1 实习原理

1.1 基于特征指数的遥感专题信息提取

1.1.1 植被指数：

① 比值植被指数：

由于近红外波段与可见光红光波段对绿色植物的光谱响应有很大的不同，它们之间的比值可以反映出红光波段与近红外波段反射率之间的差。

$$\mathrm{RVI} = \frac{R_n}{R_t}$$

② 归一化植被指数：

$$\mathrm{NDVI} = (\mathrm{NIR} - \mathrm{R})/(\mathrm{NIR} + \mathrm{R})$$

③ 土壤调节植被指数：

降低土壤亮度对植被指数的影响

$$\mathrm{SAVI} = \frac{R_n - R_r}{R_n + R_r + L}(1 + L) \quad (\mathrm{L}\ 取\ 0.5)$$

1.1.2 水体指数：

① 水体归一化指数：

主要利用了在近红外波段水体强吸收几乎没有反射而植被反射率很强的特点，通过抑制植被和突出水体用来提取影像中的水体信息。

$$\mathrm{NDWI} = \frac{b_{green} - b_{NIR}}{b_{green} + b_{NIR}}$$

② 自动水体提取指数 AWEI：

针对水体信息提取存在的分类精度低、阈值选取相对不固定等

因素，利用 LandsatTM 影像进行了实验，提出了 AWEI，分别适用于没有阴影的场景、阴影较多的场景。（SWIR1 为中红外，SWIR2 为远红外）

$$\mathrm{AWEI} = \frac{4(GREEN - SWIR1)}{(0.25NIR + 2.75SWIR2)}$$

1.1.3 **建筑指数：**

① 建筑归一化指数：

利用城镇用地灰度值在 TM4，5 两个波段间与其他地类的可分性，提出了归一化差值建筑用地指数。

$$\mathrm{NDBI} = \frac{(TM5 - TM4)}{(TM5 + TM4)}$$

② 建筑用地指数 IBI：

将遥感影像的多个原始波段压缩为三个专题指数:土壤调节植被指数 SAV，归一化建筑指数 NDBI 和改进型归一化水体指数 MNDWI。

$$\mathrm{IBI} = \frac{\{NDBI - (SAVI + MNDW)/2\}}{\{NDBI + (SAVI + MNDW)/2\}}$$

1.1.4 **阈值分割：**

专题指数图像可以突出专题与其他地物之间的光谱差异，但是要提取专题边界，还需要 使用阈值对指数图像进行分割。为保证最佳的分割效果。利用图像中要提取的目标区域与其背景在灰度特性上的差异，把图像看作具有不同灰度级的两类区域(目标区域和背景区域)的组合，选取一个比较合理的阈值，以确定图像中每个像素点应该属于目标区域还是背景区域，从而产生相应的二值图像。

本次实习使用迭代法，其思想是设置阈值的初始值为图像灰度最大值和最小值的平均，根据阈值划分图像为目标和背景，并分别将其灰度值求和，计算目标和背景的平均灰度，并判断阈值是否等于目标和背景平均灰度的和的平均，若相等，则阈值即为其平均，否则，将阈值设置为目标和背景灰度平局值的和的一半，继续迭代，直至计算出阈值。

1.2 遥感影像阴影检测

1.2.1 基于 HSV 彩色空间的阴影检测：

在 HSV 彩色空间中，遥感影像阴影区域与非阴影区域相比有以下 3 个特点：阴影区域具有更大的色调值;阴影区域的散射光线主要来自波长更短的蓝紫色光，因此具有高饱和度值;阴影区域太阳光线被阻挡，导致低亮度值。 HSV 彩色体统基于柱坐标系，将 RGB(笛卡尔坐标系) 映射至 HSV(柱坐标系) 的方程如下：

$$\mathrm{V}=\frac{1}{3}(\mathrm{R}+\mathrm{G}+\mathrm{B})$$

$$S=1-\frac{3}{\mathrm{R}+\mathrm{G}+\mathrm{B}}min(\mathrm{R},\mathrm{G},\mathrm{B})$$

$$H=\begin{cases}\theta & \mathrm{B}\le G\\ 360^{\circ}-\theta & \mathrm{B}>G\end{cases}$$

$$\theta=arccos\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}$$

① 将彩色影像进行 RGB 到 HSV 色彩空间变换。

② 依据阴影区域的高色调值、低亮度值和高饱和度特性，定义 M=(S-V)/(H+S+V)进行阈值选择分割出区域。

③ 对分割出的区域进行数学形态学的闭运算处理，从而得到较为精确的阴影区域。

1.2.2 基于 C1C2C3 彩色空间的阴影检测：

对阴影区域进行检测: 将彩色影像进行 RGB 到 HSV 色彩空间变换，依据阴影区域的高色调值、低亮度值和高饱和度特性，定义 M= (S－V) /(H+S+V) 进行阈值选择分割出阴影区域形态学后处理。

利用 RGB 彩色空间单色波段进行阴影检测精度低，可引入 C1C2C3 彩色空间进行阴影检测。在 C1C2C3 彩色空间的 C3 分量中，阴影区域主要占据的是高像素值，通过对 C3 分量图采用阈值分割的方法得到初步阴影区域。但原始影像中的偏蓝色地物在 C3 分量中就具有很高的像素值，必须将这些区域从阴影区域中去除。为此，需要将 C3 分量图和 B 分量图相结合，采用双阈值来进行阴影检测。只有在 C3 分量中高于某个阈值，并在 B 分量中低于某个阈值的区域，才被检测成为阴影区域。

$$C_1 = \tan^{-1}(\frac{R}{(\max(G,B))})$$
$$C_2 = \tan^{-1}(\frac{G}{(\max(R,B))})$$
$$C_3 = \tan^{-1}(\frac{B}{(\max(R,G))})$$

2 实习步骤

2.1 基于特征指数的遥感专题信息提取

使用 opencv 库来存储和处理图像，使用 Mat 类保存图像，使用 opencv 内置函数处理图像内计算及图像可视化等等功能。

2.1.1 植被指数

①比值植被指数：根据公式编程

$$\mathrm{RVI} = \frac{R_n}{R_t}$$

```
Mat RVI(Mat Rn, Mat Rr)
{
    //Rn 近红外波段
    //Rr 红波段
    //RVI=Rn/Rr
    Mat Ratio(Rn.size(), CV_64FC1);
    divide(Rn, Rr, Ratio);//相除
    Ratio.convertTo(Ratio, CV_64FC1);//转换格式 防止double类型丢失
    return Ratio;
}
```

②归一化植被指数：根据公式编程，由于公式格式相同，所以水体归一化指数与建筑归一化指数方法相同，仅输入参数不同。

$$\mathrm{NDVI} = (\mathrm{NIR} - \mathrm{R})/(\mathrm{NIR} + \mathrm{R})$$

```
Mat Normalize(Mat s1, Mat s2)//NDVI植被归一化指数、NDWI归一化水体指数、NDBI归一化裸地指数
{
 //NDVI
 //NIR 近红外波段
```

```
//R 红光波段
//NDVI=(NIR-R)/(NIR+R)

//NDWI
//G 绿波段
//nR 近红外波段
//NDWI=(G-nR)/(G+nR)

//NDBI
//mR 中红外波段
//nR 近红外波段
//NDBI=(mR-nR)/(mR+nR)



Mat m(s1.size(), CV_64FC1);   //分子
Mat n(s2.size(), CV_64FC1);   //分母

subtract(s1, s2, m);
add(s1, s2, n);

Mat NDVI(s1.size(), CV_64FC1);    //计算得到的指数 用double存储

m.convertTo(m, CV_64FC1);   //运算会导致类型不为double
n.convertTo(n, CV_64FC1);   // 此处进行强制转换

divide(m, n, NDVI);
return NDVI;}
```

③ 土壤调节植被指数：根据公式编程

$$\mathrm{SAVI} = \frac{R_n - R_r}{R_n + R_r + L}(1 + L) \quad (\text{L 取 } 0.5)$$

```
#define L 0.5
Mat SAVI(Mat Rn, Mat Rr)//土壤调节植被指数
{
    //Rn近红外波段
    //Rr红光波段
    //SAVI=(Rn-Rr)(1+L)/(Rn+Rr+L)  (取 L=0.5)

    Mat SAVI(Rn.size(), CV_64FC1);
    Mat m(Rn.size(), CV_64FC1);    //分子
    Mat n(Rn.size(), CV_64FC1);    //分母
    subtract(Rn , Rr,m);
    m.convertTo(m, CV_64FC1);
    multiply(m, 1 + L, m);

    add(Rn, Rr, n);
    add(n, L, n);
    n.convertTo(n, CV_64FC1);

    divide(m, n, SAVI);
    SAVI.convertTo(SAVI, CV_64FC1);
    return SAVI;
}
```

2.2.2 水体指数

① 水体归一化指数：归一化方法编程与植物归一化相同，共用函数

$$\mathrm{NDWI} = \frac{b_{green} - b_{NIR}}{b_{green} + b_{NIR}}$$

② 自动水体提取指数 AWEI：根据公式编程

$$\mathrm{AWEI} = \frac{4(GREEN - SWIR1)}{(0.25NIR + 2.75SWIR2)}$$

```
Mat AWEI(Mat G, Mat nR, Mat mR, Mat dR) {
    //自动水体提取指数
    //G 绿光
    //nR 近红外波段
    //mR 中红外波段
    //dR 远红外波段
    //AWEI=4(G-mR)/(0.25nR+2.75dR)

    Mat m(G.size(), CV_64FC1);//分子
    Mat n(G.size(), CV_64FC1);//分母
    Mat t(G.size(), CV_64FC1);

    nR.convertTo(nR, CV_64FC1);
    dR.convertTo(nR, CV_64FC1);
```

```
subtract(G, mR, m);
multiply(m, 4, m);
m.convertTo(m, CV_64FC1);

multiply(nR, 0.25, n);
multiply(dR, 2.75, t);
n.convertTo(n, CV_64FC1);
t.convertTo(t, CV_64FC1);
add(n, t, n);
n.convertTo(n, CV_64FC1);

Mat img(G.size(), CV_64FC1);
divide(m, n, img);
img.convertTo(img, CV_64FC1);
return img;}
```

2.1.3 建筑指数：

① 建筑归一化指数：归一化方法编程与植物归一化相同，共用函数

$$\mathrm{NDBI} = \frac{(TM5 - TM4)}{(TM5 + TM4)}$$

② 建筑用地指数 IBI：

根据公式编程，其中根据格式也利用了归一化函数

$$\mathrm{IBI} = \frac{\{NDBI - (SAVI + MNDW)/2\}}{\{NDBI + (SAVI + MNDW)/2\}}$$

```
Mat IBI(Mat NDBI, Mat SAVI, Mat NDWI) {
    //建筑用地指数
    //NDBI归一化建筑指数
    //SAVI土壤调节植被指数
    //NDWI归一化水体指数
    //IBI=(NDBI-(SAVI+NDWI)/2)/(NDBI+(SAVI+NDWI)/2)
    Mat m(NDBI.size(), CV_64FC1);
    add(SAVI, NDWI, m);
    divide(m, 2, m);
    m.convertTo(m, CV_64FC1);

    Mat img = Normalize(NDBI, m);

    return img;
}
```

2.1.4 图像伪彩色化：

根据地物类型不同，将图像伪彩色化。

```
#define Plant 0
#define Water 1
#define Buliding 2
Mat Color(Mat Index,int type)//伪彩色化
{
    Mat Img(Index.size(), CV_8UC3);
    if (type == Plant)
    {
        for (int i = 0; i < Index.rows; i++)
        {
            for (int j = 0; j < Index.cols; j++)
            {
                if (Index.at<double>(i, j) > 0)
                {
                    Img.at<Vec3b>(i, j)[0] = 0;
                    Img.at<Vec3b>(i, j)[1] =
saturate_cast<uchar>(Index.at<double>(i, j) * 255);   //添加绿色;
                    Img.at<Vec3b>(i, j)[2] = 0;
                }
                else
                {

                    Img.at<Vec3b>(i, j)[0] = 0;
                    Img.at<Vec3b>(i, j)[1] = 0;
                    Img.at<Vec3b>(i, j)[2] = 0;
                }
            }
        }
    }
    else if (type == Water)
    {
        for (int i = 0; i < Index.rows; i++)
            for (int j = 0; j < Index.cols; j++)
            {
                if (Index.at<double>(i, j) > 0)
                {
                    Img.at<Vec3b>(i, j)[0] =
saturate_cast<uchar>(Index.at<double>(i, j) * 255);  //添加蓝色
                    Img.at<Vec3b>(i, j)[1] = 0;
                    Img.at<Vec3b>(i, j)[2] = 0;
                }
                else
```

```
                {
                    Img.at<Vec3b>(i, j)[0] = 0;
                    Img.at<Vec3b>(i, j)[1] = 0;
                    Img.at<Vec3b>(i, j)[2] = 0;
                }
            }
        }
        else if (type == Buliding)
        {
            for (int i = 0; i < Index.rows; i++)
                for (int j = 0; j < Index.cols; j++)
                {
                    if (Index.at<double>(i, j) > 0)
                    {
                        Img.at<Vec3b>(i, j)[0] = 0;
                        Img.at<Vec3b>(i, j)[1] =
    saturate_cast<uchar>(Index.at<double>(i, j) * 255);  //添加黄色
                        Img.at<Vec3b>(i, j)[2] =
    saturate_cast<uchar>(Index.at<double>(i, j) * 255);
                    }
                    else
                    {
                        Img.at<Vec3b>(i, j)[0] = 0;
                        Img.at<Vec3b>(i, j)[1] = 0;
                        Img.at<Vec3b>(i, j)[2] = 0;
                    }
                }
        }
        return Img;
    }
```

2.1.5 阈值分割二值化：

```
Mat Binary(Mat Index)
{
 Mat out(Index.size(), CV_8UC1);

 double temp = 0;
 double max = 0;
 double min = 255;

 //最大类间误差法
 //灰度拉伸 从0-255
 for (int i = 0; i < Index.rows; i++)
     for (int j = 0; j < Index.cols; j++)
```

```
    {
        temp = Index.at<double>(i, j);
        if (temp > max)
        {
            max = temp;
        }
        if (temp < min)
        {
            min = temp;
        }
    }
double scale = 255 / (max - min);

for (int i = 0; i <Index.rows; i++)
    for (int j = 0; j < Index.cols; j++)
    {
        out.at<uchar>(i, j) = (unsigned char)((int)((Index.at<double>(i, j) -
min) * scale));
    }

//动态计算阈值
int thre = 0;
double f1 = 0; double f2 = 0;          //前后景灰度频数
double s_temp = 0; double s = 0;      //方差
double avg1 = 0; double avg2 = 0;     //前后景分别均值

for (int t = 0; t < 256; t++)           //找每一个阈值
{
    f1 = 0;
    f2 = 0;
    avg1 = 0;
    avg2 = 0;

    for (int i = 0; i < out.rows; i++)
        for (int j = 0; j < out.cols; j++)
        {
            int val = (int)(out.at<uchar>(i, j));
            if (val < t)
            {
                f1++;
                avg1 += val;
            }
            else
            {
```

```
                f2++;
                avg2 += val;
            }
        }
    avg1 /= f1;
    avg2 /= f2;
    s_temp = f1 * f2 * pow((avg1 - avg2), 2) / pow((out.rows - out.cols), 2);

    if (s_temp > s)    //找到动态规划后的阈值
    {
        thre = t;
        s = s_temp;
    }
}

Mat res(out.size(), out.type());

//二值化
for (int i = 0; i < res.rows; i++)
    for (int j = 0; j < res.cols; j++)
    {
        int val = (int)(out.at<uchar>(i, j));
        if (val > thre)
        {
            res.at<uchar>(i, j) = 255;
        }
        else
        {
            res.at<uchar>(i, j) = 0;
        }
    }

return res;
}
```

2.1.6 真彩色合成：

将三个 b，g，r 三个波段合成

```
Mat ColorRGB(Mat b, Mat g, Mat r)
{
    //真彩色合成
    Mat img(b.size(), CV_8UC3);
    Mat channel[3] = { b,g,r };

    merge(channel, 3, img);
    return img;
}
```

2.1.7 main 函数执行：

①读取图像

```
int main()
{
    Mat img_b, img_g, img_r, img_nR, img_mR, img_dR;//波段
    //读取图像
    printf_s("正在读取图像……\n");
    img_b = imread("./tm/tm1.tif", 0);  //蓝波段
    img_g = imread("./tm/tm2.tif", 0);//绿波段
    img_r = imread("./tm/tm3.tif", 0);//红波段
    img_nR = imread("./tm/tm4.tif", 0);//近红外波段
    img_mR = imread("./tm/tm5.tif", 0);//中红外波段
    img_dR = imread("./tm/tm6.tif", 0);//远红外波段
```

②输入所需参数进行相关地物图像指数计算，地物图像指数图像伪彩色化、二值化，并将其可视化，输出到文件夹Output_Img，不同地物反复执行

```
    //植被
    //RVI比值植被指数
    printf_s("等待RVI比值植被指数处理……\n");
    Mat mRVI =RVI(img_nR,img_r );
    Mat color_RVI = Color(mRVI,Plant);//伪彩色化(绿色)
    Mat Binary_RVI = Binary(mRVI);//二值化
    /*imshow("比值植被指数_伪彩色", color_RVI);*/
    imshow("比值植被指数_二值化", Binary_RVI);
    waitKey(0);
    imwrite("./Output_Img/比值植被指数_二值化.jpg", Binary_RVI);
    /*imwrite("./Output_Img/比值植被指数_伪彩色.jpg", color_RVI);*/

    //NDVI归一化植被指数
    printf_s("等待NDVI归一化植被指数处理……\n");
    Mat mNDVI = Normalize(img_nR, img_r);
    Mat color_NDVI = Color(mNDVI,Plant);//伪彩色化(绿色)
    Mat Binary_NDVI = Binary(mNDVI);//二值化
    imshow("归一化植被指数_伪彩色", color_NDVI);
    imshow("归一化植被指数_二值化", Binary_NDVI);
    waitKey(0);
    imwrite("./Output_Img/归一化植被指数_伪彩色.jpg", color_NDVI);
```

```
    imwrite("./Output_Img/归一化植被指数_二值化.jpg", Binary_NDVI);

//SAVI土壤调节植被指数
printf_s("等待SAVI土壤调节植被指数处理……\n");
Mat mSAVI = SAVI(img_nR, img_r);
Mat color_SAVI = Color(mSAVI,Plant);//伪彩色化(绿色)
Mat Binary_SAVI = Binary(mSAVI);//二值化
imshow("土壤调节植被指数_伪彩色", color_SAVI);
imshow("土壤调节植被指数_二值化", Binary_SAVI);
waitKey(0);
imwrite("./Output_Img/土壤调节植被指数_伪彩色.jpg", color_SAVI);
imwrite("./Output_Img/土壤调节植被指数_二值化.jpg", Binary_SAVI);


//水体
//NDWI归一化水体指数
printf_s("等待NDWI归一化水体指数处理……\n");
Mat mNDWI =Normalize(img_g, img_nR);
Mat color_NDWI = Color(mNDWI,Water);//伪彩色化(绿色)
Mat Binary_NDWI = Binary(mNDWI);//二值化
imshow("归一化水体指数_伪彩色", color_NDWI);
imshow("归一化水体指数_二值化", Binary_NDWI);
waitKey(0);
imwrite("./Output_Img/归一化水体指数_伪彩色.jpg", color_NDWI);
imwrite("./Output_Img/归一化水体指数_二值化.jpg", Binary_NDWI);

//AWEI自动水体提取指数
printf_s("等待AWEI自动水体提取指数处理……\n");
Mat mAWEI = AWEI(img_g,img_nR,img_mR,img_dR);
Mat color_AWEI = Color(mAWEI,Water);//伪彩色化(绿色)
Mat Binary_AWEI = Binary(mAWEI);//二值化
imshow("自动水体提取指数_伪彩色", color_AWEI);
imshow("自动水体提取指数_二值化", Binary_AWEI);
waitKey(0);
imwrite("./Output_Img/自动水体提取指数_伪彩色.jpg", color_AWEI);
imwrite("./Output_Img/自动水体提取指数_二值化.jpg", Binary_AWEI);

//建筑
//NDBI归一化裸地指数
printf_s("等待NDBI归一化裸地指数处理……\n");
Mat mNDBI =Normalize(img_mR,img_nR);
Mat color_NDBI = Color(mNDBI, Buliding);//伪彩色化(绿色)
Mat Binary_NDBI = Binary(mNDBI);//二值化
imshow("归一化裸地指数_伪彩色", color_NDBI);
```

```
imshow("归一化裸地指数_二值化", Binary_NDBI);
waitKey(0);
imwrite("./Output_Img/归一化裸地指数_伪彩色.jpg", color_NDBI);
imwrite("./Output_Img/归一化裸地指数_二值化.jpg", Binary_NDBI);

//IBI建筑用地指数
printf_s("等待IBI建筑用地指数处理……\n");
Mat mIBI = IBI(mNDBI,mSAVI,mNDWI);
Mat color_IBI = Color(mIBI, Buliding);//伪彩色化(绿色)
Mat Binary_IBI = Binary(mIBI);//二值化
imshow("建筑用地指数_伪彩色", color_IBI);
imshow("建筑用地指数_二值化", Binary_IBI);
waitKey(0);
imwrite("./Output_Img/建筑用地指数_伪彩色.jpg", color_IBI);
imwrite("./Output_Img/建筑用地指数_二值化.jpg", Binary_IBI);
```

③生成真彩色图像

```
//真彩色合成
printf_s("等待真彩色合成处理……\n");
Mat Color=ColorRGB(img_b, img_g, img_r);
imshow("真彩图", Color);
waitKey(0);
imwrite("./Output_Img/真彩图.jpg", Color);
printf_s("运行结束！");
```

2.2 遥感影像阴影检测

2.2.1 基于 HSV 彩色空间的阴影检测：

先进行 HSV 空间转换，再进行阴影识别：

```
void RGBToHSV(Mat r, Mat g,Mat b, Mat* h, Mat* s, Mat* v) {
  //HSV空间转换
  r.convertTo(r, CV_64FC1);
  g.convertTo(g, CV_64FC1);
  b.convertTo(b, CV_64FC1);

  Mat temp1(r.size(), CV_64FC1), temp2(r.size(), CV_64FC1);

  //  V=1/3 *(R+G+B)
  add(r, g, temp1);
  add(temp1, b, temp1);
  divide(temp1, 3, *v);
```

```
//  S=1-min(R,G,B)/V
for (int i = 0; i < r.rows; i++)
     for (int j = 0; j < r.cols; j++)
     {
          double val = 0;
          val = min(r.at<double>(i, j), g.at<double>(i, j));
          temp2.at<double>(i, j) = min(val, b.at<double>(i, j));
     }

divide(temp2, *v, temp2);
subtract(1, temp2, *s);


//  H = 0   360-0
for (int i = 0; i < r.rows; i++)
     for (int j = 0; j < r.cols; j++)
     {
          double valr = r.at<double>(i, j);
          double valg = g.at<double>(i, j);
          double valb = b.at<double>(i, j);

          if (valb <= valg)
          {
               h->at<double>(i, j) = 180 / Pi * acos(0.5 * (valr - valg + valr - valb)
                    / sqrt((valr - valg) * (valr - valg) + (valr - valb) * (valg -
valb)));
          }
          else
          {
               h->at<double>(i, j) = 360 - 180 / Pi * acos(0.5 * (valr - valg + valr -
valb)
                    / sqrt((valr - valg) * (valr - valg) + (valr - valb) * (valg -
valb)));
          }

     }

         }
    Mat Shadow(Mat h, Mat s, Mat v)
    {
     Mat Img;

     Mat res1, res2;  // 分子分母
```

```
subtract(s, v, res1);
add(h, s, res2);
add(res2, v, res2);

divide(res1, res2, Img);

return Img;
          }
```

2.2.2 基于 C1C2C3 彩色空间的阴影检测：

进行计算处理得到阴影：

```
Mat RGBToC1C2C3(Mat r, Mat g, Mat b)
{
    r.convertTo(r, CV_64FC1);
    g.convertTo(g, CV_64FC1);
    b.convertTo(b, CV_64FC1);

    Mat C3(r.size(), CV_64FC1);

    for (int i = 0; i < r.rows; i++)
        for (int j = 0; j < r.cols; j++)
        {
            C3.at<double>(i, j) = atan(b.at<double>(i, j) /
max(r.at<double>(i, j), g.at<double>(i, j)));
        }

    return C3;
        }
```

2.2.3 二值化处理：

图像在输出前需要进行二值化处理。方法与3.2.5相同。

2.2.4 连通域处理：

小面积去除，大面积合并。

```
Mat Merge(Mat img, int type)
{
 Mat temp = img.clone();

 if (type == HSV)
 {
     //腐蚀和膨胀操作形成连通域
     Mat element1 = getStructuringElement(MORPH_ELLIPSE, Size(2, 2));
```

```
    Mat element2 = getStructuringElement(MORPH_ELLIPSE, Size(3, 3));

    dilate(temp, temp, element1, Point(-1, -1), 6);
    dilate(temp, temp, element2, Point(-1, -1), 3);
    erode(temp, temp, element1, Point(-1, -1), 1);

    //寻找连通区域
    Mat res = temp.clone();
    vector<vector<Point>> contours;
    vector<Vec4i> hierarchy;
    findContours(res, contours, hierarchy, RETR_LIST, CHAIN_APPROX_SIMPLE);

    for (int i = 0; i < contours.size(); i++)
    {
        if (contourArea(contours[i]) < 100)    //选取连通区域的阈值
        {
            drawContours(res, contours, i, Scalar(0), -1);
        }
    }


    return res;
}
else if (type == C1C2C3)
{
    //腐蚀和膨胀操作形成连通域
    Mat element1 = getStructuringElement(MORPH_ELLIPSE, Size(2, 2));
    Mat element2 = getStructuringElement(MORPH_ELLIPSE, Size(3, 3));

    dilate(temp, temp, element1, Point(-1, -1), 2);
    dilate(temp, temp, element2, Point(-1, -1), 1);
    erode(temp, temp, element1, Point(-1, -1), 1);

    //寻找连通区域
    Mat res = temp.clone();
    vector<vector<Point>> contours;
    vector<Vec4i> hierarchy;
    findContours(res, contours, hierarchy, RETR_LIST, CHAIN_APPROX_SIMPLE);

    for (int i = 0; i < contours.size(); i++)
    {
        if (contourArea(contours[i]) < 100)    //选取连通区域的阈值
        {
            drawContours(res, contours, i, Scalar(0), -1);
```

```
            }
        }

        return res;
    }


}
```

2.2.5 main 运行：

读取图像后分成三通道，基于不同阴影检测方法进行运行，再进行图像二值化，并可视化和输出合并前和合并后的阴影检测图。

```
int main()
{
    Mat img = imread("Color.bmp");

    vector<Mat>channel;
    split(img, channel);    //分成B G R三频道

    Mat H(img.size(), CV_64FC1), S(img.size(), CV_64FC1), V(img.size(), CV_64FC1);

    //基于彩色模型的图像阴影检测
    printf_s("运行基于HSV空间的阴影检测……\n");
    RGBToHSV(channel[2], channel[1], channel[0], &H, &S, &V);//HSV空间转换
    Mat HSV_img=Shadow(H,S,V);  //HSV阴影识别
    Mat HSV_img_binary = Segment(HSV_img, HSV);
    imshow("合并前基于HSV空间的阴影检测", HSV_img_binary);
    waitKey(0);
    imwrite("./Output/合并前基于HSV空间的阴影检测.jpg", HSV_img_binary);
    Mat HSV_merge = Merge(HSV_img_binary, HSV);
    imshow("合并后基于HSV空间的阴影检测", HSV_merge);
    waitKey(0);
    imwrite("./Output/合并后基于HSV空间的阴影检测.jpg", HSV_merge);

    //基于C1C2C3彩色空间的阴影检测
    printf_s("运行基于C1C2C3彩色空间的阴影检测……\n");
    Mat C1C2C3_img = RGBToC1C2C3(channel[2], channel[1], channel[0]); // C1C2C3空间转换
    Mat C1C2C3_img_binary = Segment(C1C2C3_img,C1C2C3);
    imshow("合并前基于C1C2C3彩色空间的阴影检测",C1C2C3_img_binary);
    waitKey(0);
    imwrite("./Output/合并前基于C1C2C3彩色空间的阴影检测.jpg",C1C2C3_img_binary);
    Mat C1C2C3_Merge = Merge(C1C2C3_img_binary, C1C2C3);
    imshow("合并后基于C1C2C3彩色空间的阴影检测", C1C2C3_Merge);
    waitKey(0);
    imwrite("./Output/合并后基于C1C2C3彩色空间的阴影检测.jpg", C1C2C3_Merge);


}
```

3 实验结果与分析

3.1 基于特征指数的遥感专题信息提取

3.1.1 植被指数

①比值植被指数：

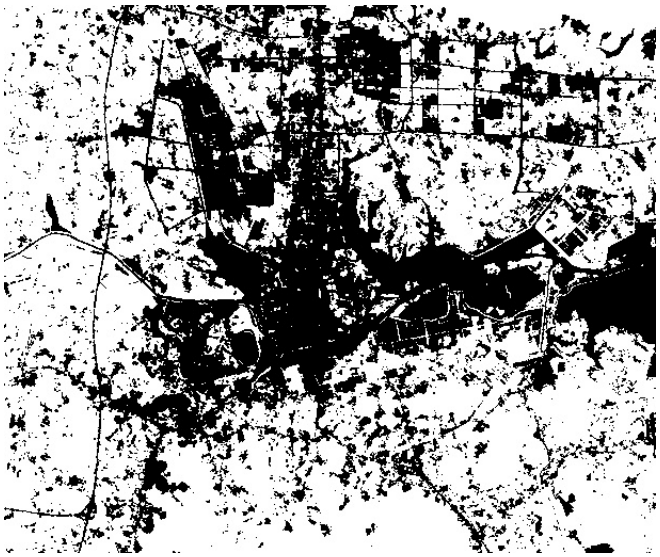

图 25：比值植被指数-二值化图

②归一化植被指数：

图 26：归一化植被指数-二值化图

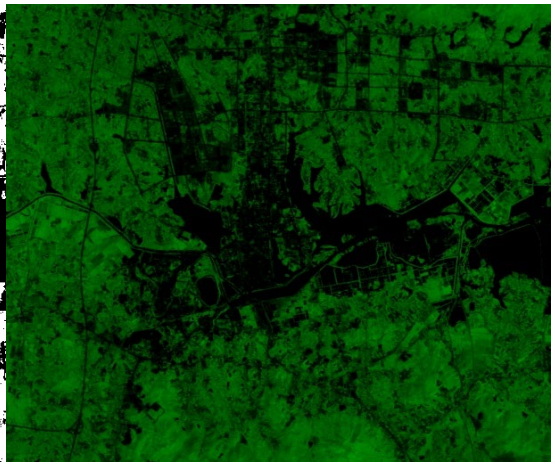

图 27：归一化植被指数-伪彩色图

③土壤调节植被指数：

图 28：土壤调节植被指数-二值化图

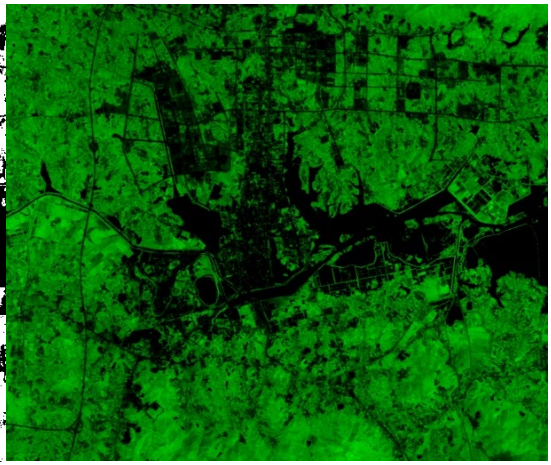

图 29：土壤调节植被指数-伪彩色图

3.2.2 水体指数

①归一化水体指数：

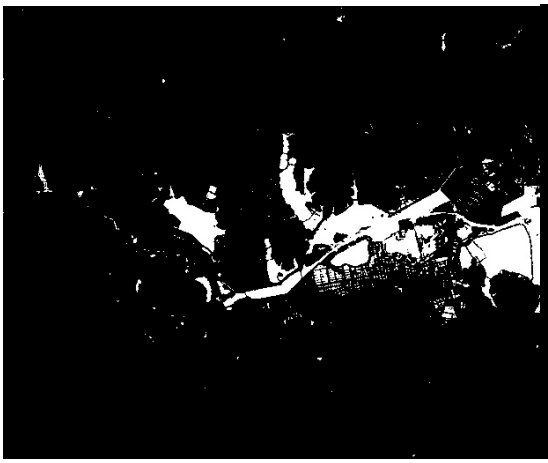

图 30：归一化水体指数-二值化图

图 31：归一化水体指数-伪彩色图

②自动水体提取指数：

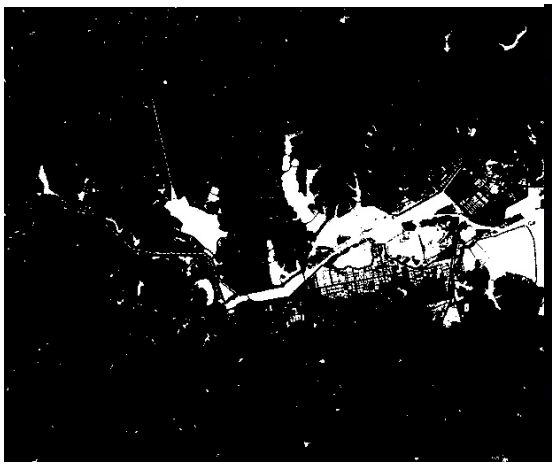

图 32：自动水体提取指数-二值化图

图 33：自动水体提取指数-伪彩色图

3.2.3 建筑指数

①归一化建筑指数：

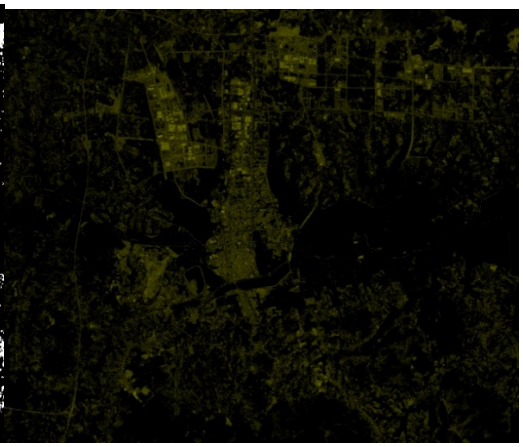

图 34：归一化建筑指数-二值化图　图 35：归一化建筑指数-伪彩色图

②建筑用地指数：

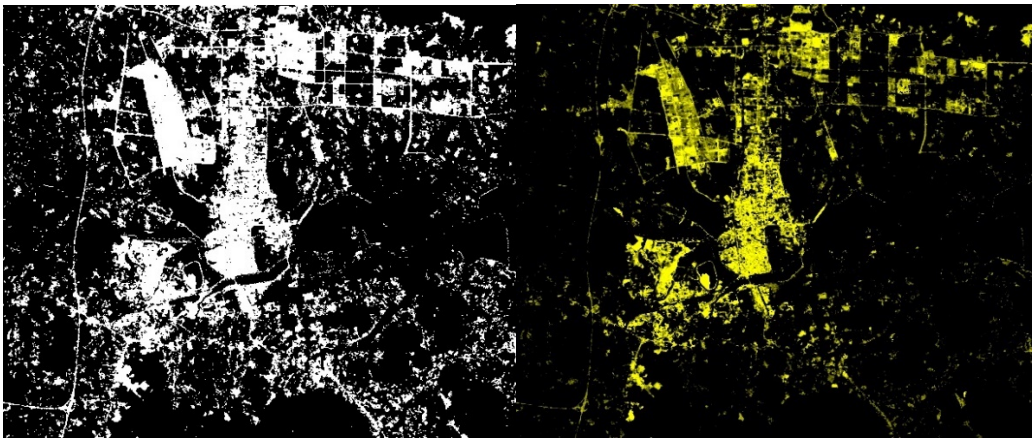

图 36：建筑用地指数-二值化图　　图 37：建筑用地指数-伪彩色图

3.2.4 真彩色合成

图 38：真彩色图

3.3 遥感影像阴影检测

①基于 HSV 空间的阴影检测：

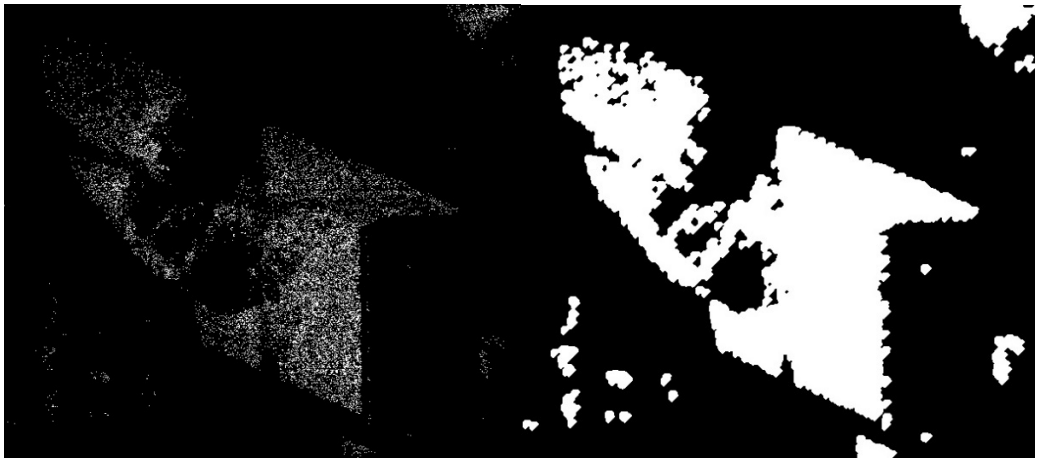

图 39：合并前结果（HSV）　　图 40：合并后结果（HSV）

②基于 C1C2C3 彩色空间的阴影检测：

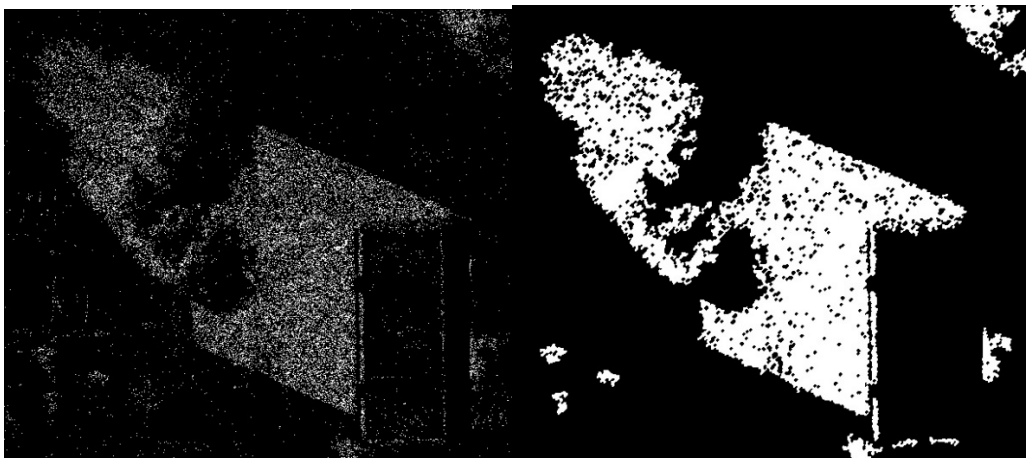

图 41：合并前结果（C1C2C3）　　图 42：合并后结果（C1C2C3）