理特征从而进行相关的应用。

1. 对比度（Contrast）：对比度反映了图像的清晰度和纹理的沟纹深浅。对比度越大，纹理的沟纹越深，反差越大，效果清晰；反之，对比度小，则沟纹浅，效果模糊。计算公式如下：

$$\text{Contrast} = \sum_i \sum_j p(i,j) * (i - \text{j})^2$$

2. 能量（Energy）：能量是对图像纹理的灰度变化稳定程度的度量，反应了图像灰度分布均匀程度和纹理粗细度。能量值大表明当前纹理是一种规则变化较为稳定的纹理。计算公式如下：

$$\text{ASM} = \sum_i \sum_j p(i,j)^2$$

3. 相关性（Correlation）：相关性描述了图像中灰度级的相关性程度，相关性越大说明图像在该方向上的相似性越大。计算公式如下：

$$\text{Correlation} = \sum_i \sum_j \frac{(i - Mean) * (j - Mean) * p(i,j)^2}{Variance}$$

4. 逆差矩（Homogeneity）：逆差矩是图像局部灰度均匀性的度量，如果图像局部的灰度均匀，逆差矩取值较大。计算公式如下：

$$\text{Homogeneity} = \sum_i \sum_j p(i,j) * \frac{1}{1+(i-j)^2}$$

5. 熵（Entropy）：熵是图像包含信息量的随机性度量，表现了图像的复杂程度。当灰度共生矩阵中所有值均相等或者像素值表现出最大的随机性时，熵最大；因此熵值表明了图像灰度分布的复杂程度，熵值越大，图像越复杂。计算公式如下：

$$\text{Entropy} = \sum_i \sum_j p(i,j) * \ln p(i,j)$$

通过对灰度共生矩阵进行统计分析，可以得到以上五种常用的纹理特征，这些特征可以用于图像分类、目标识别等应用领域。

2.3.实际应用：

本次实习，我们小组基于灰度共生矩阵的纹理特征提取和分析，实现了林地提取。

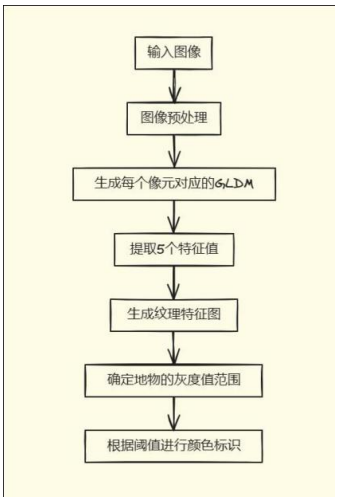
基本思路：

首先，设置n*n的窗口遍历整个图像，将每个窗口的灰度共生矩阵的特征值赋予中心像素，存储再二维容器vector<vector<double>> feature中，每个特征值可生成相应的纹理特征图，每个窗口都有四个方向、五个维度特征值的20个特征值。

之后，根据不同纹理特征图中目标地物的灰度值范围，通过不断的尝试，综合确定提取地物的方向、特征值以及特征值的阈值。

最后，根据遍历彩色图像，将符合地物阈值的像素进行颜色标识，从而提取出图像中的林地。

流程图如下：



3. 流程设计与实现

3.1. 灰度共生矩阵（详见project2.h）：

0度：

```
void getglcm_horison(Mat& input, Mat& dst)
{
    Mat src = input;
    CV_Assert(1 == src.channels());
    src.convertTo(src, CV_32S);
    int height = src.rows;
    int width = src.cols;
    int max_gray_level = 0;
```

首先定义一个函数，输入和输出都是 Mat 类型的图像。然后将输入图像复制到一个新的变量。并且使用CV_Assert确保输入图像是单通道的灰度图像。然后将图像的数据类型转换为32位有符号整数。然后获取图像的高度和宽度。并且初始化一个变量，用于存储像素的最大灰度值。

```
for (int j = 0; j < height; j++)//寻找像素灰度最大值
{
    int* srcdata = src.ptr<int>(j);
    for (int i = 0; i < width; i++)
    {
        if (srcdata[i] > max_gray_level)
        {
            max_gray_level = srcdata[i];
        }
    }
}
```

外层循环遍历图像的所有行，获取当前行所有的像素值。内层循环遍历当前行的所有像素，检查当前像素值是否大于已找到的最大灰度值。如果当前像素值更大，则更新最大灰度值。循环结束。

```
max_gray_level++;
//像素灰度最大值加1即为该矩阵所拥有的灰度级数
if (max_gray_level > 16)
//若灰度级数大于16，则将图像的灰度级缩小至16级，减小灰度共生矩阵的大小。
{
    for (int i = 0; i < height; i++)
    {
        int* srcdata = src.ptr<int>(i);
        for (int j = 0; j < width; j++)
        {
            srcdata[j] = (int)srcdata[j] / gray_level;
        }
    }

    dst.create(gray_level, gray_level, CV_32SC1);
    dst = Scalar::all(0);
    for (int i = 0; i < height; i++)
    {
        int* srcdata = src.ptr<int>(i);
        for (int j = 0; j < width - 1; j++)
        {
            int rows = srcdata[j];
            int cols = srcdata[j + 1];
            dst.ptr<int>(rows)[cols]++;
        }
    }
}
```

像素灰度最大值加1即为该矩阵所拥有的灰度级数。检查最大的灰度级别是否超过了16。 如果最大灰度级别超过16，则遍历图像的所有行，获取当前行的所有像素值。遍历当前行的所有像素，将当前像素的灰度值除以灰度级别gray_level，将图像的灰度级别从大于16的级别降低到16级。循环结束。

重新创建输出图像 dst 的尺寸，设置灰度级别为16，并使用32位有符号整数数据类型，并且将输出图像 dst 初始化为全零矩阵。外层循环遍历图像的所有行，通过获取该行的所有像素值。内层循环遍历当前行，但不包括最后一个像素，获取当前像素和下一个像素的灰度值。然后，增加灰度共生矩阵中对应位置的计数。

```
else//若灰度级数小于16，则生成相应的灰度共生矩阵
{
    dst.create(max_gray_level, max_gray_level, CV_32SC1)
    dst = Scalar::all(0);
    for (int i = 0; i < height; i++)
    {
        int* srcdata = src.ptr<int>(i);
        for (int j = 0; j < width - 1; j++)
        {
            int rows = srcdata[j];
            int cols = srcdata[j + 1];
            dst.ptr<int>(rows)[cols]++;
        }
    }
}
```

如果灰度级别小于16，则生成相应的灰度共生矩阵。根据最大的灰度级别max_gray_level，创建一个二维矩阵dst，数据类型为32位有符号整数。将dst矩阵的所有元素初始化为0。 外层循环遍历图像的所有行，获取当前行的所有像素值。内层循环遍历当前行，但不包括最后一个像素，获取当前像素的灰度值和获取下一个像素的灰度值。在灰度共生矩阵中，增加对应位置的计数。循环结束。

45度：在0度的基础上将i值下移一个距离。

```
else
{
    dst.create(max_gray_level, max_gray_level, CV_32SC1);
    dst = Scalar::all(0);
    for (int i = 0; i < height - 1; i++)
    {
        int* srcdata = src.ptr<int>(i);
        int* srcdata1 = src.ptr<int>(i + 1);
        for (int j = 0; j < width - 1; j++)
        {
            int rows = srcdata[j];
            int cols = srcdata1[j + 1];
            dst.ptr<int>(rows)[cols]++;
        }
    }
}
```

90度：在45度基础上将j+1的值改为j的值，即向左移一个距离。

```
else
{
    dst.create(max_gray_level, max_gray_level, CV_32SC1);
    dst = Scalar::all(0);
    for (int i = 0; i < height - 1; i++)
    {
        int* srcdata = src.ptr<int>(i);
        int* srcdata1 = src.ptr<int>(i + 1);
        for (int j = 0; j < width; j++)
        {
            int rows = srcdata[j];
            int cols = srcdata1[j];
            dst.ptr<int>(rows)[cols]++;
```

135度：在90度的基础上再向左移一个单位。

```
dst.create(max_gray_level, max_gray_level, CV_32SC1);
dst = Scalar::all(0);
for (int i = 0; i < height - 1; i++)
{
    int* srcdata = src.ptr<int>(i);
    int* srcdata1 = src.ptr<int>(i + 1);
    for (int j = 1; j < width; j++)
    {
        int rows = srcdata[j];
        int cols = srcdata1[j - 1];
        dst.ptr<int>(rows)[cols]++;
    }
```

3.2.特征值计算：

在得到各方向的灰度共生矩阵后，通过对其进行统计分析可以得到五个常用的纹理特征，其中代码编写思路如下：

首先对输入的灰度共生矩阵进行归一化处理，具体操作为：首先计算输入灰度共生矩阵的矩阵元素和，之后新建Mat类copy存储归一化灰度共生矩阵，遍历原矩阵，将原矩阵元素分别除以矩阵元素总和，得到归一化灰度共生矩阵。

代码如下：

```
void feature_computer(Mat& src, double& Asm, double& Eng, double& Con, double& Idm, double& Rel)
{//src是得到的灰度共生矩阵，Asm是能量，Eng是熵，Con是对比度，Idm是逆差矩，Rel是相关
    int height = src.rows;
    int width = src.cols;
    int total = 0;
    for (int i = 0; i < height; i++)
    {
        int* srcdata = src.ptr<int>(i);
        for (int j = 0; j < width; j++)
        {
            total += srcdata[j];//求灰度共生矩阵的元素总和
        }
    }
    Mat copy;
    copy.create(height, width, CV_64FC1);
    for (int i = 0; i < height; i++)
    {
        int* srcdata = src.ptr<int>(i);
        double* copydata = copy.ptr<double>(i);
        for (int j = 0; j < width; j++)
        {
            copydata[j] = (double)srcdata[j] / (double)total;//图像每一个像素的的值除以像素总和
        }
    }
```

然后对得到的归一化灰度共生矩阵进行纹理特征计算，其中能量、熵、对比度、逆差矩的计算较为相似，因此将其放入一个for循环中共同计算，代码如下：

```
for (int i = 0; i < height; i++)
{
    double* srcdata = copy.ptr<double>(i);
    for (int j = 0; j < width; j++)
    {
        Asm += srcdata[j] * srcdata[j];//能量
        if (srcdata[j] > 0)
            Eng -= srcdata[j] * log(srcdata[j]);//熵
        Con += (double)(i - j) * (double)(i - j) * srcdata[j];//对比度
        Idm += srcdata[j] / (1 + (double)(i - j) * (double)(i - j));//逆差矩
    }
}
```

相关性的计算涉及到u1,u2,delta1,delta2的计算，因此分别循环计算这些值，最后汇总得到相关性特征值的结果，代码如下：

```
double u1 = 0, u2 = 0, delta1 = 0, delta2 = 0, temp = 0;
for (int i = 0; i < height; i++)
{
    double* srcdata = copy.ptr<double>(i);
    temp = 0;
    for (int j = 0; j < width; j++)
    {
        temp += srcdata[j];
    }
    u1 += temp * i;
}
for (int j = 0; j < width; j++)
{
    double* srcdata = copy.ptr<double>(j);
    temp = 0;
    for (int i = 0; i < height; i++)
    {
        temp += srcdata[i];
    }
```

```
    u2 += temp * j;
}
int n = 0;
for (int i = 0; i < height; i++)
{
    double* srcdata = copy.ptr<double>(i);
    temp = 0;
    for (int j = 0; j < width; j++)
    {
        temp += srcdata[j];
        n += i * j * srcdata[j];
    }
    delta1 += temp * (i - u1) * (i - u1);
}
for (int j = 0; j < width; j++)
{
    double* srcdata = copy.ptr<double>(j);
    temp = 0;
    for (int i = 0; i < height; i++)
    {
        temp += srcdata[i];
    }
    delta2 += temp * (j - u2) * (j - u2);
}
Re1 = (n - u1 * u2) / (delta1 * delta2);//相关
```

在主函数中调用灰度共生矩阵函数及纹理特征计算函数，可以得到如下图的结果：

```
int main()
{
    cout << "Hello world!" << endl;
    Mat color =imread("C:/Users/ASUS/Desktop/帅帅图图/微信图片_20221121193810.jpg");//输入的彩色图像
    imshow("color.bmp", color);
    Mat src_gray;//彩色图像转成的灰度图像
    cvtColor(color, src_gray, COLOR_BGR2GRAY);
    int n = 16;//一个部分包含的像素宽/长;
    int rows = color.rows;
    int cols = color.cols;
    const int num = 30;
    double energy0[num][num]={};//能量
    double energy45[num][num]={};//能量
    double energy90[num][num] = {};//能量
    double energy135[num][num] = {};//能量
    double relevance0[num][num] = {};//相关
    double relevance45[num][num] = {};//相关
    double relevance90[num][num] = {};//相关
    double relevance135[num][num] = {};//相关
    double entropy0[num][num] = {};//熵
    double entropy45[num][num] = {};//熵
    double entropy90[num][num] = {};//熵
    double entropy135[num][num] = {};//熵
    double contrast0[num][num] = {};//对比度
    double contrast45[num][num] = {};//对比度
    double contrast90[num][num] = {};//对比度
    double contrast135[num][num] = {};//对比度
    double smoment0[num][num] = {};//逆差矩
    double smoment45[num][num] = {};//逆差矩
    double smoment90[num][num] = {};//逆差矩
    double smoment135[num][num] = {};//逆差矩
```

```
int x = 0, y = 0;
for (int i = 0; i < rows - n + 1; i += n)
{
    for (int j = 0; j < cols - n + 1; j += n)
    {
        Mat part(n, n, CV_32FC1);
        Mat dst_0, dst_90, dst_45, dst_135;
        for (int k = 0; k < n; k++)
        {
            for (int g = 0; g < n; g++)
            {
                part.at<float>(k, g) = src_gray.at<uchar>(i + k, j + g);
            }
        }
        getglcm_horison(part, dst_0);
        getglcm_vertical(part, dst_45);
        getglcm_45(part, dst_90);
        getglcm_135(part, dst_135);
        feature_computer(dst_0, energy0[x][y], entropy0[x][y],contrast0[x][y], smoment0[x][y],relev
        feature_computer(dst_45, energy45[x][y], entropy45[x][y], contrast45[x][y], smoment45[x][y]
        feature_computer(dst_90, energy90[x][y], entropy90[x][y], contrast90[x][y], smoment90[x][y]
        feature_computer(dst_135, energy135[x][y], entropy135[x][y], contrast135[x][y], smoment135[
        if (x == 0 && y == 0)
        {
            cout << "以图像左上角第一个像素组为例：\n";
            cout << "0度方向能量为" << energy0[x][y] << ",熵为" << entropy0[x][y] << ",对比度为" <<
            cout << "\n45度方向能量为" << energy45[x][y] << ",熵为" << entropy45[x][y] << ",对比度为
            cout << "\n90度方向能量为" << energy90[x][y] << ",熵为" << entropy90[x][y] << ",对比度为
            cout << "\n135度方向能量为" << energy135[x][y] << ",熵为" << entropy135[x][y] << ",对比
            //能量大时纹理粗，能量小时纹理细；对比度越大则纹理沟纹越深，效果越清晰；相关则用来衡量不
            //熵越大说明纹理复杂程度较大；逆差矩是图像局部灰度均匀性的度量，如果图像局部的灰度均匀，
        }
        y++;
    }
    x++;
    y = 0;
}

return 0;
```

其中，能量反映图像纹理的稳定性和一致性；熵衡量图像中纹理的复杂度和随机性；对比度反映图像的清晰度和纹理的沟纹深浅；逆差矩反映图像中纹理的方向性和连续性；相关性描述图像中素灰度值之间的空间依赖关系。

通过对这些纹理特征的总结和分析，我们可以解决纹理分类、地物提取等实际问题。

3.3.纹理特征图：

遍历图像，分析每个窗口block的灰度特征矩阵：

```
vector<vector<double>> GLDM(int& TH, int& TW, Mat M)
{
    int height = M.rows;
    int width = M.cols;
    int i, j, m, n;
    //int b = (height / TH) * (width / TW);
    vector<vector<double>> feature;
    vector<double> B;
    Mat block = Mat::zeros(TH, TW, CV_8UC1);
    Mat dst_horison;
    for (i = TH / 2; i < height - TH / 2; i++)
    {
        for (j = TW / 2; j < width - TW / 2; j++)//窗口中心移动遍历图像给block赋值
        {
            for (m = 0; m < TH; m++)
            {
                for (n = 0; n < TW; n++)//卷积
                {
                    block.at<uchar>(m, n) =  M.at<uchar>(i - TH / 2 + m, j  - TW / 2 + n) ;
                }
            }
```

生成0、90、45、135灰度共生矩阵和5个特征值，存入容器：

```
//block的灰度共生矩阵0度
getglcm_horison(block, dst_horison);
//统计block的特征值
double asm_horison = 0, eng_horison = 0, con_horison = 0, idm_horison = 0,  rel_horison = 0;
feature_computer(dst_horison, asm_horison, eng_horison, con_horison, idm_horison, rel_horison);//block相当于8位
//填入容器
B.push_back(asm_horison);
B.push_back(eng_horison);
B.push_back(con_horison);
B.push_back(idm_horison);
B.push_back(rel_horison);
//90
getglcm_vertical(block, dst_horison);
double eng_horison90 = 0, con_horison90 = 0, idm_horison90 = 0, asm_horison90 = 0, rel_horison90 = 0;
feature_computer(dst_horison, asm_horison90, eng_horison90, con_horison90, idm_horison90, rel_horison90);//bloc
B.push_back(asm_horison90);
B.push_back(eng_horison90);
B.push_back(con_horison90);
B.push_back(idm_horison90);
B.push_back(rel_horison90);
//45
getglcm_45(block, dst_horison);
double eng_horison45 = 0, con_horison45 = 0, idm_horison45 = 0, asm_horison45 = 0, rel_horison45 = 0;
feature_computer(dst_horison, asm_horison45, eng_horison45, con_horison45, idm_horison45, rel_horison45);//bloc
B.push_back(asm_horison45);
B.push_back(eng_horison45);
B.push_back(con_horison45);
B.push_back(idm_horison45);
B.push_back(rel_horison45);
//135
getglcm_135(block, dst_horison);
double eng_horison135 = 0, con_horison135 = 0, idm_horison135 = 0, asm_horison135 = 0, rel_horison135 = 0;
feature_computer(dst_horison, asm_horison135, eng_horison135, con_horison135, idm_horison135, rel_horison135);
B.push_back(asm_horison135);
B.push_back(eng_horison135);
B.push_back(con_horison135);
B.push_back(idm_horison135);
B.push_back(rel_horison135);
feature.push_back(B);
B.clear();
```

生成纹理特征图（以asm特征图为例）：

```
// feature1生成纹理特征图
for (i = TH / 2; i < height - TH / 2; i++)
{
    for (j = TW / 2; j < width - TW / 2; j++)
    {
        Texture.at<uchar>(i, j) = feature[q][0]*50;
        q++;
    }
}
namedWindow("纹理特征图", WINDOW_NORMAL);
imshow("纹理特征图", Texture);  // 显示图片
waitKey();
imwrite("C:/Users/18440/Desktop/纹理特征图.jpg", Texture);
```

确定阈值并标识颜色：

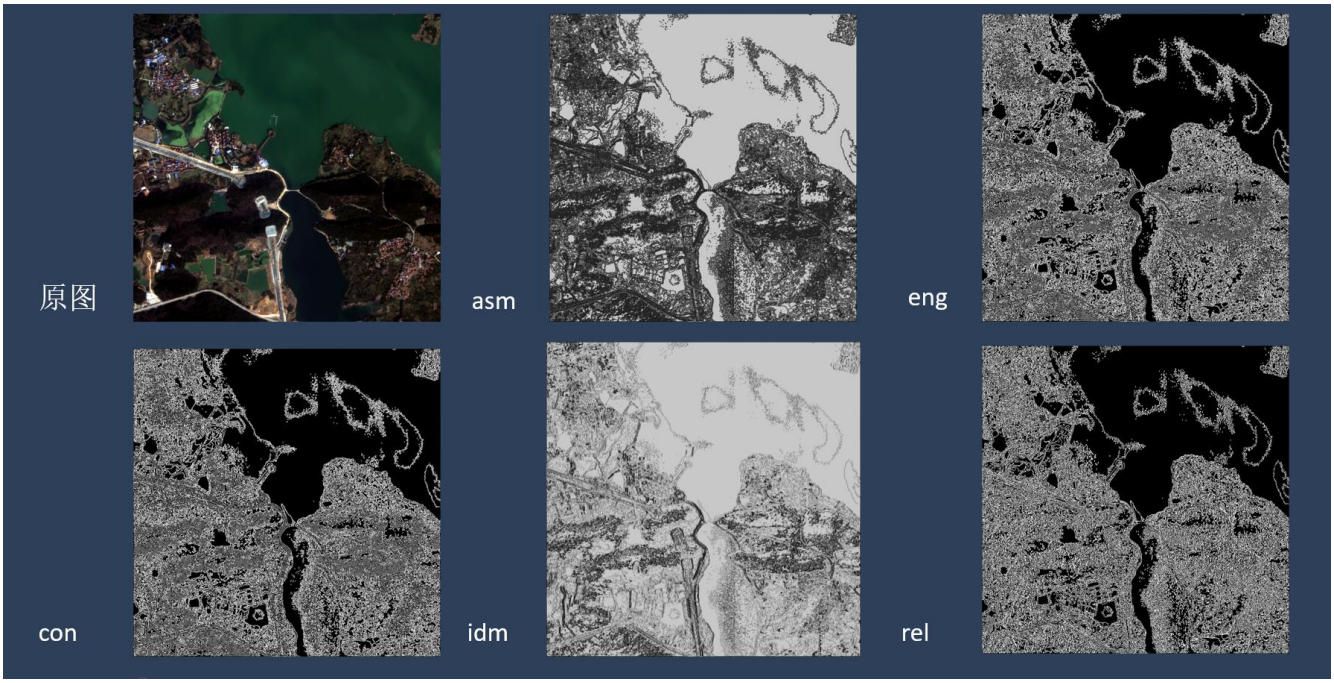
```
//确定阈值进行分类标示
int k = 0;
for (i = TH / 2; i < height - TH / 2; i++)
{
    for (j = TW / 2; j < width - TW / 2; j++)
    {
        if (feature[k][0] * 100<20&& feature[k][3] * 130 < 73) {
            M0.at<Vec3b>(i, j)=Vec3b(255,0,0);
        }
        k++;
    }
}
namedWindow("", WINDOW_NORMAL);
imshow("林地", M0);  // 显示图片
waitKey();
imwrite("C:/Users/18440/Desktop/结果图.jpg", M0);
feature.clear();
return 0;
```

4. 实验结果与分析

基于灰度共生矩阵的特征值的输出：

```
以图像左上角第一个像素组为例：
0度方向能量为0.75625,熵为0.593147,对比度为0.283333,逆差矩为0.933333,相关为-31.1628
45度方向能量为0.756458,熵为0.583177,对比度为0.233333,逆差矩为0.943333,相关为-25.4009
90度方向能量为0.755338,熵为0.499278,对比度为0.493333,逆差矩为0.897333,相关为-24.9173
135度方向能量为0.749886,熵为0.572501,对比度为0.471111,逆差矩为0.897778,相关为-70.0195
```

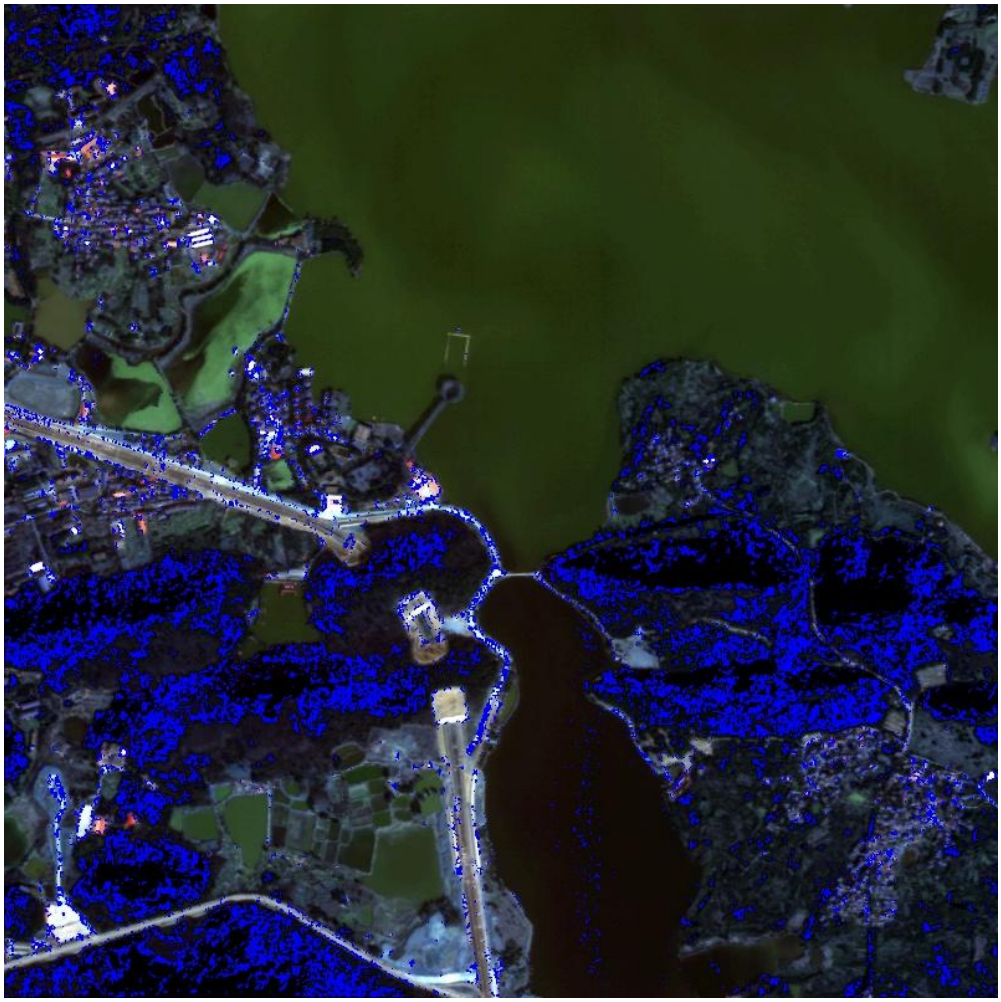
程序可生成4个不同方向的五个特征值图像，如图：



提取林地结果图如下：

原图：

分析图：



5. 结论

最后调试的参数为：模板的长度和宽度为3*3，feature[k][0]＊100<20&& feature[k][3]＊130＜73，即方向为0度，且选取特征值为能量，以及其阈值为<0.2；方向为0度，特征值为对比度，并且阈值为<73/130。

根据此参数进行标识得到林地提取图，由结果图像看出，提取林地对大片林地的部分识别效果比较好，水库和道路有部分被误识别成林地，总体识别效果不错。

在林地提取之后，小组成员也使用此程序对于其他图像进行不同方向、不同特征值、不同阈值的尝试和更改，进一步深入对于应用的探索。

6. 小组成员分工说明

6.1.编程方面：

祁红利负责四个方位的灰度共生矩阵的生成;程欣悦负责归一化和五个纹理特征值的计算和提取；阮思琳负责基于灰度共生矩阵的纹理特征提取的遥感图像林地提取的应用。

6.2.PPT制作和汇报方面：

祁红利负责灰度共生矩阵的纹理特征提取与分析的原理和小组分工的部分;程欣悦负责灰度共生矩阵的编程实现和纹理特征提取的编程实现的部分;阮思琳灰度共生矩阵在图像处理中应用的编程实现和总结的部分。

6.3.实习报告方面：

每个人按照自己算法编程部分的分工完成相应的研究内容和目标、算法原理、流程设计、编程结果，并且一起讨论实验结果并且进行分析，撰写自己的感悟和心得。

7. 实习心得与体会

祁红利：学习了解掌握了灰度共生矩阵的产生和构成，对于数字图像处理课程上所学习的理论进行了实践和深入。自行编写了四个方向上灰度共生矩阵生成的函数，遍历寻找最大灰度以求出灰度级数，对灰度级数进行压缩，将不同方向上的像素值与中心像素建立对应的计数关系，并且使用矩阵记录，生成了灰度共生矩阵。而后其他小组成员编写了特征值的计算函数，并且对于图像进行了林地提取的应用。参考了一些相关文献，扩大了我的视野。虽然后面的部分不是由我编写程序实现，但是我们也在一直一起讨论、集思广益，不论哪一步遇到了问题都一起关注，包括后面的应用要对图像进行裁剪和多个图像的特征值处理，然后将这些特征值使用矩阵表示，再对矩阵设置阈值从而找出具有相似特征的部分，并且对于不同特征值处理效果进行讨论和评价，这也增强了我的合作能力以及沟通交流的能力。一份较大工作量的程序在小组作业的分配体制下能高效完成。

程欣悦：在本次小组任务中，我主要负责基于灰度共生矩阵的纹理特征计算，在编程中，我通过对灰度共生矩阵的归一化操作后的遍历操作实现了这些特征的获取，其中如何简便算法，以精炼的代码实现目的是我主要在思考的问题，一开始我分别将五个特征定义为不同的函数，在这种情况下，每一个函数内都要进行一次对灰度共生矩阵的遍历，使代码十分冗长，同时考虑到后续的处理操作可能会同时用到这五个特征，我最后选择将五个特征放入同一个函数中输出，这样不仅减少了代码量，也方便了后续同学在应用上的处理操作。在这个过程中，我更加深入地了解了这一个个数字背后的真正含义，熵、能量、逆差矩、对比度……这一个个数值背后映射出图像纹理各方面的特征，通过对这些数据的调用，我们能够解决许多实际问题，如纹理分类和不同地

物的提取等，希望之后可以将我们在这节实习课程上学习到的知识应用到更多的领域，解决更多的实际问题。

阮思琳：在本次小组合作中，我负责主函数的编写，对每个中心像元进行灰度共生矩阵和特征值提取，从而生成多个纹理特征图，最后分析纹理特征图实现目标地物的识别。在最后难度较大的小组作业中，我意识到要真正实现一个复杂的、严密特征提取分析程序所需要的远远不只于我们在课堂上所学的简单原理。在提取特征值分析的过程中，我发现单纯裁剪图片得到的特征值差异并不明显，最后的识别误差也很大，在询问老师之后，改进采用遍历窗口的模式，将每个窗口的特征值赋予中心像素，再生成纹理特征图提取地物，最后误差减小了很多；在林地提取确定阈值的过程中，我利用画图工具对纹理特征图中差异明显的林地进行粗略的灰度估计，在一次次调试中确定最好的阈值参数，最终生成图像。在小组合作中，我们完成了林地特征提取分析程序的编写，虽然尚不成熟，但在不断面对问题、解决问题中学到许多知识，提高了团队合作能力、编程实践能力、解决问题能力，对图像处理基本原理和应用有了更深刻的认识。

参考文献

[1] 冯建辉，杨玉静. 基于灰度共生矩阵提取纹理特征图像的研究. 昆明理工大学国土资源工程学院, 2007.

[2] 刘凯，汤国安，陶旸，蒋圣.基于灰度共生矩阵的DEM地形纹理特征量化研究. 南京师范大学虚拟地理环境教育部重点实验室.2012.

[3] 胡启明.基于灰度共生矩阵的地形纹理特征量化研究.福州大学.2019.